

Abdelkader Hameurlain  
Stephen W. Liddle  
Klaus-Dieter Schewe  
Xiaofang Zhou (Eds.)

LNCS 6861

# Database and Expert Systems Applications

22nd International Conference, DEXA 2011  
Toulouse, France, August/September 2011  
Proceedings, Part II

**2** Part II

**DEXA 2011**

 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*

Abdelkader Hameurlain Stephen W. Liddle  
Klaus-Dieter Schewe Xiaofang Zhou (Eds.)

# Database and Expert Systems Applications

22nd International Conference, DEXA 2011  
Toulouse, France, August 29 - September 2, 2011  
Proceedings, Part II

## Volume Editors

Abdelkader Hameurlain

Paul Sabatier University, IRIT Institut de Recherche en Informatique de Toulouse  
118, route de Narbonne, 31062 Toulouse Cedex, France

E-mail: hameur@irit.fr

Stephen W. Liddle

Brigham Young University, 784 TNRB

Provo, UT 84602, USA

E-mail: liddle@byu.edu

Klaus-Dieter Schewe

Software Competence Centre Hagenberg

Softwarepark 21, 4232 Hagenberg, Austria

E-mail: kd.schewe@sch.at

Xiaofang Zhou

University of Queensland, School of Information Technology  
and Electrical Engineering

Brisbane QLD 4072, Australia

E-mail: zxf@uq.edu.au

ISSN 0302-9743

e-ISSN 1611-3349

ISBN 978-3-642-23090-5

e-ISBN 978-3-642-23091-2

DOI 10.1007/978-3-642-23091-2

Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011934284

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

© Springer-Verlag Berlin Heidelberg 2011

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.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

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

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

# Preface

This volume includes invited papers, research papers, and short papers presented at DEXA 2011, the 22<sup>nd</sup> International Conference on Database and Expert Systems Applications, held in Toulouse, France. DEXA 2011 continued the long and successful DEXA tradition begun in 1990, bringing together a large collection of bright researchers, scientists, and practitioners from around the world to share new results in the areas of database, intelligent systems, and related advanced applications.

The call for papers resulted in the submission of 207 papers, of which 52 were accepted as regular research papers, and 40 were accepted as short papers. The authors of these papers come from 47 different countries. These papers discuss a range of topics including:

- Query processing and Skyline queries
- Search (Web, Semantic Web, database)
- Data integration, transactions, optimization, and design
- Physical aspects of databases, storage
- Database semantics
- Security and privacy
- Spatial and temporal data
- Semantic Web
- Web applications
- Data mining
- Ontologies
- Distribution
- Information retrieval
- XML querying and views
- Business applications
- User support

Three internationally recognized scholars submitted papers and delivered keynote speeches:

Patrick Valduriez: Principles of Distributed Data Management 2020?

Bernhard Thalheim: The Science of Conceptual Modelling

Gabriele Kern-Isberner: Probabilistic Logics in Expert Systems: Approaches, Implementations, and Applications

In addition to the main conference track, DEXA 2011 also included 12 workshops that explored the conference theme within the context of life sciences, specific application areas, and theoretical underpinnings.

We are grateful to the hundreds of authors who submitted papers to DEXA 2011 and to our large Program Committee for the many hours they spent carefully reading and reviewing these papers. The Program Committee was also assisted by a number of external referees, and we appreciate their contributions and detailed comments.

We are thankful to the Institut de Recherche en Informatique de Toulouse at the Université Paul Sabatier for organizing DEXA 2011, and for the excellent working atmosphere provided. In particular, we recognize the efforts of the conference Organizing Committee, including Makoto Takizawa (Seikei University, Japan; Honorary Chairperson), Abdelkader Hameurlain (IRIT, Paul Sabatier University, France; General Chair), Riad Mokadem (IRIT, Paul Sabatier University; Local Organization), Vladimir Marik (Czech Technical University, Czech Republic; Publication Chair), Franck Morvan (IRIT, Paul Sabatier University, Toulouse, France; Workshops Co-chair), A Min Tjoa (Technical University of Vienna, Austria; Workshops Co-chair), and Roland R. Wagner (FAW, University of Linz, Austria; Workshops Co-chair). Without the diligent efforts of these people, DEXA 2011 would not have been possible.

Finally, we are especially grateful to Gabriela Wagner, whose professional attention to detail and skillful handling of all aspects of the Program Committee management and proceedings preparation was most helpful.

August 2011

Stephen W. Liddle  
Klaus-Dieter Schewe  
Xiaofang Zhou

# Program Committee

## Honorary Chairperson

Makoto Takizawa                      Seikei University, Japan

## General Chair

Abdelkader Hameurlain              IRIT, Paul Sabatier University, Toulouse,  
France

## Conference Program Chair

Stephen Liddle                      Brigham Young University, USA  
Klaus-Dieter Schewe                Software Competence Center Hagenberg and  
Johannes Kepler University Linz, Austria  
Xiaofang Zhou                      University of Queensland, Australia

## Program Committee

Witold Abramowicz	The Poznan University of Economics, Poland
Osman Abul	TOBB University, Turkey
Rafael Accorsi	University of Freiburg, Germany
Hamideh Afsarmanesh	University of Amsterdam, The Netherlands
Riccardo Albertoni	CNR-IMATI-GE, Italy
Toshiyuki Amagasa	University of Tsukuba, Japan
Rachid Anane	Coventry University, UK
Annalisa Appice	Università degli Studi di Bari, Italy
Mustafa Atay	Winston-Salem State University, USA
James Bailey	University of Melbourne, Australia
Spiridon Bakiras	City University of New York, USA
Ladjel Bellatreche	ENSMA-Poitiers University, France
Morad Benyoucef	University of Ottawa, Canada
Catherine Berrut	Grenoble University, France
Bishwaranjan Bhattacharjee	IBM Thomas J. Watson Research Center, USA
Debmalya Biswas	SAP Research, Germany
Agustinus Borgy Waluyo	Institute for Infocomm Research, Singapore
Patrick Bosc	IRISA/ENSSAT, France
Athman Bouguettaya	CSIRO, Australia
Danielle Boulanger	University of Lyon, France
Omar Boussaid	University of Lyon, France

Stephane Bressan	National University of Singapore, Singapore
Patrick Brezillon	University Paris VI, France
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
Cindy Chen	University of Massachusetts Lowell, USA
Phoebe Chen	La Trobe University, Australia
Shu-Ching Chen	Florida International University, USA
Hao Cheng	University of Central Florida, USA
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
Gao Cong	Microsoft Research Asia, China
Oscar Corcho	Universidad Politécnica de Madrid, Spain
Bin Cui	Peking University, China
Emiran Curtmola	University of California, San Diego, USA
Alfredo Cuzzocrea	University of Calabria, Italy
Deborah Dahl	Conversational Technologies, worldwide
Jérôme Darmont	Université Lumière Lyon 2, France
Valeria De Antonellis	Università di Brescia, Italy
Andre de Carvalho	University of Sao Paulo, Brazil
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
Dirk Draheim	Universität Innsbruck, Austria



Cedric du Mouza	CNAM, France
Johann Eder	University of Vienna, Austria
David Embley	Brigham Young University, USA
Suzanne M. Embury	The University of Manchester, UK
Christian Engelmann	Oak Ridge National Laboratory, USA
Bettina Fazzinga	University of Calabria, Italy
Leonidas Fegaras	The University of Texas at Arlington, USA
Flavio Ferrararotti	Victoria University of Wellington, New Zealand
Stefano Ferilli	University of Bari, Italy
Eduardo Fernandez	Florida Atlantic University, USA
Filomena Ferrucci	Università di Salerno, Italy
Flavius Frasinca	Erasmus University Rotterdam, The Netherlands
Bernhard Freudenthaler	Software Competence Center Hagenberg, Austria
Hiroaki Fukuda	Shibaura Institute of Technology, Japan
Steven Furnell	University of Plymouth, UK
Aryya Gangopadhyay	University of Maryland Baltimore County, USA
Yunjun Gao	Zhejiang University, China
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
William Grosky	University of Michigan, USA
Volker Gruhn	Leipzig University, Germany
Jerzy Grzymala-Busse	University of Kansas, USA
Francesco Guerra	Università degli Studi di Modena e Reggio Emilia, Italy
Giovanna Guerrini	University of Genova, Italy
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
Francisco Herrera	University of Granada, Spain
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	Semantic Identity, 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
Ejub Kajan	State University of Novi Pazar, Serbia
Anne Kao	Boeing Phantom Works, USA
Stefan Katzenbeisser	Technical University of Darmstadt, Germany
Yiping Ke	Chinese University of Hong Kong, Hong Kong
Sang-Wook Kim	Hanyang University, Korea
Markus Kirchberg	Hewlett-Packard Laboratories, Singapore
Hiroyuki Kitagawa	University of Tsukuba, Japan
Carsten Kleiner	University of Applied Sciences and Arts Hannover, Germany
Ibrahim Korpeoglu	Bilkent University, Turkey
Harald Kosch	University of Passau, Germany
Michal Krátký	VSB-Technical University of Ostrava, Czech Republic
Arun Kumar	IBM India Research Lab., India
Ashish Kundu	Purdue University, USA
Josef Küng	University of Linz, Austria
Kwok-Wa Lam	University of Hong Kong, Hong Kong
Nadira Lammari	CNAM, France
Gianfranco Lamperti	University of Brescia, Italy
Anne Laurent	LIRMM, Université Montpellier 2, France
Mong Li Lee	National University of Singapore, Singapore
Alain Toinon Leger	Orange - France Telecom R&D, France
Daniel Lemire	Université du Québec à Montréal, Canada
Pierre Lévy	Public Health Department, France
Lenka Lhotska	Czech Technical University, Czech Republic
Wenxin Liang	Dalian University of Technology, China
Stephen W. Liddle	Brigham Young University, USA
Lipyeow Lim	IBM T.J. Watson Research Center, 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
Jianguo Lu	University of Windsor, Canada
Alessandra Lumini	University of Bologna, Italy
Cheng Luo	Coppin State University, USA
Hui Ma	Victoria University of Wellington, New Zealand
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
Pierre-Francois Marteau	Université de Bretagne Sud, France
Elio Masciari	ICAR-CNR, Italy
Norman May	SAP Research Center, Germany
Jose-Norberto Mazon	University of Alicante, Spain
Dennis McLeod	University of Southern California, USA
Brahim Medjahed	University of Michigan - Dearborn, USA
Harekrishna Misra	Institute of Rural Management Anand, India
Jose Mocito	INESC-ID/FCUL, Portugal
Lars Mönch	FernUniversität in Hagen, Germany
Riad Mokadem	IRIT, Paul Sabatier University, France
Yang-Sae Moon	Kangwon National University, Korea
Reagan Moore	San Diego Supercomputer Center, USA
Franck Morvan	IRIT, Paul Sabatier University, Toulouse, France
Mirco Musolesi	University of Cambridge, UK
Ismael Navas-Delgado	University of Málaga, Spain
Wilfred Ng	University of Science and Technology, Hong Kong
Javier Nieves Acedo	Deusto University, Spain
Selim Nurcan	University Paris 1 Pantheon Sorbonne, France
Mehmet Orgun	Macquarie University, Australia
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
Marcin Paprzycki	Polish Academy of Sciences, Warsaw Management Academy, Poland
Oscar Pastor	Universidad Politecnica de Valencia, Spain
Jovan Pehcevski	MIT University, Skopje, Macedonia
Reinhard Pichler	Technische Universität Wien, Austria
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
Gerald Quirchmayr	University of Vienna, Austria and University of South Australia, Australia
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
Rodolfo F. Resende	Federal University of Minas Gerais, Brazil
Claudia Roncancio	Grenoble University / LIG, France
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
Shazia Sadiq	University of Queensland, Australia
Simonas Saltenis	Aalborg University, Denmark
Demetrios G Sampson	University of Piraeus, Greece
Carlo Sansone	Università di Napoli "Federico II", 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
Klaus-Dieter Schewe	Software Competence Centre Hagenberg, Austria
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
Leonid Sokolinsky	South Ural State University, Russia
Bala Srinivasan	Monash University, Australia
Umberto Straccia	Italian National Research Council, Italy
Darijus Strasunskas	Norwegian University of Science and Technology (NTNU), Norway
Lena Stromback	Linköpings Universitet, Sweden
Aixin Sun	Nanyang Technological University, Singapore
Raj Sunderraman	Georgia State University, USA
Ashish Sureka	Infosys Technologies Limited, India
David Taniar	Monash University, Australia
Cui Tao	Brigham Young University, USA

Maguelonne Teisseire	LIRMM, Université 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
Bernhard Thalheim	Christian-Albrechts-Universität Kiel, Germany
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
Vicenc Torra	Universitat Autònoma de Barcelona, Spain
Farouk Toumani	Université Blaise Pascal, France
Traian Truta	Northern Kentucky University, USA
Vassileios Tsetsos	National and Kapodistrian University of Athens, Greece
Theodoros Tzouramanis	University of the Aegean, Greece
Roberto Uribeetxebarria	Mondragon University, Spain
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 Mannheim, Germany
Jianyong Wang	Tsinghua University, China
Junhu Wang	Griffith University, Brisbane, Australia
Kewen Wang	Griffith University, Brisbane, Australia
Qing Wang	University of Otago, Dunedin, New Zealand
Wei Wang	University of New South Wales, Sydney, Australia
Wendy Hui Wang	Stevens Institute of Technology, USA
Andreas Wombacher	University of Twente, The Netherlands
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
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
Xiaofang Zhou	University of Queensland, Australia
Qiang Zhu	The University of Michigan, USA

Yan Zhu Southwest Jiaotong University,  
Chengdu, China  
Urko Zurutuza Mondragon University, Spain

## External Reviewers

Mohamad Othman Abdallah University of Grenoble, LIG, France  
 Sandra de Amo University of Uberlandia, Brazil  
 Laurence Rodrigues do Amaral Federal University of Goias, Brazil  
 Sandra de Amo Federal University of Uberlandia, Brazil  
 Diego Arroyuelo Yahoo! Research Latin America, Chile  
 Tigran Avanesov INRIA Nancy Grand Est, France  
 Ali Bahrami Boeing, USA  
 Zhifeng Bao National University of Singapore, Singapore  
 Devis Bianchini University of Brescia, Italy  
 Souad Boukhadouma University of USTHB, Algeria  
 Paolo Bucciol UDLAP, LAFMIA, Mexico  
 Diego Ceccarelli ISTI-CNR Pisa, Italy  
 Camelia Constantin Université Pierre et Marie Curie-Paris VI,  
Paris, France  
 Yan Da University of Science and Technology,  
Hong Kong  
 Matthew Damigos NTUA, Greece  
 Franca Debole ISTI-CNR Pisa, Italy  
 Paul de Vreize Bournemouth University, UK  
 Raimundo F. DosSantos Virginia Tech, USA  
 Gilles Dubois MODEME, University of Lyon-Jean  
Moulin Lyon 3, France  
 Qiong Fang University of Science and Technology,  
Hong Kong  
 Fabio Fassetti DEIS, University of Calabria, Italy  
 Ingo Feinerer Vienna University of Technology, Austria  
 Daniel Fleischhacker University of Mannheim, Germany  
 Fausto Fleites Florida International University, USA  
 Filippo Furfaro DEIS, University of Calabria, Italy  
 Claudio Gennaro ISTI-CNR Pisa, Italy  
 Panagiotis Gouvas Ubitech EU  
 Carmine Gravino University of Salerno, Italy  
 Hsin-Yu Ha Florida International University, USA  
 Allel Hadj-Ali IRISA-University of Rennes 1, France  
 Duong Hoang Vienna University of Technology, Austria  
 Enrique de la Hoz Universidad de Alcala, Spain  
 Hai Huang Swinburne University of Technology, Australia  
 Gilles Hubert IRIT, Université Paul Sabatier, Université de  
Toulouse, France  
 Mohammad Rezwatul Huq University of Twente, The Netherlands

Saiful Islam	Swinburne University of Technology, Australia
Min-Hee Jang	Hanyang University, Korea
Hélène Jaudoin	IRISA-University of Rennes 1, France
Lili Jiang	Lanzhou University, China
Selma Khouri	ESI, Algeria
Emin Yigit Koksal	Bilkent University, Turkey
Theodorich Kopetzky	SCCH - Software Competence Center Hagenberg, Austria
Olivier le-Goaer	University of Pau, France
Paea LePendu	Stanford University, USA
Kenneth Leung	University of Science and Technology, Hong Kong
Wenxin Liang	Dalian University of Technology, China
Haishan Liu	University of Oregon, USA
Rosanne Liu	The University of Michigan - Dearborn, USA
Xuezheng Liu	Google Inc., USA
Xutong Liu	Virginia Tech, USA
Yannick Loiseau	LIMOS, Blaise Pascal University, France
Carlos Manuel López Enríquez	Grenoble INP - UDLAP, LIG-LAFMIA, France
Min Luo	Tokyo Institute of Technology, Japan
Laura Maag	Alcatel-Lucent Bell-Labs, France
Lucrezia Macchia	University of Bari, Italy
Ivan Marsa-Maestre	Universidad de Alcala, Spain
Michele Melchiori	University of Brescia, Italy
Ruslan Miniakhmetov	South Ural State University, Chelyabinsk, Russia
Ronald Ocampo	Florida International University, USA
Anna Oganyan	Georgia Southern University, USA
Ermelinda Oro	ICAR-CNR, Italy
Francesco Pagliarecci	Università Politecnica delle Marche, Italy
Constantin Pan	South Ural State University, Chelyabinsk, Russia
Vassia Pavlaki	NTUA, Greece
Olivier Pivert	IRISA-University of Rennes 1, France
Xu Pu	Tsinghua University, China
Gang Qian	University of Central Oklahoma, USA
Michele Risi	University of Salerno, Italy
Flavio Rizzolo	Business Intelligence Network (BIN), Canada
Edimilson Batista dos Santos	Federal University of Rio de Janeiro, Brazil
Federica Sarro	University of Salerno, Italy
Vadim Savenkov	Vienna University of Technology, Austria
Wei Shen	Tsinghua University, China
Grégory Smits	IRISA-University of Rennes 1, France
Sofia Stamou	Ionian University, Corfu, Greece

Lubomir Stanchev	Indiana University - Purdue University Fort Wayne, USA
Chad M.S. Steel	Virginia Tech, USA
Thomas Stocker	University of Freiburg, Germany
Nick Amirreza Tahamtan	Vienna University of Technology, Austria
Guilaine Talens	MODEME, University of Lyon-Jean Moulin Lyon 3, France
Lu-An Tang	University of Illinois at Urbana Champaign, USA
Caglar Terzi	Bilkent University, Turkey
Gabriel Tolosa	National University of Lujan, Argentina
Claudio Vairo	ISTI-CNR Pisa, Italy
Changzhou Wang	Boeing, USA
Yousuke Watanabe	Tokyo Institute of Technology, Japan
Andreas Weiner	University of Kaiserslautern, Germany
Yimin Yang	Florida International University, USA
Soek-Ho Yoon	Hanyang University, Korea
Sira Yongchareon	Swinburne University of Technology, Australia
Tomoki Yoshihisa	Osaka University, Japan
Jing Yuan	University of Science and Technology of China, China
Chao Zhu	The University of Michigan - Dearborn, USA
Mikhail Zymbler	South Ural State University, Chelyabinsk, Russia



## Table of Contents – Part II

### XML Querying and Views

On Equivalence and Rewriting of XPath Queries Using Views under DTD Constraints . . . . .	1
<i>Pantelis Aravogiadis and Vasilis Vassalos</i>	
Incremental Maintenance of Materialized XML Views . . . . .	17
<i>Leonidas Fegaras</i>	
Ingredients for Accurate, Fast, and Robust XML Similarity Joins . . . . .	33
<i>Leonardo Andrade Ribeiro and Theo Härder</i>	
Twig Pattern Matching: A Revisit . . . . .	43
<i>Jiang Li, Junhu Wang, and Maolin Huang</i>	
Boosting Twig Joins in Probabilistic XML . . . . .	51
<i>Siqi Liu and Guoren Wang</i>	

### Data Mining

Prediction of Cerebral Aneurysm Rupture Using Hemodynamic, Morphologic and Clinical Features: A Data Mining Approach . . . . .	59
<i>Jesus Bisbal, Gerhard Engelbrecht, Mari-Cruz Villa-Uriol, and Alejandro F. Frangi</i>	
Semantic Translation for Rule-Based Knowledge in Data Mining . . . . .	74
<i>Dejing Dou, Han Qin, and Haishan Liu</i>	
Mining Frequent Disjunctive Selection Queries . . . . .	90
<i>Inès Hilali-Jaghdam, Tao-Yuan Jen, Dominique Laurent, and Sadok Ben Yahia</i>	
A Temporal Data Mining Framework for Analyzing Longitudinal Data . . . . .	97
<i>Corrado Loglisci, Michelangelo Ceci, and Donato Malerba</i>	
How to Use “Classical” Tree Mining Algorithms to Find Complex Spatio-Temporal Patterns? . . . . .	107
<i>Nazha Selmaoui-Folcher and Frédéric Flouvat</i>	

## Queries and Search

Inferring Fine-Grained Data Provenance in Stream Data Processing: Reduced Storage Cost, High Accuracy . . . . .	118
<i>Mohammad Rezwanul Huq, Andreas Wombacher, and Peter M.G. Apers</i>	
Approximate Query on Historical Stream Data . . . . .	128
<i>Qiyang Duan, Peng Wang, MingXi Wu, Wei Wang, and Sheng Huang</i>	
An Incremental Approach to Closest Pair Queries in Spatial Networks Using Best-First Search . . . . .	136
<i>Chunan Chen, Weiwei Sun, Baihua Zheng, Dingding Mao, and Weimo Liu</i>	
Fast Top-K Query Answering . . . . .	144
<i>Claus Dabringer and Johann Eder</i>	
Towards an On-Line Analysis of Tweets Processing . . . . .	154
<i>Sandra Bringay, Nicolas Béchet, Flavien Bouillot, Pascal Poncelet, Mathieu Roche, and Maguelonne Teisseire</i>	
The Fix-Point Method for Discrete Events Simulation Using SQL and UDF . . . . .	162
<i>Qiming Chen, Meichun Hsu, and Bin Zhang</i>	

## Semantic Web

Approximate and Incremental Processing of Complex Queries against the Web of Data . . . . .	171
<i>Thanh Tran, Günter Ladwig, and Andreas Wagner</i>	
Conjunctive Query Optimization in OWL2-DL . . . . .	188
<i>Petr Křemen and Zdeněk Kouba</i>	
RoSeS: A Continuous Content-Based Query Engine for RSS Feeds . . . . .	203
<i>Jordi Creus Tomàs, Bernd Amann, Nicolas Travers, and Dan Vodislav</i>	

## Information Retrieval

The Linear Combination Data Fusion Method in Information Retrieval . . . . .	219
<i>Shengli Wu, Yaxin Bi, and Xiaoqin Zeng</i>	
Approaches and Standards for Metadata Interoperability in Distributed Image Search and Retrieval . . . . .	234
<i>Ruben Tous, Jordi Nin, Jaime Delgado, and Pere Toran</i>	

A Distributed Architecture for Flexible Multimedia Management and Retrieval . . . . .	249
<i>Mihaela Brut, Dana Codreanu, Stefan Dumitrescu, Ana-Maria Manzat, and Florence Sedes</i>	

## Business Applications

Deontic BPMN . . . . .	264
<i>Christine Natschläger</i>	
Improving Stock Market Prediction by Integrating Both Market News and Stock Prices . . . . .	279
<i>Xiaodong Li, Chao Wang, Jiawei Dong, Feng Wang, Xiaotie Deng, and Shanfeng Zhu</i>	
Querying Semantically Enriched Business Processes . . . . .	294
<i>Michele Missikoff, Maurizio Proietti, and Fabrizio Smith</i>	
Introducing Affective Agents in Recommendation Systems Based on Relational Data Clustering . . . . .	303
<i>João C. Xavier-Junior, Alberto Signoretti, Anne M.P. Canuto, Andre M. Campos, Luiz M.G. Gonçalves, and Sergio V. Fialho</i>	
Converting Conversation Protocols Using an XML Based Differential Behavioral Model . . . . .	311
<i>Claas Busemann and Daniela Nicklas</i>	

## User Support

Facilitating Casual Users in Interacting with Linked Data through Domain Expertise . . . . .	319
<i>Cormac Hampson and Owen Conlan</i>	
Using Expert-Derived Aesthetic Attributes to Help Users in Exploring Image Databases . . . . .	334
<i>Cormac Hampson, Meltem Gürel, and Owen Conlan</i>	

## Indexing

SkyMap: A Trie-Based Index Structure for High-Performance Skyline Query Processing . . . . .	350
<i>Joachim Selke and Wolf-Tilo Balke</i>	
A Path-Oriented RDF Index for Keyword Search Query Processing . . . . .	366
<i>Paolo Cappellari, Roberto De Virgilio, Antonio Maccioni, and Mark Roantree</i>	

Variable Length Compression for Bitmap Indices . . . . . 381  
*Fabian Corrales, David Chiu, and Jason Sawin*

**Queries, Views and Data Warehouses**

Modeling View Selection as a Constraint Satisfaction Problem . . . . . 396  
*Imene Mami, Remi Coletta, and Zohra Bellahsene*

Enabling Knowledge Extraction from Low Level Sensor Data . . . . . 411  
*Paolo Cappellari, Jie Shi, Mark Roantree, Crionna Tobin, and Niall Moyna*

Join Selectivity Re-estimation for Repetitive Queries in Databases . . . . . 420  
*Feng Yu, Wen-Chi Hou, Cheng Luo, Qiang Zhu, and Dunren Che*

Matching Star Schemas . . . . . 428  
*Dariush Riazati and James A. Thom*

**Ontologies**

Automated Construction of Domain Ontology Taxonomies from Wikipedia 439  
*Damir Jurić, Marko Banek, and Zoran Skočir*

Storing Fuzzy Ontology in Fuzzy Relational Database . . . . . 447  
*Fu Zhang, Z.M. Ma, Li Yan, and Jingwei Cheng*

Using an Ontology to Automatically Generate Questions for the Determination of Situations . . . . . 456  
*Marten Teitsma, Jacobijn Sandberg, Marinus Maris, and Bob Wielinga*

**Physical Aspects of Databases**

Indexing Frequently Updated Trajectories of Network-Constrained Moving Objects . . . . . 464  
*Zhiming Ding*

Online Index Selection in RDBMS by Evolutionary Approach . . . . . 475  
*Piotr Kolaczowski and Henryk Rybiński*

Towards Balanced Allocations for DHTs . . . . . 485  
*George Tsatsanifos and Vasilis Samoladas*

Caching Stars in the Sky: A Semantic Caching Approach to Accelerate Skyline Queries . . . . . 493  
*Arnab Bhattacharya, B. Palvali Teja, and Sourav Dutta*

**Design**

Generating Synthetic Database Schemas for Simulation Purposes . . . . .	502
<i>Carlos Eduardo Pires, Priscilla Vieira, Márcio Saraiva, and Denilson Barbosa</i>	
A New Approach for Fuzzy Classification in Relational Databases . . . . .	511
<i>Ricardo Hideyuki Tajiri, Eduardo Zanoní Marques, Bruno Bogaz Zarpelão, and Leonardo de Souza Mendes</i>	
Anomaly Detection for the Prediction of Ultimate Tensile Strength in Iron Casting Production . . . . .	519
<i>Igor Santos, Javier Nieves, Xabier Ugarte-Pedrero, and Pablo G. Bringas</i>	

**Distribution**

<i>LinkedPeers: A Distributed System for Interlinking Multidimensional Data</i> . . . . .	527
<i>Athanasia Asiki, Dimitrios Tsoumakos, and Nectarios Koziris</i>	
A Vertical Partitioning Algorithm for Distributed Multimedia Databases . . . . .	544
<i>Lisbeth Rodriguez and Xiaouu Li</i>	
Diffusion in Dynamic Social Networks: Application in Epidemiology . . . . .	559
<i>Erick Stattner, Martine Collard, and Nicolas Vidot</i>	

**Miscellaneous Topics**

Probabilistic Quality Assessment Based on Article’s Revision History . . . . .	574
<i>Jingyu Han, Chuandong Wang, and Dawei Jiang</i>	
Propagation of Multi-granularity Annotations . . . . .	589
<i>Ryo Aoto, Toshiyuki Shimizu, and Masatoshi Yoshikawa</i>	
<b>Author Index</b> . . . . .	605

# Table of Contents – Part I

## Keynote Talks

Principles of Distributed Data Management in 2020? .....	1
<i>Patrick Valduriez</i>	
The Science of Conceptual Modelling .....	12
<i>Bernhard Thalheim</i>	
Probabilistic Logics in Expert Systems: Approaches, Implementations, and Applications .....	27
<i>Gabriele Kern-Isberner, Christoph Beierle, Marc Finthammer, and Matthias Thimm</i>	

## Query Processing

Multi-objective Optimal Combination Queries .....	47
<i>Xi Guo and Yoshiharu Ishikawa</i>	
NEFOS: Rapid Cache-Aware Range Query Processing with Probabilistic Guarantees .....	62
<i>Spyros Sioutas, Kostas Tsichlas, Ioannis Karydis, Yannis Manolopoulos, and Yannis Theodoridis</i>	
Reuse-Oriented Mapping Discovery for Meta-querier Customization ....	78
<i>Xiao Li and Randy Chow</i>	

## Database Semantics

Attribute Grammar for XML Integrity Constraint Validation .....	94
<i>Béatrice Bouchou, Mirian Halfeld Ferrari, and Maria Adriana Vidigal Lima</i>	
Extracting Temporal Equivalence Relationships among Keywords from Time-Stamped Documents .....	110
<i>Parvathi Chundi, Mahadevan Subramaniam, and R.M. Aruna Weerakoon</i>	
Codd Table Representations under Weak Possible World Semantics .....	125
<i>Flavio Ferrarotti, Sven Hartmann, Van Bao Tran Le, and Sebastian Link</i>	

**Skyline Queries**

Efficient Early Top- $k$ Query Processing in Overloaded P2P Systems . . . .	140
<i>William Kokou Dédzoé, Philippe Lamarre, Reza Akbarinia, and Patrick Valduriez</i>	
Top- $k$ Query Evaluation in Sensor Networks with the Guaranteed Accuracy of Query Results . . . . .	156
<i>Baichen Chen, Weifa Liang, and Geyong Min</i>	
Learning Top- $k$ Transformation Rules . . . . .	172
<i>Sunanda Patro and Wei Wang</i>	

**Security and Privacy**

Privacy beyond Single Sensitive Attribute . . . . .	187
<i>Yuan Fang, Mafruz Zaman Ashrafi, and See Kiong Ng</i>	
Privacy-Aware DaaS Services Composition . . . . .	202
<i>Salah-Eddine Tbahriti, Michael Mrissa, Brahim Medjahed, Chirine Ghedira, Mahmoud Barhamgi, and Jocelyne Fayn</i>	
An Empirical Study on Using the National Vulnerability Database to Predict Software Vulnerabilities . . . . .	217
<i>Su Zhang, Doina Caragea, and Xinming Ou</i>	

**Spatial and Temporal Data**

Similar Subsequence Search in Time Series Databases . . . . .	232
<i>Shrikant Kashyap, Mong Li Lee, and Wynne Hsu</i>	
Optimizing Predictive Queries on Moving Objects under Road-Network Constraints . . . . .	247
<i>Lasanthi Heendaliya, Dan Lin, and Ali Hurson</i>	
Real-Time Capable Data Management Architecture for Database-Driven 3D Simulation Systems . . . . .	262
<i>Jürgen Roßmann, Michael Schluse, Ralf Waspe, and Martin Hoppen</i>	
Collecting and Managing Network-Matched Trajectories of Moving Objects in Databases . . . . .	270
<i>Zhiming Ding and Ke Deng</i>	
On Guaranteeing $k$ -Anonymity in Location Databases . . . . .	280
<i>Anh Tuan Truong, Tran Khanh Dang, and Josef Küng</i>	

## Semantic Web Search

Smeagol: A “Specific-to-General” Semantic Web Query Interface Paradigm for Novices . . . . .	288
<i>Aaron Clemmer and Stephen Davies</i>	
Browsing-Oriented Semantic Faceted Search . . . . .	303
<i>Andreas Wagner, Günter Ladwig, and Thanh Tran</i>	
An Efficient Algorithm for Topic Ranking and Modeling Topic Evolution . . . . .	320
<i>Kumar Shubhankar, Aditya Pratap Singh, and Vikram Pudi</i>	
Sampling the National Deep Web . . . . .	331
<i>Denis Shestakov</i>	
A Bipartite Graph Model and Mutually Reinforcing Analysis for Review Sites . . . . .	341
<i>Kazuki Tawaramoto, Junpei Kawamoto, Yasuhito Asano, and Masatoshi Yoshikawa</i>	

## Storage and Search

Genetic Algorithm for Finding Cluster Hierarchies . . . . .	349
<i>Christian Böhm, Annahita Oswald, Christian Richter, Bianca Wackersreuther, and Peter Wackersreuther</i>	
A File Search Method Based on Intertask Relationships Derived from Access Frequency and RMC Operations on Files . . . . .	364
<i>Yi Wu, Kenichi Otagiri, Yousuke Watanabe, and Haruo Yokota</i>	
A General Top-k Algorithm for Web Data Sources . . . . .	379
<i>Mehdi Badr and Dan Vodislav</i>	
Improving the Quality of Web Archives through the Importance of Changes . . . . .	394
<i>Myriam Ben Saad and Stéphane Gançarski</i>	

## Web Search

Alternative Query Generation for XML Keyword Search and Its Optimization . . . . .	410
<i>Tetsutaro Motomura, Toshiyuki Shimizu, and Masatoshi Yoshikawa</i>	
K-Graphs: Selecting Top-k Data Sources for XML Keyword Queries . . . .	425
<i>Khanh Nguyen and Jinli Cao</i>	



Detecting Economic Events Using a Semantics-Based Pipeline . . . . .	440
<i>Alexander Hogenboom, Frederik Hogenboom, Flavius Frasinca,</i> <i>Uzay Kaymak, Otto van der Meer, and Kim Schouten</i>	
Edit Distance between XML and Probabilistic XML Documents . . . . .	448
<i>Ruiming Tang, Huayu Wu, Sadegh Nobari, and Stéphane Bressan</i>	
Towards an Automatic Characterization of Criteria . . . . .	457
<i>Benjamin Duthil, François Troussel, Mathieu Roche, Gérard Dray,</i> <i>Michel Plantié, Jacky Montmain, and Pascal Poncelet</i>	

## Data Integration, Transactions and Optimization

A Theoretical and Experimental Comparison of Algorithms for the Containment of Conjunctive Queries with Negation . . . . .	466
<i>Khalil Ben Mohamed, Michel Leclère, and Marie-Laure Mugnier</i>	
Data Integration over NoSQL Stores Using Access Path Based Mappings . . . . .	481
<i>Olivier Curé, Robin Hecht, Chan Le Duc, and Myriam Lamolle</i>	
An Energy-Efficient Concurrency Control Algorithm for Mobile Ad-Hoc Network Databases . . . . .	496
<i>Zhaowen Xing and Le Gruenwald</i>	

## Web Applications

An Ontology-Based Method for Duplicate Detection in Web Data Tables . . . . .	511
<i>Patrice Buche, Juliette Dibia-Barthélemy, Rania Khefifi, and</i> <i>Fatiha Saïs</i>	
Approaches for Semantically Annotating and Discovering Scientific Observational Data . . . . .	526
<i>Huiping Cao, Shawn Bowers, and Mark P. Schildhauer</i>	
A Scalable Tag-Based Recommender System for New Users of the Social Web . . . . .	542
<i>Valentina Zanardi and Licia Capra</i>	
<b>Author Index . . . . .</b>	<b>559</b>

# On Equivalence and Rewriting of XPath Queries Using Views under DTD Constraints

Pantelis Aravogiadis and Vasilis Vassalos

Department of Informatics, Athens University of Economics and Business,  
Athens, Greece

**Abstract.** It has long been recognized that query rewriting techniques are important tools for query optimization and semantic caching and are at the heart of data integration systems. In particular, the problem of rewriting queries using view definitions has received a lot of attention in these contexts. At the same time, the XPath language has become very popular for processing XML data, and there is much recent progress in semantic XPath optimization problems, such as XPath containment, and, more recently, XPath rewriting using views. In this paper we address the open problems of finding equivalent query rewritings using views for XPath queries and views that include the child, predicate and wildcard features (i.e., they are in  $XP(/, [], *)$ ) under DTD constraints. In the process, we also develop novel containment tests for queries in  $XP(/, [], *)$  under DTD constraints.

**Keywords:** XML data, XPath processing, query rewriting using views.

## 1 Introduction

XML has become a widely used standard for data representation and exchange over the Internet. To address the increasing need to query and manipulate XML data, the W3C has proposed the XML query language XQuery [7]. XPath [6] is the XQuery subset for navigating XML documents, and is designed to be embedded in a host language such as XQuery. XPath expressions often define complicated navigation, resulting in expensive query processing. Numerous techniques to speed up evaluation of XPath queries have been developed (e.g. [8]). Rewriting queries using materialized views is known to provide a powerful tool for query optimization, semantic caching, and data integration, for both relational and semistructured data (e.g. see [12,19]), and consequently the rewriting challenge for XPath queries and views has started receiving more attention [13,14,23,26].

The rewriting query using views problem is traditionally formulated in two different ways. In the *equivalent rewriting* formulation, the problem is, given a query  $q$  and a view  $v$ , to find a rewriting of  $q$  using  $v$  that is equivalent to  $q$ . This formulation is motivated by classical query optimization and solutions to the problem address the problem of speeding up XPath query evaluation [13,26]. The *maximally-contained rewriting* problem is the problem of finding, given a query  $q$  and a view  $v$ , a rewriting of  $q$  using  $v$  that is contained in  $q$  and is

maximal. This problem arises mostly in the context of information integration, where we cannot always find an equivalent rewriting due to limited coverage of data sources [14,23].

Even though there is important progress in the equivalent rewriting problem for various fragments of XPath (see [1,26] and also Sections 3 and 5 for more discussion), existing work assumes the absence of schema constraints. Given the many business, scientific and other uses of XML where schema information is widely available, the problem of equivalent query rewriting in the presence of a schema is at least equally important. Using Document Type Definitions (DTDs) as our schema language, we present the following results:

- Novel sound and complete algorithms for the containment and the query rewriting problem for XPath queries and views in  $XP(/, [], *)$  under a duplicate-free DTD or an acyclic and choice-free DTD. The algorithms are based on a novel containment test relying on the extraction of appropriate constraints from the DTD.
- While the above mentioned algorithm is in PTIME for choice-free DTD, it is in EXPTIME for duplicate-free DTD. Hence we also develop sound polynomial algorithms for the containment and the XPath query rewriting using views problems for XPath queries and views in  $XP(/, [], *)$  under a duplicate-free DTD. The algorithms are shown to be complete when the query meets a specific condition, namely its  $C$ -satisfying set (section 4.1) is not empty.

The paper is organized as follows: Section 2 reviews the necessary material concerning XPath, tree pattern queries and techniques for XPath containment. The problem of XPath query rewriting is formally defined in Section 3, where known results (in the absence of schema information) are also discussed. Section 4 develops the containment tests for  $XP(/, [], *)$  as well as the two rewriting algorithms mentioned above. Section 5 presents the related work, and we conclude with Section 6 where we discuss ongoing and future work.

## 2 XPath, DTD and XPath Containment

We first give some definitions for the XPath language and DTDs. Then we discuss the XPath query containment problem. The definitions follow closely [15] and [26].

### 2.1 XPath and Tree Pattern Queries

XML documents are modeled as rooted, ordered, unranked node-labeled trees, called XML trees, over an infinite alphabet  $S$ . We denote all XML trees over  $S$  as  $T_S$ . Given an XML tree  $t$ , following [26], for any node  $n$  in  $t$  we denote the subtree of  $t$  rooted at  $n$  as  $(t)_{sub}^n$ .

Every XPath query in  $XP(/, //, [], *)$  can be represented as a labeled tree, called a **tree pattern**, which we define in Definition [1](#).

**Definition 1.** A tree pattern  $p$  is a tree  $\langle V_p, E_p, r_p, o_p \rangle$  over  $S \cup \{*\}$ , where  $V_p$  is the node set and  $E_p$  is the edge set, and: (i) Each node  $n \in V_p$  has a label from  $S \cup \{*\}$ , denoted as  $n.label$ , (ii) Each edge  $e \in E_p$  has a label from  $\{', //, []\}$ , denoted as  $e.label$ , and (iii)  $r_p, o_p \in V_p$  are the root and the selection node of  $p$  respectively.

An edge with label  $/$  is called a child edge, otherwise a descendant edge. Given a pattern  $p \langle V_p, E_p, r_p, o_p \rangle$  we say that  $p' \langle V_{p'}, E_{p'}, r_{p'}, o_{p'} \rangle$  is a **subpattern** of  $p$  if the following conditions hold: (1)  $V_{p'} \subseteq V_p$ , (2)  $E_{p'} = (V_{p'} \times V_{p'}) \cap E_p$ , and (3) if  $o_p \in V_{p'}$ , then  $o_p$  is also the selection node of  $p'$ . We denote with  $(p)_{sub}^n$  the subpattern of  $p$  rooted at node  $n \in V_p$  and containing all descendants of  $n$ . We next provide the definition of the notion of embedding from a tree pattern to an XML tree.

**Definition 2.** [\[13\]](#) Given an XML tree  $t \langle V_t, E_t, r_t \rangle$  and a pattern  $p \langle V_p, E_p, r_p, o_p \rangle$ , an **embedding** from  $p$  to  $t$  is a function  $e : V_p \rightarrow V_t$ , with the following properties: (i) root preserving:  $e(r_p) = r_t$ , (ii) label preserving:  $\forall n \in V_p$ , if  $n.label \neq *'$ , then  $n.label = e(n).label$ , and (iii) structure preserving:  $\forall e' = (n_1, n_2) \in E_p$ , if  $e'.label = /$  then  $e(n_2)$  is a child of  $e(n_1)$  in  $t$ ; otherwise  $e(n_2)$  is a descendant of  $e(n_1)$  in  $t$ .

Each embedding maps the selection node  $o_p$  of  $p$  to a node  $n$  in  $t$ . We say that the subtree  $(t)_{sub}^n$  of  $t$  is the result of the embedding. We can now define the result of evaluating the tree pattern query  $p$  over an XML tree  $t$ .

**Definition 3.** Given a tree pattern query  $p$  and an XML tree  $t$ , then the result of evaluating  $p$  over  $t$ , denoted as  $p(t)$ , is defined as:  $p(t) = \cup_{e \in EB} \{(t)_{sub}^{e(o_p)}\}$ , where  $EB$  is the set of all embeddings from  $p$  to  $t$ .

## 2.2 Schema and DTDs

DTDs provide a means for typing, and therefore constraining, the structure of XML documents. We provide a formal definition of a DTD.

**Definition 4.** A DTD  $D$  over a finite alphabet  $S$  consists of a start symbol, denoted as  $root(D)$ , and a mapping for each symbol  $a \in S$  to a regular expression over  $S$ . Every such mapping is called a rule of the DTD and the regular expression corresponding to  $a$  is denoted as  $R^a$ .

A tree  $t \in T_S$  satisfies a DTD  $D$  over  $S$  if  $root(t).label = root(D).label$  and, for each node  $x$  in  $t$  with a sequence of children  $y_1, \dots, y_n$ , the string  $y_1.label, \dots, y_n.label$  is in  $L(R^{x.label})$ , i.e., in the regular language described by the regular expression on the right-hand side of a rule with left-hand side  $x.label$ . The set of trees satisfying DTD  $D$  is denoted as  $SAT(D)$ .

In the following, we will consider two categories of DTDs, duplicate-free DTDs and acyclic and choice-free DTDs. An acyclic and choice-free DTD does not contain alternation in any of its rules and also does not support recursion. A DTD is duplicate-free if in each right-hand side of any DTD rule each element name appears at most once. According to [17] most real-world DTDs are duplicate-free DTDs.

### 2.3 XPath Query Containment

XPath query containment is central to most query rewriting algorithms. We briefly discuss techniques applicable to this problem for subsets of the expressive XPath fragment  $XP(/, //, [], *)$ . We also present an overview of relevant results for XPath containment under DTD constraints, focusing especially on the use of the chase [14, 25]. Table 1 in appendix shows known complexity results for the query containment problem with or without DTD constraints for several XPath fragments.

**Query Containment in the absence of schema.** For an XML tree  $t$  and an XPath query  $q$  we have that  $t \models q$  if  $q(t) \neq \emptyset$ . We define the boolean XPath query containment problem.

**Definition 5.** *XPath query  $p$  is contained in  $q$ , denoted as  $p \subseteq_0 q$ , if and only if  $t \models p$  implies  $t \models q$ , for every XML tree  $t$ .*

According to [13] a more general setting of the containment problem, which is XPath query  $p$  is contained in  $q$ , denoted as  $p \subseteq q$ , if and only if  $p(t) \subseteq q(t)$  for every XML tree  $t$ , can be polynomially on the number of query nodes reduced to the boolean containment problem. Also, XPath query  $p$  is equivalent to  $q$ , denoted as  $p \equiv q$ , if and only if  $p \subseteq q$  and  $q \subseteq p$ .

We can reason about query containment using the canonical models technique [15, 20]. Although this technique provides a sound and complete algorithm for the containment problem in  $XP(/, //, [], *)$ , it is exponential to the number of descendant edges of the query. On the contrary, the homomorphism technique [15, 20] provides a sound and complete polynomial algorithm only for the containment problem for the subfragments of  $XP(/, //, [], *)$ , i.e. for  $XP(/, //, [])$ ,  $XP(/, //, *)$  and  $XP(/, [], *)$ .

We briefly describe the homomorphism technique as we will adapt and make use of it in what follows.

**Definition 6.** *A homomorphism from tree pattern  $q$  to tree pattern  $p$  is a function  $h : nodes(q) \rightarrow nodes(p)$  such that: (1) The root of  $q$  must be mapped to the root of  $p$ , (2) If  $(u, v)$  is a child-edge of  $q$  then  $(h(u), h(v))$  is a child-edge in  $p$ , (3) If  $(u, v)$  is a descendant-edge of  $q$  then  $h(v)$  has to be below  $h(u)$  in  $p$ , and (4) if  $u$  is labeled with  $e \neq *$  then  $h(u)$  also has to carry label  $e$ .*

The existence of a homomorphism is a necessary and sufficient condition for containment decision in the subfragments of  $XP(/, //, [], *)$ , though is only sufficient for fragment  $XP(/, //, [], *)$  [15].

**Query Containment with DTDs.** In the presence of a DTD, the definition of containment between two XPath queries is as follows.

**Definition 7.** *Given two XPath queries  $p, q$ , then  $q$   $D$ -contains  $p$ , written  $p \subseteq_{SAT(D)} q$ , if for every tree  $t \in SAT(D)$  it holds  $p(t) \subseteq q(t)$ .*

The containment problem for  $XP(/, [])$  in the presence of a duplicate-free DTD is in PTIME [25] and for  $XP(/, //, [])$  under an acyclic and choice-free DTD is also in PTIME [14]. In order to decide efficiently the containment problem the chase technique [14, 25] has been employed. The basic idea is to chase one of the two queries with a set of constraints *extracted from the DTD* and then decide containment between the queries with the homomorphism technique. Two of these constraints, which we use in section 4.1 are the Sibling (SC) and Functional (FC) Constraints [25]. An XML tree  $t$  satisfies the SC  $a: B \Downarrow c$  if whenever a node labeled  $a$  in  $t$  has children labeled from set  $B$  then it has a child node labeled  $c$ . Also,  $t$  satisfies the FC  $a \Downarrow b$  if no node labeled  $a$  in  $t$  has two distinct children labeled with  $b$ .

A *chasing sequence* of a query  $p$  by a set of Constraints  $C$  extracted from a DTD is a sequence  $p = p_0, \dots, p_k$  such that for each  $0 \leq i \leq k - 1$ ,  $p_{i+1}$  is the result of applying some constraint in  $C$  to  $p_i$ , and no constraint can be applied to  $p_k$ . We define the  $p_k$  expression as the *chase of  $p$  by  $C$* , denoted as  $chase_C(p)$ .

### 3 XPath Query Rewriting

We first provide the definition of the concatenation operator between two tree patterns, as in [26].

**Definition 8.** *Given two tree pattern queries  $p, q$  then the concatenation operator, denoted as  $\oplus$ , when applied on them as  $p \oplus q$  results in a tree pattern constructed by merging the root of  $p$ , denoted as  $r_p$ , and the selection node of  $q$ , denoted as  $o_q$ , into one node, provided that their labels are compatible that is (1) they carry the same label or (2) at least one of them is wildcard labeled. The merged node is labeled with their common label (case 1) or the non-wildcard label (case 2) and  $r_q, o_p$  are the root and the selection node of  $p \oplus q$  respectively.*

Given a tree pattern query  $q$  and a DTD  $D$ , then  $q$  is  $D$ -satisfiable if there is an XML tree  $t$  that satisfies  $D$  such that  $q(t) \neq \emptyset$ . We now give the definition of equivalent XPath query rewriting in the presence of a DTD.

**Definition 9.** *Given a DTD  $D$ , an XPath query  $q$  and a view  $v$ , which are  $D$ -satisfiable, the query rewriting problem is formulated as an: (1) *Existence Problem*: Decide if  $q$  is rewritable using view  $v$ , which is decide if a compensation query  $e$  exists s.t.  $e \oplus v \equiv_{SAT(D)} q$ , or a (2) *Construction Problem*: Find this compensation query  $e$  and thus the equivalent rewriting is the XPath expression  $e \oplus v$  [4].*

<sup>1</sup> In the absence of intersection, we cannot use more than one view in a rewriting in this setting.

### 3.1 Rewriting in the Absence of Schema

To the best of our knowledge there is no existing work addressing the problem in Definition 9. However, the equivalent rewriting problem in the absence of schema is addressed by Xu and Özsoyoglu [26]. The following theorem from [26] provides a basis for deciding the equivalent rewriting existence problem.

**Theorem 1.** *Let  $u, p$  be two tree patterns in the subfragments of  $XP(/, //, [], *)$ , and let the selection node of  $u$  be the  $i$ -th node in its selection path (i.e. the path from root of  $u$  to its selection node) while  $n_p$  be the  $i$ -th node in the selection path of  $p$ , then: if a compensation pattern  $p'$  of  $p$  using  $u$  exist, the subpattern  $(p)_{sub}^{n_p}$  of  $p$  is a compensation pattern of  $p$  using  $u$ .*

Theorem 1 reduces the equivalent rewriting existence problem to the decision problem of whether  $(p)_{sub}^{n_p} \oplus u \equiv p$ . Since for the subfragments of  $XP(/, //, [], *)$  the equivalence problem is in PTIME, the rewriting existence problem in these fragments is in PTIME.

## 4 Query Rewriting and Containment in the Presence of DTD

In this section we address the query containment and the equivalent query rewriting problem in the presence of a DTD for the XPath fragment  $XP(/, [], *)$ .

### 4.1 The XPath Fragment $XP(/, [], *)$ with Duplicate-Free DTD

We discuss the containment and the equivalent rewriting problem for XPath expressions in  $XP(/, [], *)$  under the assumption that the considered DTDs are duplicate-free. In order to provide an efficient solution we will attempt to reduce the DTD into an appropriate set of constraints, thus reducing the necessary containment test under a DTD to a containment test under said constraints. Then we will develop the rewriting algorithm by again appropriately utilizing Theorem 1. Following this approach we will develop a sound algorithm. Getting a sound and complete algorithm will require a non-trivial extension, as shown in the remaining of section 4.1.

**Query Containment for  $XP(/, [], *)$ .** Containment of queries in  $XP(/, [], *)$  in the presence of a duplicate free DTD is CoNP-hard [17]. In this section we provide a PTIME algorithm for the containment problem that is always sound, and is complete when the query meets a particular condition (described in Definition 10 and Theorem 2). Then we proceed to give an unconditional sound and complete EXPTIME algorithm.

Towards the efficient solution we describe first the constraints that need to be inferred by a duplicate-free DTD  $D$ . Since the XPath fragment we consider includes only the child axis we will see that we need to infer only Sibling (SC), Functional (FC) [25] and the newly defined *Unique child (UC)* constraints. A

Unique child constraint, denoted as UC:  $a \rightarrow b$ , declares that if a node labeled  $a$  in an XML tree  $t$  has children, then all of them are labeled with  $b$ . Extracting a UC from a duplicate-free DTD is in PTIME since every rule  $a \rightarrow b$  or  $a \rightarrow b^*$  from a DTD  $D$  yields the  $a \rightarrow b$  UC.

To handle the wildcard labeled nodes we define the corresponding wildcard versions of the previously mentioned constraints. Let  $S$  be the finite alphabet over which the DTD  $D$  is defined. From the set of SCs  $S' \subseteq C$ , we extract the **wildcard applicable SCs (WSC)** as follows. If in  $S'$  there are SCs of the form  $x: B \Downarrow c$  for every  $x \in S$  such that  $x$  has children with labels in  $B$ , a WSC with the form  $*: B \Downarrow c$  is derived. Also, from the set of FCs  $F \subseteq C$ , we extract the **wildcard applicable FCs (WFC)** as follows. If there are FCs in  $F$  of the form  $x \downarrow b$  for every  $x \in S$ , a WFC of the form  $* \downarrow b$  is inferred. Finally, from the set of UCs  $UI \subseteq C$ , we extract the **wildcard applicable UCs (WUC)**. If there are UCs in  $UI$  of the form  $x \rightarrow b$  for every  $x \in S$ , a WUC the form  $* \rightarrow b$  is inferred.

We observe that the alphabet  $S$  from which the labels of any tree  $t$  satisfying  $D$  come from is finite. Therefore, we note that the “options” for the wildcards in XPath expressions in  $XP(DTD, /, [], *)$  are finite (we will use this observation in the algorithms of Section 4.1). Furthermore, all the above constraints can be extracted in PTIME, just following the definitions and utilizing the PTIME extraction of SCs and FCs from duplicate-free DTDs [25]. In the following, we use terms FC, SC and UC meaning also the WFC, WSC and WUC constraints.

Let  $p$  be a query in  $XP(/, [], *)$  and  $C$  be the set of SCs, FCs and UCs implied by a DTD  $D$ . We describe how to chase  $p$  applying these constraints inspired by [25]:

*Sibling Constraints:* Let  $s \in C$  be a SC of the form  $a: B \Downarrow c$  or a WSC of the form  $*: B \Downarrow c$ , where  $B = \{b_1, \dots, b_n\}$ . Let  $u$  be a node of  $p$  with children  $v_1, \dots, v_n$ , such that  $u.label = a$  for the SC case or  $u.label = *$  for the WSC case and  $v_i.label = b_i$ ,  $1 \leq i \leq n$ , and  $u$  does not have a child labeled  $c$ , then the chase result on  $p$  will be the adding of a child-edge to  $p$  between  $u$  and  $c$ .

*Functional Constraints:* Let  $f \in C$  be a FC of the form  $a \downarrow b$  or a WFC of the form  $* \downarrow b$ . Let  $u$  be a node of  $p$  with distinct children  $v$  and  $w$ , such that  $u.label = a$  for the FC case or  $u.label = *$  for the WFC case and  $v.label = w.label = b$ , then the chase result on  $p$  is the merging of  $v, w$  of  $p$ .

*Unique Child Constraints:* Let  $ui \in C$  be a UC of the form  $a \rightarrow b$  or a WUC of the form  $* \rightarrow b$ . Let  $u$  be a node of  $p$  with distinct wildcard or  $b$  labeled children  $v_1, \dots, v_n$  (and for at least one  $v_i$ ,  $v_i.label = b$  in the WUC case), such that  $u.label = a$  for the UC case or  $u.label = *$  for the WUC case, then the chase result on  $p$  is turning the labels of  $v_1, \dots, v_n$  to  $b$ .

**Lemma 1.** *Let  $D$  be a duplicate-free DTD. Then  $q \equiv_{SAT(D)} chase_C(q)$ .*

Moreover, containment of query  $q$  in  $p$  under the set of constraints  $C$  is reducible to containment of  $chase_C(q)$  in  $p$ .



**Lemma 2.** *Let  $D$  be a duplicate-free DTD and  $C$  be the set of FCs, SCs and UCs implied by  $D$ . For  $XP(/, [], *)$  queries  $p$  and  $q$  it holds  $p \supseteq_{SAT(C)} q$  if and only if  $p \supseteq \text{chase}_C(q)$ .*

*Proof.* See proof in [2]. □

Given the above, we would like to show that  $p \supseteq_{SAT(D)} q$  if and only if  $p \supseteq_{SAT(C)} q$ . As it turns out, this is not always the case for queries in  $XP(/, [], *)$ . The following definition adopted from [25] helps in proving the required condition under which it holds that  $p \supseteq_{SAT(D)} q$  if and only if  $p \supseteq_{SAT(C)} q$ .

**Definition 10.** [25] *Let  $q$  be a  $D$ -satisfiable query in  $XP(/, [], *)$  in the presence of DTD  $D$  and  $R \subseteq SAT(D)$  be the set of trees with a subtree 1-1 homomorphic to  $\text{chase}_C(q)$ . We call  $R$  the  **$C$ -satisfying** set for  $q$ . Each tree in  $R$  has a core subtree which is 1-1 homomorphic to  $\text{chase}_C(q)$  and each node in that core subtree is called **core node**. Each node which is not a core node is called a **non-core node**.*

Specifically, we have proven theorem [2] using lemma [3].

**Lemma 3.** *Let  $D$  be a duplicate-free DTD and  $p, q$  be queries in  $XP(/, [], *)$ . Let  $q$  be  $D$ -satisfiable,  $C$  be the set of SCs, FCs and UCs implied by  $D$ , and  $R$  the  $C$ -satisfying set for  $q$ , such that  $R \neq \emptyset$ . If  $p \supseteq_{SAT(D)} q$ , then for each node  $w$  in  $p$ , either  $w$  can be mapped to a core node in every tree in  $R$  or  $w$  can be mapped to a non-core node in every tree in  $R$ .*

**Theorem 2.** *Let  $D$  be a duplicate-free DTD and  $C$  be the set of FCs, SCs, UCs implied by  $D$ . Let  $p, q$  be queries in  $XP(/, [], *)$ , such that the  $C$ -satisfying set for  $q$   $R \neq \emptyset$ . Then  $p \supseteq_{SAT(D)} q$  if and only if  $p \supseteq_{SAT(C)} q$ .*

*Proof.* (if) Assume  $p \supseteq_{SAT(C)} q$  then  $p \supseteq_{SAT(D)} q$ , because  $SAT(C) \supseteq SAT(D)$ .

(only if) Assume  $p \supseteq_{SAT(D)} q$  but  $p \not\supseteq_{SAT(C)} q$ . We will derive a contradiction. Since there is an embedding from  $q$  to every  $t \in R$  and  $p \supseteq_{SAT(D)} q$ , then there is also an embedding from  $p$  to every tree  $t \in R$ .

If  $p \not\supseteq_{SAT(C)} q$  then by lemma [2] there is no homomorphism from  $p$  to  $\text{chase}_C(q)$ . We have the following two cases:

Case A: a single path in  $p$  fails to map to any path in  $\text{chase}_C(q)$  : There is a node  $x$  in  $p$  with parent  $y$ , such that  $y$  is mapped to a node in  $\text{chase}_C(q)$  but no mapping from  $x$  to any node in  $\text{chase}_C(q)$  exists. We originally consider that node  $y$  does not map to a leaf in  $\text{chase}_C(q)$ , which means that  $x.\text{label}$  can not be wildcard. So in any  $t \in R$ ,  $x$  can never be mapped to a core node while  $y$  can always be mapped to a core node  $u$ . Lemma [3] and  $p \supseteq_{SAT(D)} q$  imply that  $x$  can always be mapped to a non-core node  $v$  in  $t \in R$ . Since  $v$  is a non-core node and  $x$  can not be mapped to a core node,  $D$  can not imply the SC  $u: B \Downarrow v$ , where  $u$  and  $v$  are the nodes that  $y$  and  $x$  mapped in a  $t \in R$  and  $B$  is the set of core node children of  $u$ . The same holds, if node  $y$  is mapped to a wildcard labeled node in  $\text{chase}_C(q)$ . This is because  $D$  can not imply the SC  $*: B \Downarrow v$ , which means there is a node labeled  $u$  that  $D$  can not imply the SC  $u: B \Downarrow v$ .

So, there must be a tree  $U \in SAT(D)$  which has a node  $w$  labeled the same as  $u$  and children labeled from  $B$ , but with no child labeled as  $v$  node. Then, we can replace the non-core child subtrees of  $u$  by the non-core child subtrees of  $w$  and still have a tree  $t' \in R$ . Node  $x$  can not be mapped to any non-core node in  $t'$ . - A contradiction.

If we consider that the path in  $p$ , which fails to map to any path in  $chase_C(q)$ , is shorter from every path in  $chase_C(q)$  (i.e.  $y$  is mapped to a leaf node in  $chase_C(q)$ ) then we have two cases: (i) For the non wildcard-labeled leaves in  $chase_C(q)$ ,  $D$  can not imply the SC  $u:\emptyset \Downarrow v$ , which means that either  $p$  is unsatisfiable or the arguments in the proof still hold, (ii) For the wildcard-labeled leaves in  $chase_C(q)$ ,  $D$  can not imply the SC  $*:\emptyset \Downarrow v$ , which means there is at least one node labeled say  $u$  that can not have as child a node labeled  $v$ . If all nodes can not have as child a node  $v$ , then  $p$  is unsatisfiable. If at least one node can have as child node  $v$ , then the proof's arguments still hold.

Case B: each path in  $p$  can map but for some node  $w$  which is common ancestor of nodes  $u$  and  $v$ , the two mapped paths for  $u$  and  $v$  in  $chase_C(q)$  can not be joint on  $w$  : Let  $w_n$  be the common ancestor of  $u, v$  that is closest to the root of  $p$ , such that the path from root to  $u$  via  $w_n$  (i.e.  $root \rightarrow w_n \rightarrow u$ ) maps to a path in  $chase_C(q)$  using homomorphism  $h_1$  and the path from root to  $v$  via  $w_n$  (i.e.  $root \rightarrow w_n \rightarrow v$ ) maps to a path in  $chase_C(q)$  using homomorphism  $h_2$ , but  $h_1(w_n) \neq h_2(w_n)$ . Since  $chase_C(q)$  has a single root and  $h_1(w_n) \neq h_2(w_n)$ , then none of these nodes labeled the same as  $w_n$  is  $chase_C(q)$  root. Therefore,  $w_o$ , the root of  $chase_C(q)$ , is a parent of two children labeled the same as  $w_n$  and for the root of  $p$ , say  $wp_o$ , it holds  $h_1(wp_o) = h_2(wp_o) = w_o$ .

Case B1  $w_n.label \neq *$  : Note that  $D$  can not imply the FC  $w_o \Downarrow w_n$  (since the  $w_n$ -labeled nodes in  $chase_C(q)$  would have been merged) and that means that  $w_o$  can have an unbounded number of children labeled the same as  $w_n$ . Let  $t$  be the smallest tree in  $R$ , such that the node in  $t$  that corresponds to  $w_o$  in  $chase_C(q)$  has two children nodes with labels the label of  $w_n$ . Then there is no embedding from  $p$  to  $t$ . - A contradiction.

Case B2  $w_n.label = *$  : Note that  $D$  can not imply the UC  $w_o \rightarrow b$  and the FC  $w_o \Downarrow w_n$ , for some label  $b \in S$ , where  $S$  is the alphabet from  $D$  (since the  $w_n$ -labeled nodes in  $chase_C(q)$  would have been merged) and that means that  $w_o$  can have at least two children labeled from  $S$ . Let  $t$  be the smallest tree in  $R$ , such that the node in  $t$  that corresponds to  $w_o$  in  $chase_C(q)$  has two children nodes with labels from  $S$ . Then there is no embedding from  $p$  to  $t$ . - A contradiction.  $\square$

Note that in theorem [2](#) the direction “if  $p \supseteq_{SAT(D)} q$  then  $p \supseteq_{SAT(C)} q$ ” is trivial and always holds. The opposite direction requires  $R \neq \emptyset$ . Example [1](#) describes a case in which  $R = \emptyset$  and the effects that this has in the decision of  $p \supseteq_{SAT(D)} q$ .

*Example 1.* Let  $D$  be a duplicate-free DTD:  $a \rightarrow (b, c)$ ,  $b \rightarrow (i, j)$ ,  $c \rightarrow (k)$ , and let  $p \equiv /a[b[i][j]]$  and  $q \equiv /a[*[i]][*j]$  be two XPath expression in  $XP(/, [], *)$ . It is obvious that  $chase_C(q) \equiv q$  and also that  $p \not\supseteq chase_C(q)$  since there is no homomorphism from  $p$  to  $chase_C(q)$ . It is the case in the example that  $chase_C(q)$

is not 1-1 homomorphic to any tree in  $SAT(D)$ , or in other words  $R = \emptyset$ . Hence, theorem 2 does not hold. Indeed, while  $p \supseteq_{SAT(D)} q$ , according to the theorem it should not.

Deciding the emptiness of  $R$  for a query  $q$ , which is a critical condition in theorem 2 is possible as proved in theorem 3.

**Theorem 3.** *Given an XPath query  $q$  in  $XP(/, [], *)$  and a duplicate-free DTD  $D$  the problem of determining whether the  $C$ -satisfying set for  $q$  is empty, i.e.  $R = \emptyset$ , is decidable.*

*Proof.* See proof in 2. □

**A Sound and Complete Algorithm for Query Containment.** Given  $q, p$  XPath queries in  $XP(/, [], *)$  and a duplicate-free DTD  $D$  we provide a sound and complete algorithm for deciding  $p \supseteq_0 q$ . The basic idea is to eliminate wildcards from  $q$  and then by utilizing theorem 2 and lemma 2 we are able to decide the containment problem.

First we describe how to reduce a query in  $XP(/, [], *)$  to a disjunction of queries in  $XP(/, [])$ . We will do so by providing a useful definition and then describing a full procedure for eliminating wildcards from  $q$ .

**Definition 11.** *Let  $q$  be an XPath query in  $XP(/, [], *)$  and  $D$  be a duplicate-free DTD over  $S$ . Let also  $n$  be the number of nodes  $x \in V_q$ , such that  $x.label = *$ . We define as  $L(x_1, \dots, x_n)$  the  $n$ -ary tuple for which the following hold: (1) each  $x_i \in S$ , with  $1 \leq i \leq n$ , (2) each  $x_i$  corresponds to a wildcard-labeled node from  $q$  and (3) the query  $q'$ , that comes from  $q$  by replacing every wildcard node with its corresponding  $x_i$ , is  $D$ -satisfiable.*

*The wildcard-set of  $q$ , denoted  $L(q)$ , is the maximum set that contains tuples of the form  $L(x_1, \dots, x_n)$ .*

*Example 2.* Let  $q \equiv /a/*g$  be an XPath query in  $XP(/, [], *)$  and the following duplicate free DTD  $D$ :  $a \rightarrow (b, c, d, e^*)$ ,  $b \rightarrow (f, g, a^*)$ ,  $e \rightarrow (f, g)$ ,  $c \rightarrow (i)$ ,  $d \rightarrow (i)$ ,  $f \rightarrow (i)$ ,  $g \rightarrow (i)$ .

The alphabet is  $S = \{a, b, c, d, e, f, g, i\}$  and the query  $q$  has only one wildcard label. It is obvious to see that  $L(q) = \{(b), (e)\}$ , since  $/a/b/g$  and  $/a/e/g$  are  $D$ -satisfiable queries.

### Algorithm 1. Wildcard Elimination

**Input:** An XPath query  $q$  in  $XP(/, [], *)$  and a DTD  $D$ .

Compute wildcard-set  $L(q)$ .

$i \leftarrow 0$

FOR EACH  $n$ -ary relation  $L(x_1, \dots, x_n) \in L(q)$  DO

$i \leftarrow i + 1$

$q_i \leftarrow$  replace every wildcard in  $q$  with each corresponding  $x_i$

$q' \leftarrow q' \mid q_i$

END FOR

**Output:** XPath query  $q' \equiv q_1 \mid \dots \mid q_m$ , s.t.  $q_i$  is in  $XP(/, [])$

The reduction of an XPath query from  $XP(/, [], *)$  to  $XP(/, [])$  under a duplicate-free DTD produces an XPath query that is formulated as the disjunction of a set of queries in  $XP(/, [])$ . Wildcard elimination algorithm is not polynomial due to the complexity of computing the wildcard-set. If we consider all possible values for each wildcard labeled node in computing the wildcard set this yields an exponential algorithm. Note that the  $D$ -satisfiability does not influence this complexity, since its complexity for XPath queries in  $XP(/, [])$  under a duplicate-free DTD is PTIME [17]. It is obvious that the number of queries in the disjunction is equal to the cardinality of the wildcard-set, which in worst case is exponential to the number of the symbols in the alphabet of the DTD.

Algorithm 2 below is a sound and complete decision procedure for the query containment problem for XPath expressions in  $XP(/, [], *)$ . It reduces  $q$  into a set of XPath queries in  $XP(/, [])$  using algorithm 1. Then, using the fact that an XPath expression contains a disjunction of XPath expressions if it contains every single expression that constitutes the disjunction, theorem 2 and lemma 2 we can decide  $p \supseteq_0 q$ , since for each query in  $XP(/, [])$   $R \neq \emptyset$  [25]. The complexity is dominated by the wildcard-Set computation. Also note that  $chase_C(q_i)$  is computed considering only FCs and SCs because each  $q_i$  is in  $XP(/, [])$ .

### Algorithm 2. Containment Check

**Input:**  $q, p$  in  $XP(/, [], *)$ , a set  $C$  of SCs, FCs inferred by a duplicate-free DTD.  
 $q' \leftarrow \text{Wildcard} - \text{Elimination}(q)$   
**FOR EACH**  $q_i \in q'$  **DO**  
     $q_i \leftarrow chase_C(q_i)$   
    **IF** there is no homomorphism from  $p$  to  $q_i$   
        **THEN RETURN** “no”  
**END FOR**  
**RETURN** “yes”;  
**Output:** “yes” if  $p \supseteq_0 q$ ; “no” otherwise.

*Example 3.* (continuing Example 1) Algorithm 2 first will call the Wildcard-elimination procedure on  $q$  that results in an expression  $q' \equiv /a[b[i]][b[j]]$ . Then the **FOR** loop will be executed only once and in this execution first the chase of  $q'$  will be computed that is  $chase_C(q') \equiv /a[b[i]][j]$  applying the FC  $a \downarrow b$  and then will be checked the homomorphism existence from  $p$  to  $chase_C(q')$ . Since there exists such a homomorphism then the algorithm will output “yes” meaning that  $p \supseteq_{SAT(D)} q$ .

**A PTIME Algorithm for Query Containment.** Utilizing the equivalence  $p \supseteq_{SAT(D)} q$  if and only if  $p \supseteq chase_C(q)$ , we can derive from theorem 2 and lemma 2 a PTIME procedure for deciding containment of queries in  $XP(/, [], *)$  under a duplicate-free DTD. Since theorem 2 requires the  $C$ -satisfying set  $R$  for query  $q$  to be non-empty, the algorithm is complete only for queries with this property. It is trivially sound since, if  $p \supseteq chase_C(q)$  then  $p \supseteq_{SAT(D)} q$ . It uses a

PTIME chase procedure inspired by [25], that in contrast to applying every rule to the query until no other rule application changes the query, applies the rules on  $q$  along its selection path in a way that takes into account the query  $p$ .

**Algorithm 3. PTIME Containment Check**

**Input:**  $q, p$  in  $XP(/, [], *)$ , a duplicate free DTD  $D$ .

Step 1: Compute the chase $_C(p)$  –  $C$  set of FCs, SCs, UCs inferred by  $D$ .

$p \leftarrow \text{chase}_C(p, q, D)$

Step 2: Check Containment

IF there exist a homomorphism from  $q$  to  $p$  THEN RETURN “yes”

ELSE RETURN “no”

**Output:** “yes” if  $p \subseteq_0 q$ ; “no” otherwise.

**Algorithm 4. Chase( $p, q, D$ )**

**Input:**  $p, q \in XP(/, [], *)$ ,  $D$  a duplicate-free DTD.

Apply UCs to  $p$  inferred by  $D$

Apply FCs to  $p$  inferred by  $D$

Traverse in top-down manner expressions  $p, q$

Find nodes  $x \in q$  and  $u \in p$  s.t.

( $x.\text{label} = u.\text{label}$  OR  $u.\text{label} = *$  OR  $x.\text{label} = *$ ) AND ( $x$  has a child  $y$ )

AND ( $u$  has no child with  $y.\text{label} \neq *$  OR  $y.\text{label} = *$ )

IF ( $y.\text{label} \neq *$ )

Apply the inferred SC  $u.\text{label}: B \Downarrow y.\text{label}$  from  $D$  to  $p$ ,

where  $B$  is the set of labels of children of  $u \in p$ .

ELSE – ( $y.\text{label} = *$ )

Apply every inferred SC  $u.\text{label}: B \Downarrow z.\text{label}$  from  $D$  to  $p$ ,

where  $B$  is the set of labels of children of  $u \in p$ .

**Output:** chase $_C(p)$ .

**Query Rewriting for  $XP(/, [], *)$ .** For a view  $v$  and a query  $q$  in  $XP(/, [], *)$ , it is CoNP-hard to decide if there is a rewriting of  $q$  using  $v$  in the presence of a duplicate free DTD  $D$ . This follows immediately from theorem 2 of [17] and lemma 4.1 of [26]. In what follows we provide two algorithms for the query rewriting problem described in definition 9.

**A Sound and Complete Query Rewriting Algorithm.** Algorithm 5 below, when we use as containment test Algorithm 2, is a sound and complete algorithm for the rewriting problem for queries in  $XP(/, [], *)$  under a duplicate-free DTD.

**Algorithm 5. Query Rewriting using views**

**Input:**  $q, v$  in  $XP(/, [], *)$ ,  $D$  a duplicate-free DTD  $D$ .

Step 1: Check rewriting existence

let  $o_v$  be the output node of  $v$

let  $n_q$  be a node in  $q$  having the same position to  $o_v$

IF ( $(q)_{sub}^{n_q} \oplus v \not\equiv_{SAT(D)} q$ ) THEN RETURN “no”

*ELSE RETURN “yes”*

*Step 2: Find rewriting*

*IF (answer is “yes”)*

*THEN  $(q)_{sub}^{n_q}$  is a compensation pattern of  $q$  using  $v$ .*

**Output:** “yes” if  $q$  is rewritable using  $v$  and the compensation pattern  $(q)_{sub}^{n_q}$ .

Since the containment test is EXPTIME then algorithm 5, whose correctness follows from theorem 1, is in EXPTIME.

**Query Rewriting in PTIME.** If in algorithm 5 we use instead the PTIME containment test of section 4.1 then we obtain a PTIME query rewriting algorithm.

Example 4 shows a simple rewriting for a query in  $XP(/, [], *)$ .

*Example 4.* (continuing Example 2) Let  $v \equiv /a/*[f]$  be an XPath view in  $XP(/, [], *)$  under the duplicate free DTD  $D$  defined in example 2. Algorithm 5 will generate as compensation query  $c \equiv /*/g$ . Notice that chasing  $q$  with the WSC  $*:\{g\} \Downarrow f$  we get  $q' \equiv /a/*[f]/g$ .

## 4.2 The XPath Fragment $XP(/, [], *)$ with Acyclic and Choice-Free DTD

We can adapt algorithm 5 to solve the query rewriting problem under acyclic and choice-free DTD. For the wildcard-labeled children of any node of a query in  $XP(/, [], *)$  under an acyclic and choice-free DTD, is easy to determine to which label from the alphabet correspond. Therefore following the wildcard elimination technique we obtain a PTIME algorithm for the containment problem for queries in  $XP(/, [], *)$  under acyclic and choice-free DTD. This problem is shown in 4 to be in PTIME. Therefore, we can obtain a PTIME sound and complete algorithm for the rewriting problem.

## 5 Related Work

A framework for the problem of finding equivalent XPath query rewritings using views is provided in 3. The problem is studied extensively in 11,26 for XPath queries in  $XP(/, //, [], *)$  providing either incomplete algorithms 26 or sufficient conditions under which the problems can be shown to be CoNP-Complete 1. There is also recent work 9,21 on the problem of answering XPath queries using multiple views via the use of node intersection. Such work expands the scope of the XPath subset considered for the views and the rewritings, adding intersection. Finally, in 10 Cautis et al. have looked into the problem of rewriting XPath queries using views generated by expressive declarative specifications resulting potentially in an infinite number of views. The specifications impose certain constraints on the structure of generated views, but these constraints are orthogonal to DTD constraints, which are imposed on the database.

The XPath query containment problem, which is very closely related to the XPath query rewriting problem, has been extensively studied for various XPath fragments in [5,11,14,15,18,24,25,27], including certain constraints such as DTDs. Relevant work has been discussed in section 2. We use and extend these results developing algorithms for the problem under DTD constraints for an XPath fragment with wildcard. These results are the basis for the rewriting algorithms we propose.

## 6 Conclusions and Future Work

We develop sound and complete algorithms for XPath rewriting using views in the presence of DTD constraints for the XPath fragment  $XP(/, [], *)$ . We also provide a novel reduction of a DTD into chase-able constraints and use it to develop two algorithms for checking containment for the  $XP(/, [], *)$  fragment; one is polynomial but is complete only when the query satisfies a certain condition (not empty C-Satisfying set), the other is complete but exponential, as it relies on the elimination of wildcards from one of the queries.

We plan to continue our investigation into efficient techniques for rewriting using views for more expressive XPath fragments including backward axes. Moreover, following recent results for the containment of XPath fragments that include union and intersection [22], we are working on rewriting for queries and views for those XPath fragments.

## References

1. Afrati, F., Chirkova, R., Gergatsoulis, M., Pavlaki, V., Kimelfeld, B., Sagiv, Y.: On rewriting xpath queries using views. In: EDBT, pp. 168–179 (2009)
2. Aravogliadis, P., Vassalos, V.: Rewriting xpath queries using views under dtd constraints. Technical report, AUEB (2009), <http://wim.aueb.gr/papers/xpathrewrite-ext.pdf>
3. Balmin, A., Ozcan, F., Beyer, K., Cochrane, R., Pirahesh, H.: A framework for using materialized XPath views in XML query processing. In: Proc. of the 30th VLDB Conference, pp. 60–71 (2004)
4. Benedikt, M., Fan, W., Geerts, F.: XPath satisfiability in the presence of DTDs. In: Proc. PODS 2005, pp. 25–36 (2005)
5. Benedikt, M., Koch, C.: Xpath leashed. ACM Computing Survey 41(1) (2008)
6. Berglund, A., Boag, S., Chamberlin, D., et al.: XML Path Language (XPath) 2.0. W3C, <http://www.w3.org/TR/XPath20>
7. Boag, S., Chamberlin, D., Fernandez, M.F., et al.: XQuery 1.0: An XML Query Language. W3C, <http://www.w3.org/TR/XQuery>
8. Bruno, N., Koudas, N., Srivastava, D.: Holistic twig joins: optimal XML pattern matching. In: Proc. SIGMOD Conference, pp. 310–321 (2002)
9. Cautis, B., Deutsch, A., Onose, N.: Xpath rewriting using multiple views: Achieving completeness and efficiency. In: Proc. WebDB 2008 (2008)
10. Cautis, B., Deutsch, A., Onose, N., Vassalos, V.: Efficient rewriting of xpath queries using query set specifications. In: Proceedings of the VLDB Endowment, vol. 2, pp. 301–312 (2009)

11. Deutsch, A., Tannen, V.: Containment and integrity constraints for xpath. In: KRDB Workshop (2001)
12. Halevy, A.: Answering queries using views: A survey. *VLDB J.* 10(4), 270–294 (2001)
13. Kimelfeld, B., Sagiv, Y.: Revisiting redundancy and minimization in an xpath fragment. In: *EDBT*, pp. 61–72 (2008)
14. Lakshmanan, L., Wang, H., Zhao, Z.: Answering tree pattern queries using views. In: *Proc. of the 32th VLDB Conference*, pp. 571–582 (2006)
15. Miklau, G., Suciu, D.: Containment and equivalence for an XPath fragment. *Journal of the ACM* 51(1) (2004)
16. Milo, T., Suciu, D.: Index structures for path expressions. In: Beeri, C., Bruneman, P. (eds.) *ICDT 1999*. LNCS, vol. 1540, pp. 277–295. Springer, Heidelberg (1998)
17. Montazerian, M., Wood, P., Mousavi, S.: XPath query satisfiability is in ptime for real-world dtDs. In: *XSym*, pp. 17–30 (2007)
18. Neven, F., Schwentick, T.: On the complexity of XPath containment in the presence of disjunction, DTDs, and variables. In: *Proc. Logical Methods in Computer Science*, vol. 2 (2006)
19. Papakonstantinou, Y., Vassalos, V.: The Enosys Markets Data Integration Platform: Lessons from the trenches. In: *CIKM*, pp. 538–540 (2001)
20. Schwentick, T.: XPath query containment. *SIGMOD Record* 33(1), 101–109 (2004)
21. Tang, N., Yu, J.X., Ozsu, M.T., Choi, B., Wong, K.: Multiple materialized view selection for xpath query rewriting. In: *ICDE Proc. of the 2008*, pp. 873–882 (2008)
22. ten Cate, B., Lutz, C.: The complexity of query containment in expressive fragments of XPath 2.0. In: *Proc. PODS 2007*, pp. 73–82 (2007)
23. Wang, J., Li, J., Yu, J.X.: Answering tree pattern queries using views: a revisit. In: *Proc. EDBT* (2011)
24. Wood, P.: Minimizing simple XPath expressions. In: *Proc. WebDB 2001*, pp. 13–18 (2001)
25. Wood, P.T.: Containment for XPath fragments under DTD constraints. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) *ICDT 2003*. LNCS, vol. 2572, pp. 297–311. Springer, Heidelberg (2002)
26. Xu, W., Ozsoyoglu, Z.M.: Rewriting XPath queries using materialized views. In: *VLDB 2001*, pp. 121–132 (2005)
27. Zhou, R., Liu, C., Wang, J., Li, J.: Containment between unions of xpath queries. In: Zhou, X., Yokota, H., Deng, K., Liu, Q. (eds.) *DASFAA 2009*. LNCS, vol. 5463, pp. 405–420. Springer, Heidelberg (2009)



## Appendix

**Constraints from DTD.** In section 4.1 we define the SC, FC and UC constraints and their corresponding wildcard versions. The definition of WSC was such that no new wildcard nodes will be added to the query after chasing it. Also, rules of the form  $a \downarrow *$  are not extracted from DTD since such rules are a meaningless abbreviation of a set of functional constraints and also according to the semantics of chase such rules will not maintain the equivalence given in lemma 1. Finally, note that the UC constraint declares that a node will have children all labeled the same and obviously can not derive from SC or FC constraints.

**Table 1.** XPath query containment complexity with-without DTD

DTD	/	//	□	*	Complexity
without	+	+	+		PTIME [15]
without	+	+		+	PTIME [15][16]
without	+		+	+	PTIME [15]
without	+	+	+	+	CoNP-Complete [15]
general	+	+			PTIME [18]
general	+			+	CoNP-complete [18]
duplicate-free					PTIME [25]
general					CoNP-hard [18]
acyclic & choice-free	+	+	+		PTIME [14]
duplicate-free					CoNP-hard [17]
acyclic & choice-free	+		+	+	PTIME [4]
general	+	+	+	+	EXPTIME-complete [25]

# Incremental Maintenance of Materialized XML Views

Leonidas Fegaras

University of Texas at Arlington  
fegaras@cse.uta.edu

**Abstract.** We investigate the problem of incremental maintenance of materialized XML views. We are considering the case where the underlying database is a relational database and the view exposed to querying is a materialized XML view. Then, updates to the underlying database should be reflected to the stored XML view, to keep it consistent with the source data, without recreating the entire view from the database after each source update. Unlike related work that uses algebraic methods, we use source-to-source compositional transformations, guided by the database and view schemata. We first translate SQL updates to pure (update-free) XQuery expressions that reconstruct the entire database state, reflecting the updated values in the new state. Then, we synthesize the right-inverse of the XQuery view function, guided by the view schema. This inverse function is applied to the old view to derive the old database state, which in turn is mapped to the new database state through the update function, and then is mapped to the new view through the view function. The resulting view-to-view function is normalized and translated to XQuery updates that destructively modify the materialized view efficiently to reflect the new view values.

## 1 Introduction

We address the problem of incremental maintenance of materialized XML views. That is, given a materialized XML view over a database and an update against the database, our goal is to generate view updates so that the new view after the updates is exactly the same as the view we would have gotten if we had applied the view to the updated source data. We are considering the case where the underlying database is a relational database and the view exposed to querying is a materialized XML view, stored in a native XML database. A native XML database is a specialized database that supports storage, indexing methods, and query processing techniques specially tailored to XML data. There are already many native XML database management systems available, including Oracle Berkeley-DB XML, Qizx, Natix, etc. Our framework is targeting those native XML databases that support the XQuery Update Facility (XUF) [16] for updating XML data. Exporting data through materialized XML views is very important to data integration, not only because it speeds up query processing, but also because it provides a light-weight interface to data services, without exposing the main database. The incremental maintenance of materialized views is important for applications that require freshness of view data, but it is too expensive to recompute the view data every time the source data change.

In our framework, SQL updates are translated to plain XQueries applied to a canonical view of the relational database. More specifically, we derive a pure update function

that maps the old database to the new database. Then, we synthesize the right-inverse of the XQuery view function, guided by the view schema. This inverse function is applied to the old view to derive the old database state, which in turn is mapped to the new database state through the update function, and then mapped to the new view through the view function. Although correct, the resulting view-to-view function can be very inefficient. We provide methods to optimize this program and translate it into efficient XQuery updates, so that only the relevant parts of the materialized view are updated. More specifically, our framework performs the following tasks:

1. We map the underlying relational database  $DB$  to the XML data  $db$  using a canonical view of relational data. This wrapper, which is a virtual XML view, allows us to refer to relational data within the XQuery data model without any model extension.
2. We express the source SQL updates in pure (update-free) XQuery code over the virtual XML view, as a function  $u$  from the old to the new database state that reconstructs the entire canonical XML view of the relational database, reflecting the updates.
3. Given a view,  $view$ , expressed in XQuery, we synthesize the XQuery expression  $view'$  that is a right-inverse of  $view$ , such that  $view(view'(V)) = V$  for any valid view instance  $V$  (that is,  $view$  must be a surjective function). Note that,  $view'(view(DB))$  is not necessarily equal to  $DB$ ; it is equal to  $DB$  for isomorphic views only, which are uncommon.
4. The composition  $F(V) = view(u(view'(V)))$  is a function from the old view  $V$  to the new view (after the updates), where  $view'(V)$  creates an entire source database from the view  $V$ , then  $u$  reconstructs this database to reflect the updates, and finally  $view$  maps the new database to the new view. We use normalization techniques to optimize  $F(V)$  so that the resulting program avoids the reconstruction of most parts of the intermediate database data.
5. Finally, we rewrite the normalized XQuery expression  $F(V)$  into efficient XQuery updates that destructively modify the original view  $V$  using the XQuery Update Facility (XUF). That is, the resulting program is a direct view update expressed in XUF that, in most cases, does not access the underlying database.

The key contribution of our work is in the development of a novel framework for the incremental maintenance of materialized XML views that uses source-to-source, compositional transformations only. Unlike related approaches that use framework-specific algebras to achieve a similar goal [10,12], our work can be incorporated into any existing XQuery engine. Since it breaks the task of view maintenance into a number of more manageable tasks that are easier to implement and verify their correctness, it has the potential of becoming a general methodology for incremental view maintenance.

The most important limitation of our approach is that both the inverse synthesis (Step 3) and the XUF generation (Step 5) algorithms are schema-based. That is, they require that both the underlying relational and the XML view schemata be known at query translation time. The second limitation is that our algorithms are heuristics that work only in certain cases, such as they can handle one-to-one and one-to-many but they cannot handle many-to-many joins in the view definition. Finally, due to the lack of space, we have not covered some parts the XQuery/XUF syntax, such as descendant-or-self

steps, but we believe that there is no fundamental reason why our framework cannot be extended to cover most of the XQuery syntax.

## 2 Related Work

Although the problem of incremental view maintenance has been extensively studied for relational views (see [11] for a literature survey), there is still little work on XML view maintenance [31]. One of the earliest works on incremental maintenance of materialized views over semi-structured data was for the Lorel query language [2], which was based on the graph-based data model OEM. Their method produces a set of queries that compute the updates to the view based upon an update of the source. Although there are already a number of proposals for XQuery update languages, there is now a W3C candidate recommendation, called XQuery Update Facility (XUF) [16]. El-Sayed *et al* [12] use an algebraic approach for incremental XQuery view maintenance, where an update to the XML source is transformed into a set of update primitives that are propagated through the XML algebra tree and become incremental update primitives to be applied to the result view. The work by Sawires *et al* [14] considers the view maintenance problem for simple XPath views (without value-based predicates). It reduces the frequency of view recomputation by detecting cases where a source update is irrelevant to a view and cases where a view is self-maintainable given a base update. BEA's AquaLogic Data Services Platform allows developers to specify inverse functions that enable its XQuery optimizer to push predicates and perform updates on data services that involve the use of such functions [13]. Finally, although not directly applicable to incremental view maintenance, the inverse problem of updating XML views over relational data by translating XML view updates to embedded SQL updates has been studied by earlier work (see for example [46]).

## 3 Our Framework

In this section, we use an XML view example to describe our approach to the incremental maintenance of XML views. Consider a bibliography database described by the following relational schema:

```
Article ( aid, title, year )
Author ( aid, pid )
Person ( pid, name, email )
```

where the attributes Author.aid and Author.pid are foreign keys that reference the keys Article.aid and Person.pid, respectively. An XML view over a relational database can be defined over a canonical XML view of relational data, as is done in SilkRoute [9] and many other systems. The canonical XML view of the bibliography relational schema has the following XML type:

```
element DB {
  element Article {
    element row { element aid { xs:int }, element title { xs:string },
                 element year { xs:string } }* },
```

```

element Author {
  element row { element aid { xs: int }, element pid { xs: int } }*,
element Person {
  element row { element pid { xs: int }, element name { xs: string },
    element email { xs: string } }* } }

```

that is, a relational table is mapped to an XML element whose tag name is the table name and its children are row elements that correspond to the table tuples. In addition, this type satisfies the following key constraints:

```

key { /DB/Article/row, aid/data() }
key { /DB/Author/row, (aid,pid)/data() }
key { /DB/Person/row, pid/data() }

```

As in XML-Schema [15], for a key  $\{ p_1, p_2 \}$ , the selector path  $p_1$  defines the applicable elements and the field path  $p_2$  defines the data that are unique across the applicable elements. For example, key  $\{ /DB/Article/row, aid/data() \}$  indicates that the aid elements are unique for the Article rows.

Given this canonical view \$DB of the relational database, an XML view, view(\$DB), can be defined using plain XQuery code:

```

view($DB) = <bib>{
  for $i in $DB/Article/row
  return <article aid='{ $i/aid/data() }'>{
    $i/ title , $i/year,
    for $a in $DB/Author/row[aid=$i/aid]
    return <author pid='{ $a/pid/data() }'>{
      $DB/Person/row[pid=$a/pid]/name/data()
    }</author>
  }</article > }</bib>

```

This code converts the entire canonical view of the relational database into an XML tree that has the following type:

```

element bib {
  element article {
    attribute aid { xs: int },
    element title { xs: string }, element year { xs: string },
    element author { attribute pid { xs: int }, xs: string }* }* }

```

We assume that this view is materialized on a persistent storage, which is typically a native XML database. Our goal is to translate relational updates over the base data to XQuery updates (expressed in XUF) that incrementally modify the stored XML view to make it consistent with the base data. Our framework synthesizes a right-inverse function 'view' of the view function, view, such that, for all valid view instances  $V$ , view(view'(V)) is equal to  $V$ . Since an inverted view reconstructs the relational database from the view, it must preserve the key constraints of the relational schema. This is accomplished with an extension of the FLWOR XQuery syntax that enforces these constraints. In its simplest form, our FLWOR syntax may contain an optional 'unique' part:

```

for $v in e where p($v) unique k($v) return f($v)

```

where  $p(\$v)$ ,  $k(\$v)$ , and  $f(\$v)$  are XQuery expressions that depend on  $\$v$ . The unique  $k(\$v)$  part of the FLWOR syntax skips the duplicate nodes  $\$v$  from  $e$ , where the equality for duplicate removal is modulo the key function  $k(\$v)$ . More specifically, let the result of  $e$  be the sequence of nodes  $(x_1, \dots, x_n)$ . Then the result of this FLWOR expression is the sequence concatenation  $(y_1, \dots, y_n)$ , where

$$y_i = \begin{cases} () & \text{if } \neg p(x_i) \\ () & \text{if } \exists j < i : k(x_j) = k(x_i) \\ f(x_i) & \text{(otherwise)} \end{cases}$$

Based on this extended syntax, we can define a function  $\text{view}'(\$V)$  that is precisely the right-inverse of the previous view:

```
view'($V) = <DB><Article>{ for $ii in $V/article
    unique $ii/@aid/data()
    return <row><aid>{$ii/@aid/data()}</aid>
        {$ii / title , $ii / year}</> }</Article>
<Author>{ for $ia in $V/article , $aa in $ia/author
    unique ($ia/@aid/data() , $aa/@pid/data() )
    return <row><aid>{$ia/@aid/data()}</aid>
        <pid>{$aa/@pid/data()}</pid></row> }</Author>
<Person>{ for $ip in $V/article , $ap in $ip/author
    unique $ap/@pid/data()
    return <row><pid>{$ap/@pid/data()}</pid>
        <name>{$ap/data()}</name>
        <email>*</email></row> }</Person></DB>
```

In fact, our framework is capable of synthesizing this function automatically from the view function for many different forms of XQuery code. It will also generate the unique constraints automatically from the DB schema key constraints. For example, our system will infer that the applicable key for the first for-loop in  $\text{view}'(\$V)$  is  $\text{key } \{ /DB/Article/row, aid/data() \}$ , given that the for-loop body has the same type as  $/DB/Article/row$  and is inside DB and Article element constructions. By applying the field path  $./aid/data()$  to the for-loop body and normalizing it, our system will add the constraint **unique**  $\$ii/@aid/data()$  to the for-loop. (Note that this heuristic method is applicable to relational key constraints only, where the key selector type corresponds to a relational table.) We can see that the unique parts of the for-loops in the inverted view are necessary in order to reconstruct the relational data without duplicate key values. The star in the email element content indicates that it can be any value, since this value is not used in the view. Stars in the inverted view signify that the view itself is a surjective function, so that we may not be able to reconstruct the complete database from the view. As we will show next, when our framework synthesizes the incremental view updates, the results of the inverted view must always pass through the view function to derive a view-to-view function, which eliminates the star values.

To show that  $\text{view}(\text{view}'(\$V))$  is indeed equal to  $\$V$ , we have to use normalization rules and the properties of the key constraints. More specifically,  $\text{view}(\text{view}'(\$V))$  is:

```
<bib>{
  for $i in view'($V)/Article /row
```

```

return < article aid='{ $i/aid/data()}' >{
    $i/ title , $i/year ,
    for $a in view'($V)/Author/row[aid=$i/aid]
    return <author pid='{ $a/pid/data()}' >{
        view'($V)/Person/row[pid=$a/pid]/name/data() }</author>
    }</ article > }</bib>

```

XQuery normalization reduces general XQueries to normal forms, such as to for-loops whose domains are simple XPath. In addition to fusing layers of XQuery compositions and eliminating their intermediate results, normal forms are simpler to analyze than their original forms. Normalization can be accomplished with the help of rules, such as:

$$\mathbf{for\ \$v\ in\ (for\ \$w\ in\ e_1\ return\ e_2)\ return\ e_3 = for\ \$w\ in\ e_1,\ \$v\ in\ e_2\ return\ e_3}$$

to normalize the for-loop domain. If we expand the view'(\$V) definition and normalize the resulting code, we get the following code:

```

< bib > {
  for $ii in $V/ article
  unique $ii /@aid/data()
  return < article aid='{ $ii /@aid/data()}' > {
    $ii / title , $ii / year ,
    for $ia in $V/ article , $aa in $ia/ author
    where $ia /@aid/data() = $ii /@aid/data ()
    unique ( $ia /@aid/data () , $aa /@pid/data () )
    return < author pid='{ $aa /@pid/data()}' > {
      for $ip in $V/ article , $ap in $ip/ author
      where $ap /@pid/data() = $aa /@pid/data ()
      unique $ap /@pid/data ()
      return $ap / data ()
    }</ author >
  }</ article > }</ bib >

```

To simplify this code further, we would need to take into account the key constraints expressed in the **unique** parts of the for-loops. A general rule that eliminates nested for-loops is:

$$\begin{aligned}
 & \mathbf{for\ \$v_1\ in\ f_1(\$v), \dots, \$v_n\ in\ f_n(\$v)} \\
 & \mathbf{return\ g( for\ \$w_1\ in\ f_1(\$w), \dots, \$w_n\ in\ f_n(\$w)} \\
 & \quad \mathbf{where\ key(\$w) = key(\$v)\ unique\ key(\$w)\ return\ h(\$w) )} \\
 & = \mathbf{for\ \$v_1\ in\ f_1(\$v), \dots, \$v_n\ in\ f_n(\$v)\ return\ g( h(\$v) )}
 \end{aligned}$$

where  $key, g, h, f_1, \dots, f_n$  are XQuery expressions that depend on the XQuery variables,  $\$v$  and  $\$w$ , which are the variables  $\$v_1, \dots, \$v_n$  and  $\$w_1, \dots, \$w_n$ , respectively. This rule, indicates that if we have a nested for-loop, where the inner for-loop variables have the same domains as those of the outer variables, modulo variable renaming (that is,  $f_i(\$v)$  and  $f_i(\$w)$ , for all  $i$ ), and they have the same key values, for some key function, then the values of  $\$w$  must be identical to those of  $\$v$ , and therefore the inner for-loop can be eliminated. We can apply this rule to our previous normalized code, by noticing that the outer for-loop is over the variables  $\$ia$  and  $\$aa$  and the inner for-loop is over the variables  $\$ip$  and  $\$ap$ , where the key is  $\$ap/@pid/data()$  and the predicate is  $\$ap/@pid/data()=\$aa/@pid/data()$ . This means that the code can be simplified to:

```

<bib>{
  for $ii in $V/ article
  unique $ii/@aid/data()
  return < article aid='{ $ii/@aid/data()}'>{
    $ii / title , $ii /year,
    for $ia in $V/ article , $aa in $ia/author
    where $ia/@aid/data()=$ii /@aid/data()
    unique ($ia/@aid/data() , $aa/@pid/data())
    return <author pid='{ $aa/@pid/data()}'>{ $aa/ data () }</author>
  }</ article > }</bib>

```

Similarly, we can apply the same rule to the resulting XQuery by noticing that the outer for-loop is over the variable \$ii and the inner for-loop is over the variable \$ia, where the key is \$ia/@aid/data(), and simplify the code further:

```

ID($V) = <bib>{
  for $ii in $V/ article
  unique $ii/@aid/data()
  return < article aid='{ $ii/@aid/data()}'>{
    $ii / title , $ii /year,
    for $aa in $ii /author
    return <author pid='{ $aa/@pid/data()}'>{ $aa/ data () }</author>
  }</ article > }</bib>

```

ID(\$V) is precisely the copy function applied to the view \$V, which copies every node in the view. That is, ID(\$V) is equal to \$V.

Consider now an SQL update over the base relational tables:

```

update Article set year=2009
where exists ( select * from Author a, Person p
  where a.aid=aid and a.pid=p.pid and p.name='Smith' and title ='XQuery')

```

which finds all articles authored by Smith that have XQuery as title and replaces their year with 2009. This update can be written in plain XQuery U(\$DB) over the canonical XML view \$DB of the relational database that reconstructs the database reflecting the updates:

```

<DB><Article>{
  for $iu in $DB/Article/row
  return if some $au in $DB/Author/row, $pu in $DB/Person/row
    satisfies $au/aid = $iu/aid and $au/pid = $pu/pid
    and $pu/name = 'Smith' and $iu/ title = 'XQuery'
  then <row>{ $iu/aid, $iu/ title }<year>2009</year></row>
  else $iu
}</Article>{$DB/Person, $DB/Author}</DB>

```

The new view after the update can be reconstructed from the old view \$V using view(U(view'(\$V))). First, after normalization, U(view'(\$V)) becomes:

```

<DB><Article>{
  for $ii in $V/ article
  return if some $aa in $ii /author
    satisfies $aa/ data () = 'Smith' and $ii/ title = 'XQuery'

```



```

then <row><aid>{$ii/@aid/data()}</aid>{$ii/title,<year>2009</year>}</row>
else <row><aid>{$ii/@aid/data()}</aid>{$ii/title,$ii/year}</row>
}</Article>{...}</DB>

```

where ... are the Author and Person parts of view'(\$V) (they are not affected by the update). Similarly, after normalization, view(U(view'(\$V))) becomes:

```

<bib>{ for $ii in $V/ article
      return if some $aa in $ii /author
          satisfies $aa/data() = 'Smith' and $ii/ title = 'XQuery'
      then < article aid='{ $ii /@aid/data()}'>{
          $ii / title , <year>2009</year>,
          for $aa in $ii /author
          return <author pid='{ $aa/@pid/data()}'>{$aa/data()}</author>
      }</ article >
      else $ii }</bib>

```

Compare now this XQuery with the view copy function, ID(\$V) (the result of view(view'(\$V)) found previously). We can see that, for each \$ii, if the if-then-else condition is true, then their article elements are identical, except for the \$ii/year component, which is <year>2009</year> in the new view. If we ignore the identical components, we can derive view(U(view'(\$V))) from the original view \$V by just updating those components that differ. This means that the XUF update:

```

for $ii in $V/ article
return if some $aa in $ii /author ,
    satisfies $aa/data() = 'Smith' and $ii/ title = 'XQuery'
then replace node $ii/year with <year>2009</year>
else ()

```

will destructively modify the view to become the modified view after the update U. In the extended version of this paper [8], we present a heuristic, conservative algorithm that, given a view mapping from the old view to the new view, finds those program fragments that differ from the identity view mapping and generates an XUF update for each one. For this algorithm to be effective, it is essential that the view mapping program be normalized to make program equivalence more tractable (which is undecidable in general). It basically compares if the two programs are identical, modulo variable renaming, but it also takes into account the possible alternatives in sequence construction.

## 4 Synthesizing the Right-Inverse

Given that XQuery is computationally complete, one of hardest tasks for updating materialized views is to synthesize the inverse function of a view expressed in XQuery. This task becomes even harder when the underlying database is relational because the view must be expressed in terms of joins, which in general do not have an inverse function. In this section, we briefly sketch our heuristic algorithm that synthesizes the inverse of a view mapping. The full details of the algorithm as well as the proof of its correctness are given in the extended version of this paper [8]. Our inversion algorithm can handle many XQuery forms that are used frequently in real view mapping scenarios, including

views that use one-to-one and one-to-many joins. Our program synthesis algorithm is guided by the type (schema) of the view source code, which can be inferred from the input schema of the view (the schema of the underlying relational database).

Given an XQuery expression,  $f(x)$ , that depends on the variable  $x$  (a shorthand for the XQuery variable  $\$x$ ), the right-inverse  $\mathcal{I}_x(f(x), y)$  is an XQuery expression  $g(y)$  that depends on  $y$ , such that  $y = f(x) \Rightarrow x = g(y)$ . This definition implies that  $f(g(y)) = y$ , which means that  $g(y) = \mathcal{I}_x(f(x), y)$  is the right-inverse of  $f(x)$ . In this Section, we present the rules for extracting  $\mathcal{I}_x(e, y)$  for most forms of XQuery expression  $e$ . Some  $\mathcal{I}_x$  rules may return an error, which is denoted by  $\perp$ . For example, if the view is  $y = \text{if } x > 4 \text{ then } x-1 \text{ else } x+2$  then the right-inverse is:

**$x = \text{if } y+1 > 4 \text{ then (if } y-2 > 4 \text{ then } y+1 \text{ else } \perp) \text{ else } y-2$**

that is, if  $y$  is equal to 4, 5, or 6, then  $x$  must be a  $\perp$  value since this  $y$  can be produced in two different ways (e.g., both  $x=2$  and  $x=5$  produce  $y=4$ ). Some rules may also return  $*$ , which indicates that it can be any XML node of the proper type. For example, if the type of  $x$  is element A { element B xs:int, element C xs:int }, then the right-inverse of  $y = x/C$  is  $x = \langle A \rangle \{ *, y/\text{self}::C \} \langle /A \rangle$ , which indicates that  $x/B$  can be any B element.

The most difficult expression to invert is XQuery sequence concatenation, because, in contrast to regular tuples and records, nested sequences are flattened out to sequences of XML nodes or base values. For example, if the type of  $x$  is element A { element B xs:int, element C xs:int }, then, for  $y = (x/B, x/C)$ , we would like to derive  $x = \langle A \rangle \{ y/\text{self}::B, y/\text{self}::C \} \langle /A \rangle$ . That is, since  $y_1 = x_1/B$  implies  $x_1 = \langle A \rangle \{ y_1/\text{self}::B, * \} \langle /A \rangle$  and  $y_2 = x_2/C$  implies  $x_2 = \langle A \rangle \{ *, y_2/\text{self}::C \} \langle /A \rangle$ , the goal is to derive the previous solution for  $y = (x/B, x/C) = (y_1, y_2)$ . But we must have  $x=x_1=x_2$ , in order to have a valid solution for  $x$ . By looking at  $x_1$  and  $x_2$ , this can only be done if we match  $x_1$  with  $x_2$ , since  $*$  matches with any expression. In fact, as we will see next, we would need a unification algorithm that also matches variables with expressions by binding these variables to these expressions.

Consider now the inversion of a for-loop, such as

**$y = \text{for } \$v \text{ in } x/C/\text{data}() \text{ return } \$v+1$**

We can see that  $y$  must be equal to the sequence of integers  $\$v+1$ . Thus, each value  $\$v$  must be equal to an element of  $y$  minus one, which implies that

**$(\text{for } \$w \text{ in } y \text{ return } \$w-1) = x/C/\text{data}()$**

which can be solved for  $x$  since it has the form  $y' = x/C/\text{data}()$ . Therefore, to invert a for-loop, we must first invert the for-loop body with respect to the for-loop variable, and then invert the for-loop domain with respect to  $x$ . This rule works correctly if the for-loop body does not refer to a non-local variable (a variable other than the for-loop variable). But references to non-local variables are very common in a view mapping over a relational database, as we can see from the `view($DB)` example in Section 3. In fact, these non-local references correspond to joins. Consider, for example, the following non-local reference to  $\$x$  in the body of a for-loop on  $\$y$ :

**$\text{for } \$x \text{ in } e1$   
 **$\text{return } \langle A \rangle \{ \dots \text{for } \$y \text{ in } e2 \text{ return } \langle B \rangle \{ \dots \$x \dots \} \langle /B \rangle \dots \} \langle /A \rangle$****

When we invert the body of the inner loop  $\langle B \rangle \dots \$x \dots \langle /B \rangle$  with respect to the loop variable  $\$y$ , we encounter the non-local variable  $\$x$ . Then, at this point of the inversion process, instead of a solution  $\$y = \dots$ , we get a solution  $\$x = \dots$ . This solution must be matched (unified) with the solution for  $\$x$  found in other places in the body of the outer for-loop. We use a special binding list, called the Inverted Vars ( $\mathcal{IV}$ ) that binds variables to their inverted expressions. When we find a new non-local contribution, such as the previous  $\$x = \dots$ , we unify it with the existing binding of the variable in  $\mathcal{IV}$ , if exists. This contribution to a for-loop variable from  $\mathcal{IV}$  is added to the solution found by inverting the for-loop body with respect to this variable.

The case that really needs unification instead of matching is when inverting equality predicates in FLWOR or XPath conditions. Consider, for example, the inversion  $y = \$x/\text{Person}/\text{row}[\text{pid}=\$a/\text{pid}]$  taken from the view( $\$x$ ) definition in Section 3. The condition  $\text{pid}=\$a/\text{pid}$  is very important because it specifies a join between Author and Person. This predicate should provide a new contribution to the  $\$a$  solution that has its  $\$a/\text{pid}$  equal to the Person's  $\text{pid}$ . This can be done by solving  $y = \$x/\text{Person}/\text{row}$ , then solving  $w = \$a/\text{pid}$  and  $w = y/\text{pid}$ , where  $w$  is a new (fresh) unification variable, and then unifying the two solutions to yield a binding for  $w$  that makes the predicate  $y/\text{pid}=\$a/\text{pid}$  true. That is,  $y = \$x/\text{Person}/\text{row}$  will give us a solution for  $y$ , then  $w = y/\text{pid}$  will give us a solution for  $w$ , and finally  $w = \$a/\text{pid}$  will give us a contributing solution for  $\$a$ .

Our unification algorithm for XQuery expressions is based on XQuery code equality. Given that code equality is undecidable in general, our unification algorithm uses heuristics to check for identical XQuery program structures, modulo variable renaming. For example, a for-loop can be unified with another for-loop as long as their corresponding for-loop domains, predicates, and bodies unify. In addition,  $*$  unifies with anything while the unification of a variable with an expression binds the variable to the expression (as long as there are no cyclic references). The following are some of the unification rules. The outcome of the unification  $\text{unify}(e_1, e_2)$  is either  $\perp$  (failure to unify) or the unified expression along with possible new bindings from XQuery variables (which are the unification variables) to their bindings (XQuery expressions).

We are now ready to give some of the rules for  $\mathcal{I}_x$ . The simplest case of an XQuery expression is a variable. If this variable is equal to the inversion variable  $x$ , then  $x = y$ :

$$\mathcal{I}_x(\$x, y) = y \tag{1}$$

If the variable is different from  $x$ , then

$$\mathcal{I}_x(\$v, y) = * \tag{2}$$

that is, the solution for  $x$  is  $*$ , which does not contribute any information about  $x$ . But, as a side-effect, the Inverted Vars,  $\mathcal{IV}$ , is extended with the binding from  $v$  to  $y$ , if  $v$  is not already bound in  $\mathcal{IV}$ ; if  $v$  has already a binding in  $\mathcal{IV}$ , which is accessed by  $\mathcal{IV}[v]$ , it is replaced with  $\text{unify}(\mathcal{IV}[v], y)$ . That is, we are contributing a new solution  $y$  to  $v$ .

XQuery constants do not contribute any solution to  $x$  but they pose a restriction on the  $y$  value:

$$\mathcal{I}_x(c, y) = \text{if } y = c \text{ then } * \text{ else } \perp \tag{3}$$

Simple arithmetic expressions with a constant operand can be easily inverted in a straightforward way. For example:

$$\mathcal{I}_x(e + c, y) = \mathcal{I}_x(e, y - c) \quad (4)$$

For an if-then-else expression **if**  $p$  **then**  $e_1$  **else**  $e_2$ , we invert both the true and false parts:  $x_1 = \mathcal{I}_x(e_1, y)$  and  $x_2 = \mathcal{I}_x(e_2, y)$ . If  $x_1$  unifies with  $x_2$ , then

$$\mathcal{I}_x(\text{if } p \text{ then } e_1 \text{ else } e_2, y) = \text{unify}(x_1, x_2) \quad (5)$$

otherwise, we would have to consider the predicate value for  $x = x_1$  and  $x = x_2$ :

$$\mathcal{I}_x(\text{if } p(\$x) \text{ then } e_1 \text{ else } e_2, y) = \text{if } p(x_1) \text{ then } (\text{if } p(x_2) \text{ then } x_1 \text{ else } \perp) \text{ else } x_2 \quad (6)$$

That is, if  $p(x_1)$  is true, then  $y$  must be equal to  $e_1$ , but if in addition  $p(x_2)$  is false, then  $y$  must also be equal to  $e_2$ , which means that  $x$  must be equal to both  $x_1$  and  $x_2$ , which is assumed false.

The inverse of an XPath step of the form  $e/\text{axis}::\text{test}$  can be found by considering the inferred type of the XQuery expression  $e$ . If the type of  $e$  is inferred to be: element  $B \{ \dots, \text{element } A \ t, \dots \}$ , then:

$$\mathcal{I}_x(e/A, y) = \mathcal{I}_x(e, \langle B \rangle \{ *, \dots, y/\text{self}::A, \dots, * \} \langle /B \rangle) \quad (7)$$

That is,  $e$  must be an element construction whose all but the  $A$  children are  $*$ , while the  $A$  children must satisfy  $e/A = y$ . The other forward XPath steps of the form  $e/\text{axis}::\text{test}$  can be handled in the same way: since we can infer the type of  $e$ , we can embed  $y$  into a number of element constructions,  $c(y)$ , so that  $c(y)/\text{axis}::\text{test} = y$ . In addition, if the schema is not recursive, backward steps can always be translated to equivalent forward steps.

For the element content, we have:

$$\mathcal{I}_x(e/\text{data}(), y) = \mathcal{I}_x(e, \langle B \rangle \{ y \} \langle /B \rangle) \quad (8)$$

given that  $e$  is inferred to be of type: element  $B \ t$ . Finally, for an element construction, we have:

$$\mathcal{I}_x(\langle A \rangle \{ e \} \langle /A \rangle, y) = \mathcal{I}_x(e, y/\text{self}::A/\text{node}()) \quad (9)$$

which imposes the restriction that  $y$  be an element tagged  $A$  and gives the solution that  $e$  be the  $y$  element content.

As we discussed at the beginning of this section, we can invert a for-loop by inverting the loop body with respect to the for-loop variable, and then invert the for-loop domain with respect to  $x$ , given that  $y$  is now the result of the inverted loop body:

$$\mathcal{I}_x(\text{for } \$v \text{ in } e_1 \text{ return } e_2, y) = \mathcal{I}_x(e_1, \text{for } \$v' \text{ in } y \text{ return } \mathcal{I}_v(e_2, \$v')) \quad (10)$$

For example, if  $y = \text{for } \$v \text{ in } x \text{ return } \$v+1$ , then we invert  $\$v = \$v+1$  with respect to  $\$v$  to get  $\$v = \$v'-1$  and we get  $x = \text{for } \$v' \text{ in } y \text{ return } \$v'-1$ . Note that, by definition,  $\$v = \mathcal{I}_v(e_2, \$v')$ . After this solution for  $\$v$  is calculated, we may have a binding  $\mathcal{IV}[v]$  in Inverted Vars that accumulates non-local references to the variable  $\$v$  in  $e_2$ , as we discussed earlier (for example, when  $\$v$  is referenced in the body of an inner for-loop in  $e_2$ ). These contributions must be merged using  $\text{unify}(\mathcal{I}_v(e_2, \$v'), \mathcal{IV}[v])$ , which unifies all solutions to  $\$v$ . Then, Rule (10) becomes:

$$\begin{aligned} & \mathcal{I}_x(\text{for } \$v \text{ in } e_1 \text{ return } e_2, y) \\ &= \mathcal{I}_x(e_1, \text{for } \$v' \text{ in } y \text{ return } \text{unify}(\mathcal{I}_v(e_2, \$v'), \mathcal{IV}[v])) \end{aligned} \quad (10')$$

Also note that Rule (10) can only apply if the loop body is not of a sequence type. For example, it cannot invert  $y = \text{for } \$v \text{ in } x \text{ return } (\$v, \$v+1)$ . Finally, if the for-loop has a predicate,  $p(\$v)$ , that depends on the loop variable, we move the predicate to the loop domain and use Rule (12) described next:

$$\begin{aligned} & \mathcal{I}_x(\text{for } \$v \text{ in } e_1 \text{ where } p(\$v) \text{ return } e_2, y) \\ &= \mathcal{I}_x(\text{for } \$v \text{ in } e_1[p(\cdot)] \text{ return } e_2, y) \end{aligned} \quad (11)$$

Inverting an equality predicate,  $y = e[e_1 = e_2]$ , could be as easy as inverting  $y = e$ . The equality  $e_1 = e_2$  though may give us more information about the inverse code since it relates data produced by two different places in the view code. The most common example is a nested for-loop, which corresponds to a relational join, as we saw in Section 3. Thus, in addition to the solution  $\mathcal{I}_x(e[e_1 = e_2], y) = \mathcal{I}_x(e, y)$ , we have more contributions from the predicate. To calculate these contributions using our inversion algorithm, we let  $\$z$  be the current context of the predicate (the XQuery dot), where  $\$z$  is an XQuery variable, we let  $\$w$  be the result of each branch of the equality, where  $\$w$  is another XQuery variable, and we invert  $\$w = e_1$  and  $\$w = e_2$  with respect to  $\$z$  using  $\mathcal{I}_z(e_1, \$w)$  and  $\mathcal{I}_z(e_2, \$w)$ . All these solutions are unified, which provide a binding for  $\$w$ , and which in turn is used to eliminate  $\$w$  from the solutions. More specifically:

$$\mathcal{I}_x(e[e_1 = e_2], y) = \mathcal{I}_x(e, \text{unify}(\text{unify}(y, \mathcal{I}_z(e'_1, \$w)), \mathcal{I}_z(e'_2, \$w))) \quad (12)$$

where  $e'_1/e'_2$  is equal to  $e_1/e_2$  with the current context (the dot) replaced with  $\$z$ .

Consider, for example, inverting a part of the view defined in Section 3:

$\mathcal{I}_x( y, \$x/\text{Author}/\text{row}[\text{aid}=\$i/\text{aid}] )$

Based on Rule (12), it is equal to

$\mathcal{I}_x( \$x/\text{Author}/\text{row}, \text{unify}( \text{unify}( y, \mathcal{I}_z(\$z/\text{aid}, \$w) ), \mathcal{I}_z(\$i/\text{aid}, \$w) ) )$

for some new variables  $\$z$  and  $\$w$ . Using our inversion rules,  $\mathcal{I}_z(\$z/\text{aid}, \$w)$  gives the solution  $\$z = \langle \text{row} \rangle \{ \$w/\text{self}::\text{aid}, * \} \langle / \text{row} \rangle$  (an Author row), while  $\mathcal{I}_z(\$i/\text{aid}, \$w)$  gives the solution  $\$z = *$ , but accumulates the contribution  $\$i = \langle \text{row} \rangle \{ \$w/\text{self}::\text{aid}, * \} \langle / \text{row} \rangle$  (an Article row). Given that  $y = \langle \text{row} \rangle \{ y/\text{aid}, y/\text{pid} \} \langle / \text{row} \rangle$ , if the solutions for  $\$z$  are unified with  $y$ , we get  $\$w = y/\text{aid}$  and the contribution to  $\$i$  becomes  $\langle \text{row} \rangle \{ y/\text{aid}, * \} \langle / \text{row} \rangle$ .

As we discussed earlier, the most difficult case to handle is XQuery sequence concatenation because nested sequences are flattened out to sequences of XML nodes or base values. Let  $y = (e_1, e_2)$  (recall that in XQuery a sequence  $(e_1, e_2, \dots, e_n)$  can be written using binary sequence concatenations  $((e_1, e_2), \dots, e_n)$ ). If the types of  $e_1$  and  $e_2$  are basic types (such as strings), then  $y$  must be a sequence of two values,  $(y[1], y[2])$ . In that case, let  $x_1$  be the solution of  $y[1] = e_1$  and  $x_2$  be the solution of  $y[2] = e_2$ . Then, the solution for  $x$  must be  $\text{unify}(x_1, x_2)$ , which will fail if  $x_1$  and  $x_2$  are incompatible. In general,  $e_1$  and  $e_2$  can be of any type, including sequence types. What is needed is a method to split  $y$  into two components  $y_1$  and  $y_2$  so that  $y_1/y_2$  have the same type as  $e_1/e_2$ , respectively. Then the inverse of  $y = (e_1, e_2)$  would be the unification of the inverses for  $y_1 = e_1$  and  $y_2 = e_2$ . This splitting is accomplished with the function  $\text{split}(t_1, t_2, y)$  that derives two predicates,  $p_1$  and  $p_2$ , to break  $y$  into two components  $(y_1, y_2)$  so that  $y_1 = y[p_1]$  and  $y_2 = y[p_2]$ , and the types of  $y_1/y_2$  match those of  $e_1/e_2$ , respectively. The inverse rule for sequences is:

$$\mathcal{I}_x((e_1, e_2), y) = \text{unify}(\mathcal{I}_x(e_1, y[p_1]), \mathcal{I}_x(e_2, y[p_2])) \quad (13)$$

where  $e_1/e_2$  have types  $t_1/t_2$  and  $(p_1, p_2) = \text{split}(t_1, t_2, \cdot)$ . Here are some of the rules for the split function:

$$\text{split}(\text{element } A \ t_1, \text{element } A \ t_2, y) = \text{split}(t_1, t_2, y/\text{self}::A/\text{node}()) \quad (S1)$$

$$\text{split}(\text{element } A \ t_1, \text{element } B \ t_2, y) = (y/\text{self}::A, y/\text{self}::B) \quad (S2)$$

$$\text{split}(xs:\text{string}, xs:\text{string}, y) = (y[1], y[2]) \quad (S3)$$

$$\text{split}(t_1, t_2, y) = \perp \quad (\text{otherwise}) \quad (S4)$$

Rule (S1) indicates that if the types we try to discriminate are elements with the same tag, then we can only split  $y$  if we consider the content of  $y$ . On the other hand, Rule (S1) indicates that if the types are elements with different tags,  $A$  and  $B$ , then we can simply split  $y$  to  $y/\text{self}::A$  and  $y/\text{self}::B$ .

For example, from Rules (13) and (S3), the inverse of  $y = (x-1, x*2)$  is  $\text{unify}(\mathcal{I}_x(y[1], x-1), \mathcal{I}_x(y[2], x*2))$ , which is equal to  $\text{unify}(y[1]+1, y[2]/2)$ . Finally, after unification, we get  $x = \text{if } y[1]+1=y[2]/2 \text{ then } y[1]+1 \text{ else } \perp$ .

## 5 Implementation and Evaluation

The algorithms described in this paper have already been implemented in Haskell and the presented examples have been tested. The source code is available at:

<http://lambda.uta.edu/MaterializedViews.hs>

Although this code has not been incorporated into our XQuery DBMS, HXQ [6], we have used HXQ to evaluate the feasibility of our approach on incremental view maintenance. HXQ is a fast and space-efficient translator from XQuery to embedded Haskell code. It takes full advantage of Haskell's lazy evaluation to keep in memory only those parts of XML data needed at each point of evaluation, thus performing stream-based evaluation for forward queries. In addition to processing XML files, HXQ can store XML documents in a relational database by shredding XML into relational tuples and

view #	size (MBs)	tuples $\times 1000$	view recreation time (secs)	view update time (msecs)
0	14	130	164	23
1	29	290	191	22
2	47	470	233	27
3	70	380	215	20
4	98	540	256	19
5	130	730	310	21
6	162	910	371	24
7	185	1040	416	32
8	188	1060	423	33
9	191	1070	429	42
10	194	1090	439	39
11	198	1110	448	47
12	203	1140	459	53
13	209	1170	472	69
14	216	1210	491	68
15	223	1250	511	75
16	230	1290	530	72
17	239	1350	556	78
18	249	1410	588	80
19	262	1490	625	90

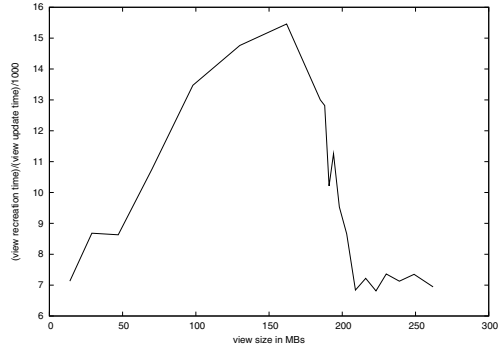


Fig. 1. A) Evaluation over a Number of XML Views. B) View Update vs. View Recreation

by translating XQueries over the shredded documents into embedded optimized SQL queries. The mapping to relational tables is based on the document’s structural summary, which is derived from the document data rather than from a schema. It uses hybrid inlining to inline attributes and non-repeating elements into a single table, thus resulting to a compact relational schema.

Our experimental setup consists of a 2.2GHz Intel Core 2 Duo with 2GB memory, running Haskell ghc-6.10.3 on a 32-bit Linux 2.6.27 kernel. Our experimental XML data were derived from the DBLP XML document [5] of size 420MB. We fed this document into our HXQ DBMS, which shredded and stored the DBLP data into the MySQL database. The resulting MySQL database consists of 4.3 million tuples. Then, we created 20 materialized XML views of this database,  $view_i$ ,  $i \in [0, 19]$ , using the following view mappings:

$$view_i = \langle dblp \rangle \{ \$DB // inproceedings [ year \bmod 20 \leq i ] \} \langle /dblp \rangle$$

that is, each  $view_i$  had progressively more elements, with  $view_{19}$  the largest that contained all inproceedings records of DBLP. Each XML view was materialized into a relational database in MySQL. More specifically, the results of each view mapping was dumped to an XML document, and this document was parsed, shredded, and stored by HXQ into MySQL automatically. Then, for each view,  $view_i$ , we evaluated the following XQuery update:

$$U_i = \text{for } \$x \text{ in } view_i // inproceedings [ author='Leonidas Fegaras' ] \\ \text{return replace value of node } \$x/year \text{ with } 2000$$

Figure 1.A shows our results for each  $view_i$ . The second column is the XML document size in MBs generated for each view, while the third column is the number of tuples (in thousands) in the materialized view. The fourth column is the time in seconds needed to create the materialized view from scratch by 1) evaluating the view query against the base relational database, 2) by dumping the view results to an XML document, and

3) by shredding and storing the resulting document into the materialized view. This would also be the time needed to recreate the view after each update. The fifth column is the time in milliseconds needed for each update  $U_i$  against the materialized view. Figure 11B draws the ratio between the fourth and fifth column data (divided by 1000) against the second column (view size). We can see that the performance gain of updating the materialized view against recreating the view is between 7,000 and 15,000 times.

## 6 Conclusion

At a first glance, it may seem counter-intuitive to transform an efficient program with updates to an inefficient program that reconstructs the entire database, just to translate it back to updates at the end. But, as we show in this paper, this is a very effective way to incorporate source updates into a view, without requiring any substantial modification to the existing XQuery model. Since it is based on compositional transformations that are easy to validate, our approach has the potential of becoming a general methodology for incremental view maintenance.

## References

1. Abiteboul, S., Bourhis, P., Marinoiu, B.: Efficient Maintenance Techniques for Views over Active Documents. In: EDBT (2009)
2. Abiteboul, S., McHugh, J., Rys, M., Vassalos, V., Wiener, J.L.: Incremental Maintenance for Materialized Views over Semistructured Data. In: VLDB 1998 (1998)
3. Arion, A., Benzaken, V., Manolescu, I., Papakonstantinou, Y.: Structured Materialized Views for XML Queries. In: VLDB 2007 (2007)
4. Braganholo, V.P., Davidson, S.B., Heuser, C.A.: From XML View Updates to Relational View Updates: old solutions to a new problem. In: VLDB 2004 (2004)
5. DBLP XML records, the DBLP Computer Science Bibliography, <http://dblp.uni-trier.de/xml/>
6. Fegaras, L.: Propagating Updates through XML Views using Lineage Tracing. In: ICDE 2010 (2010)
7. Fegaras, L.: A Schema-Based Translation of XQuery Updates. In: Lee, M.L., Yu, J.X., Belahsène, Z., Unland, R. (eds.) XSym 2010. LNCS, vol. 6309, pp. 58–72. Springer, Heidelberg (2010), <http://lambda.uta.edu/xuf10.pdf>
8. Fegaras, L.: Incremental Maintenance of Materialized XML Views (Extended Paper), <http://lambda.uta.edu/views10.pdf>
9. Fernandez, M., Kadiyska, Y., Suciu, D., Morishima, A., Tan, W.-C.: SilkRoute: A Framework for Publishing Relational Data in XML. In: TODS 2002, vol. 27(4) (2002)
10. Foster, J.N., Konuru, R., Simeon, J., Villard, L.: An Algebraic Approach to XQuery View Maintenance. In: PLAN-X 2008 (2008)
11. Gupta, A., Mumick, I.S.: Maintenance of Materialized Views: Problems, Techniques, and Applications. IEEE Bulletin on Data Engineering 18(2) (1995)
12. El-Sayed, M., Wang, L., Ding, L., Rundensteiner, E.A.: An Algebraic Approach for Incremental Maintenance of Materialized XQuery Views. In: WIDM 2002 (2002)
13. Onose, N., Borkar, V.R., Carey, M.: Inverse Functions in the AquaLogic Data Services Platform. In: VLDB 2007 (2007)



14. Sawires, A., Tatemura, J., Po, O., Agrawal, D., El-Abbadi, A., Candan, K.S.: Maintaining XPath Views in Loosely Coupled Systems. In: VLDB 2006 (2006)
15. W3C. XML Schema (2000), <http://www.w3.org/XML/Schema>
16. W3C. XQuery Update Facility 1.0. W3C Candidate Recommendation 1 (June 2009), <http://www.w3.org/TR/xquery-update-10/>

# Ingredients for Accurate, Fast, and Robust XML Similarity Joins

Leonardo Andrade Ribeiro<sup>1,\*</sup> and Theo Härder<sup>2</sup>

<sup>1</sup> Department of Computer Science,  
Federal University of Lavras, Brazil  
laribeiro@dcc.ufla.br

<sup>2</sup> AG DBIS, Department of Computer Science,  
University of Kaiserslautern, Germany  
haerder@cs.uni-kl.de

**Abstract.** We consider the problem of answering similarity join queries on large, non-schematic, heterogeneous XML datasets. Realizing similarity joins on such datasets is challenging, because the semi-structured nature of XML substantially increases the complexity of the underlying similarity function in terms of both effectiveness and efficiency. Moreover, even the selection of pieces of information for similarity assessment is complicated because these can appear at different parts among documents in a dataset. In this paper, we present an approach that jointly calculates textual and structural similarity of XML trees while implicitly embedding similarity selection into join processing. We validate the accuracy, performance, and scalability of our techniques with a set of experiments in the context of an XML DBMS.

**Keywords:** XML, Similarity Join, Fuzzy Duplicate Identification.

## 1 Introduction

As XML continues its path to becoming the universal information model, large-scale XML repositories proliferate. Very often, such XML repositories are non-schematic, or have multiple, evolving, or versioned schemas. In fact, a prime motivation for using XML to directly represent pieces of information is the ability of supporting ad-hoc or “schema-later” settings. For example, the flexibility of XML can be exploited to reduce the upfront cost of data integration services, because documents originated from multiple sources can be stored without prior schema reconciliation and queried afterwards in a best-effort manner.

Of course, the flexibility of XML comes at a price: loose data constraints can also lead to severe data quality problems, because the risk of storing inconsistent and incorrect data is greatly increased. A prominent example of such problems is the appearance of the so-called *fuzzy duplicates*, i.e., multiple and non-identical representations of a real-world entity [1]. Complications caused by

---

\* Work done at the University of Kaiserslautern, Germany.

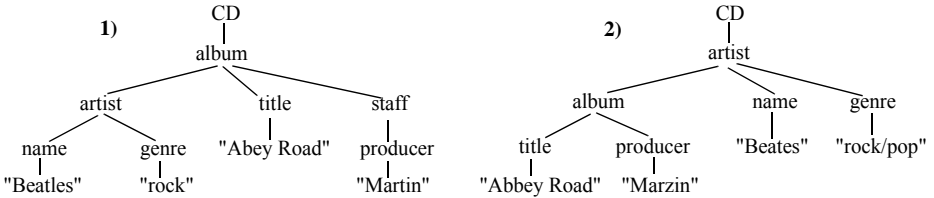


Fig. 1. Heterogeneous XML data

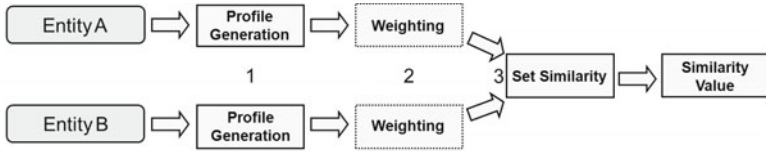
such redundant information abound in common business practices. Some examples are misleading results of decision support queries due to erroneously inflated statistics, inability of correlating information related to the same customer, and unnecessarily repeated operations, e.g., mailing, billing, and leasing of equipment. For relational data, fuzzy duplicate identification is often needed on text fields owing to misspellings and naming variations. For XML — commonly modeled as a labeled tree —, this task is even more critical, because also structure, in addition to text, may present deviations. For instance, consider the sample data from music inventory databases shown in Fig. 1. Subtrees 1) and 2) in Fig. 1 apparently refer to the same CD. However, the use of conventional operators based on exact matching to group together such duplicate data is futile: subtree 1) is arranged according to `album`, while subtree 2) is arranged according to `artist`, there are extra elements (`staff` in subtree 1), and several typos in the content of text nodes (e.g., "Beatles" and "Beates").

In this paper, we present the design and evaluation of an approach to XML similarity joins. In Sect. 2 we present the main ingredients of our approach, which are the decomposition of the computation into three components that can be independently dealt with and a strategy for delimiting textual and structural representations based on XML path similarity. We formally define the logical similarity join operator, which implicitly incorporates similarity selection as sub-operation in Sect. 3. Then, in Sect. 4, we demonstrated that our approach can be applied flexibly to assess the similarity of ordered and unordered trees, deal with large datasets, and deliver accurate results with a set of experiments in the context of an XML DBMS. Related work is discussed in Sect. 5, before we wrap up with the conclusions in Sect. 6.

## 2 Main Ingredients

### 2.1 Similarity Functions

We focus on the class of token-based similarity functions, which ascertains the similarity between two entities of interest by measuring the overlap between their set representations. We call such set representation the *profile* of an entity, the elements of the profile are called *tokens*; optionally, a *weighting scheme* can be used to associate weights to tokens. Token-based similarity functions allow measuring textual and structural similarity in a unified framework and provide a very



**Fig. 2.** Evaluation of token-based similarity functions

rich similarity space by varying profile generation methods, weighting schemes, set similarity functions, or any combination thereof. Moreover, these functions are computationally inexpensive and we can leverage a wealth of techniques for similarity joins on strings (see [2] and references therein). Fig. 2 illustrates the three main components of token-based similarity functions and the evaluation course along them towards a similarity value.

**Profile Generation.** The profile of an entity is generated by splitting its representation into a set of tokens; we call this process *tokenization*. The idea behind tokenization is that most of the tokens derived from significantly similar entities should agree correspondingly. For XML, tokenization can be applied to text, structure, or both. We next describe methods capturing text and structure in isolation; methods that generate tokens conveying both textual and structural information are employed in Sect. 4.

A well-known textual tokenization method is that of mapping a string to a set of  $q$ -grams, i.e., substrings of size  $q$ . For example, the 2-gram profile of the string "Beatles" is {'Be', 'ea', 'at', 'tl', 'le', 'es'}. Structural tokenization methods operate on element nodes capturing labels<sup>1</sup> and relationships. A simple structural (path) tokenization method consists of simply collecting all element node labels of a path. Thus, the profile of the path /CD/album/artist/name would be {'CD', 'album', 'artist', 'name'}. Note that, as described, the result of both tokenization methods could be a multi-set. We convert a multi-set to sets by concatenating the symbol of a sequential ordinal number to each occurrence of a token. Hence, the multi-set {'a', 'b', 'b'} is converted to {a◦1, b◦1, b◦2} (the symbol ◦ denotes concatenation).

**Weighting Schemes.** In many domains, tokens show non-uniformity regarding some semantic properties such as discriminating power. Therefore, the definition of an appropriate weighting scheme to quantify the relative importance of each token for similarity assessment is instrumental in obtaining meaningful similarity results. For example, the widely used *Inverse Document Frequency (IDF)* weights a token  $t$  as follows:  $IDF(t) = \ln(1 + N/f_t)$ , where  $f_t$  is the frequency of token  $t$  in a database of  $N$  documents. The intuition of IDF is that rare tokens usually carry more content information and are more discriminative. Besides statistics, other kinds of information can be used to calculate weights. The

<sup>1</sup> We assume that element labels are drawn from a common vocabulary. Semantic integration of vocabulary (or ontology) is a closely related but different problem from similarity matching of structure and textual content, which is our focus here.

Level-based Weighting Scheme (*LWS*) [3] weights structural tokens according to node nesting depth in a monotonically decreasing way: given a token  $t$  derived from a node at nesting level  $i$ , its weight is given by  $LWS(t) = e^{\beta i}$ , where  $\beta \leq 0$  is a decay rate constant. The intuition behind *LWS* is that in tree-structured data like XML more general concepts are normally placed at lower nesting depths. Hence, mismatches on such low-level concepts suggest that the information conveyed by two trees is semantically “distant”.

**Set Similarity.** Tokenization delivers an XML tree represented as a set of tokens. Afterwards, similarity assessment can be reduced to the problem of set overlap, where different ways to measure the overlap between profiles raise various notions of similarity. In the following, we formally define the *Weighted Jaccard Similarity*, which will be used in the rest of this paper. Several other set similarity measures could however be applied [2].

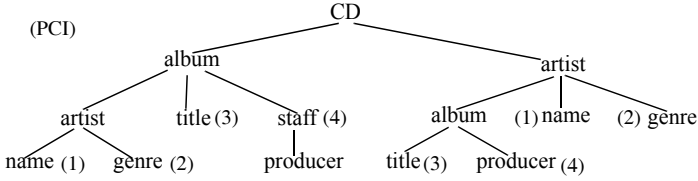
**Definition 1.** Let  $\mathcal{P}_1$  be a profile and  $w(t, \mathcal{P}_1)$  be the weight of a token  $t$  in  $\mathcal{P}_1$  according to some weighting scheme. Let the weight of  $\mathcal{P}_1$  be given by  $w(\mathcal{P}_1) = \sum_{t \in \mathcal{P}_1} w(t, \mathcal{P}_1)$ . Similarly, consider a profile  $\mathcal{P}_2$ . The *Weighted Jaccard Similarity* between  $\mathcal{P}_1$  and  $\mathcal{P}_2$  is defined as  $WJS(\mathcal{P}_1, \mathcal{P}_2) = \frac{w(\mathcal{P}_1 \cap \mathcal{P}_2)}{w(\mathcal{P}_1 \cup \mathcal{P}_2)}$ , where  $w(t, \mathcal{P}_1 \cap \mathcal{P}_2) = \min(w(t, \mathcal{P}_1), w(t, \mathcal{P}_2))$ .

*Example 1.* Consider the profiles  $\mathcal{P}_1 = \{\langle 'Be', 5 \rangle, \langle 'ea', 2 \rangle, \langle 'at', 2 \rangle, \langle 'tl', 2 \rangle, \langle 'le', 1 \rangle, \langle 'es', 4 \rangle\}$  and  $\mathcal{P}_2 = \{\langle 'Be', 5 \rangle, \langle 'ea', 2 \rangle, \langle 'at', 2 \rangle, \langle 'te', 1 \rangle, \langle 'es', 4 \rangle\}$ —note the token-weight association, i.e.,  $\langle t, w(t) \rangle$ . Therefore, we have  $WJS(\mathcal{P}_1, \mathcal{P}_2) \approx 0.76$ .

## 2.2 XML Path Clustering

There are several challenges to realizing similarity joins on heterogeneous XML data. Regarding effectiveness, structural and textual similarities have to be calculated and combined. Text data needs to be specified, because often only part of the available textual information is relevant for similarity matching, and can be only *approximately* selected, because the underlying schema is unknown or too complex. Regarding efficiency, it is important to generate compact profiles and avoid repeated comparisons of structural patterns that may appear many times across different XML documents. Next, we briefly review our approach based on path clustering, which provides the basis for tackling all the issues above. For a detailed discussion, please see [3].

Our approach consists of clustering all path classes of an XML database in a pre-processing step. Path classes uniquely represent paths occurring at least once in at least one document in a database. The similarity function used in the clustering process is defined by the path tokenization method and the *LWS* weighting scheme described earlier and some set similarity function like *WJS*. As a result, we obtain the set  $PC = \{pc^1, \dots, pc^n\}$ , where  $pc^i$  is a cluster containing similar path classes and  $i$  is referred to as *Path Cluster Identifier* (PCI). Given a path  $p$  appearing in some document, we say that  $p \in pc^i$  iff the path class of  $p$  is in  $pc^i$ . Further, we can reduce similarity matching between paths to a simple



**Fig. 3.** Path synopsis annotated with PCI values

equality comparison between their corresponding PCIs, because the actual path comparison has already been performed during the clustering process.

Prior to clustering, all path classes of a database have to be first collected. This can be done in a single pass over the data. Preferably, we can use the so-called *Path Synopsis* (PS), a tree-structured index providing and maintaining a structural summary of an XML database [4]. Each node in a PS represents a (partial) path class. Fig. 3 depicts the PS of the sample database shown in Fig. 1 annotated with PCI values, where similar paths have the same PCI.

PCI values are used to guide the selection of text data that will compose the textual representation of an entity. For this, we define the set  $PC_t \subseteq PC$ : only text nodes appearing under a path in  $PC_t$  are used to generate tokens conveying textual information. We let users specify the  $PC_t$  set by issuing simple *path queries* like  $/a/b/c$ , which are approximately matched against the elements of  $PC$ . The  $K$  path clusters with highest similarity to each path query are selected to form  $PC_t$ . To enable very short response times, path clusters are represented by a single *cluster representative*, to which path queries are compared, and implemented as little memory-resident inverted lists. The  $PC_t$  can be interactively or automatically constructed, in which path queries are embedded into the main similarity join query. In the following, we assume that  $PC_t$  is given.

### 3 Tree Similarity Join

We are now ready to define our Tree Similarity Join (TSJ) operator. This operator takes as input two XML databases and outputs all pairs of XML trees whose similarity is greater than a given threshold.

**Definition 2.** Let  $\mathcal{D}_1$  and  $\mathcal{D}_2$  be two XML databases and  $exp(\mathcal{D})$  be an XPath or XQuery expression over a database  $\mathcal{D}$ . Further, let  $tok$  be a tokenization method that, given a set  $PC_t$  of PCIs, maps an XML tree  $T$  to a profile  $tok[PC_t](T)$ ,  $ws$  be a weighting scheme that associates a weight to every element of a given input set, and  $ss$  be a set similarity measure. Let  $sf$  be the similarity function defined by the triple  $(tok[PC_t], ws, ss)$ , which returns the similarity between two XML trees  $T_1$  and  $T_2$ ,  $sf(T_1, T_2)$  as a real value in the interval  $[0, 1]$ . Finally let  $\tau$  be a similarity threshold, also in the interval  $[0, 1]$ . The Tree Similarity Join between the tree collections specified by  $exp_1(\mathcal{D}_1)$  and  $exp_2(\mathcal{D}_2)$ , denoted by  $TSJ(exp_1(\mathcal{D}_1), exp_2(\mathcal{D}_2), sf, \tau)$ , returns all scored tree pairs  $\langle (T_1, T_2), \tau' \rangle$  s.t.  $(T_1, T_2) \in exp_1(\mathcal{D}_1) \times exp_2(\mathcal{D}_2)$  and  $sf(T_1, T_2) = \tau' \geq \tau$ .

Note that we can evaluate TSJ over the same database by specifying  $\mathcal{D}_1 = \mathcal{D}_2$  or over a single XML tree collection by specifying  $exp_1(\mathcal{D}_1) = exp_2(\mathcal{D}_2)$  or simply omitting the second parameter (hence, defining a self-similarity join).

The course of the TSJ evaluation closely follows that of token-based similarity functions shown in Fig. 2. A pre-step consists of accessing and fetching the trees into a memory-resident area, forming the input of TSJ. To this end, we can fully leverage the query processing infrastructure of a host XML DBMS environment to narrow the similarity join processing to the subset of XML documents (or fragments thereof) specified by the query expression. The next steps, **1) Profile Generation**, **2) Weighting**, and **3) Set Similarity** can be independently implemented and evaluated in a pipelined fashion. Profile Generation produces tokens capturing only structure and structure in conjunction with text. As general strategy, text data appearing under a path in  $PC_t$  is converted to a set of  $q$ -grams and appended to the corresponding structural tokens. Note that, because the set  $PC_t$  is obtained by similarity matching between path queries and the elements of  $PC$  (recall Sect. 2.2), this strategy implicitly embeds similarity selection into the join processing. The realization of Weighting is straightforward. For certain weighting schemes, such as IDF, we need the frequency of all tokens in the database. We can store and easily maintain this information in a simple memory-resident token-frequency table. (Assuming four bytes each for the hashed token value and its frequency, 1.3 million tokens would require around 10MB memory space, which is hardly an issue with modern computers.) Set Similarity is implemented by the set similarity join algorithm based on inverted lists presented in [2]. This algorithm requires sorting the tokens of each profile in increasing order of frequency in the data collection as well as sorting the profiles in increasing order of their size. The sorting of tokens is done in step **2)** using the token-frequency table, while we only need an additional sort operator to deliver sorted profiles to step **3)**.

The TSJ evaluation is completely done “on-the-fly”, i.e, we do not rely on any indexing scheme to provide access or maintenance on pre-computed profiles. Besides avoiding the issue of index update in the presence of data updates, steps **2)** and **3)** do not take an exceedingly large fraction of the overall processing time. Finally, we note that on-the-fly evaluation is the only option in virtual data integration scenarios where the integrated data is not materialized.

## 4 Experiments

The objectives of our empirical experiments are to measure accuracy of the similarity functions (i), overall performance of the TSJ operator (ii), relative performance of the TSJ components (iii), their scalability (iv), and to compare the performance of TSJ using different similarity functions.

We used two real-world XML databases on protein sequences: *SwissProt* (<http://us.expasy.org/sprot/>) and *PSD* (<http://pir.georgetown.edu/>). We deleted the root node of each XML dataset to obtain sets of XML documents. The resulting documents are structurally very heterogeneous. On average, *SwissProt* has a larger number of distinct node labels and exhibits larger and wider

trees. We defined the set  $PC_t$  by issuing two path queries for each dataset: `/Ref/Author` and `Org` on *SwissProt* and `organism/formal` and `sequence` on *PSD*. The resulting text data on *PSD* is about 2x larger than on *SwissProt*.

Using these datasets, we derived variations containing fuzzy duplicates by creating exact copies of the original trees and then performing fuzzy transformations, which aimed at simulating typical deviations between fuzzy duplicates appearing in heterogeneous datasets and those resulting from schema evolution, text data entry errors, and the inherent structural heterogeneity that naturally emanates from the XML data model. Transformations on text nodes consist of word swaps and character-level modifications (insertions, deletions, and substitutions); we applied 1–5 such modifications for each dirty copy. Structural transformations consist of node operations (e.g., insertions, deletions, inversions, and relabeling) as well as deletion of entire subtrees and paths. Insertion and deletion operations follow the semantics of the tree edit distance [5], while node inversions switch the position between a node and its parent; relabeling changes the node’s label.

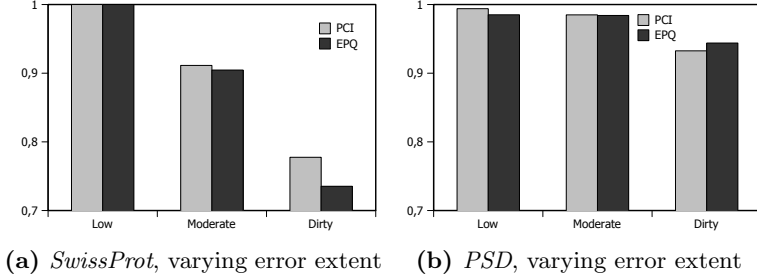
*Error extent* was defined as the percentage of tree nodes which were affected by the set of structural modifications. We considered as affected such nodes receiving modifications (e.g., a rename) and all its descendants. We classify the fuzzy copies generated from each data set according to the error extent used: we have *low* (10%), *moderate* (30%), and *dirty* (50%) error datasets. IDF is used as weighting scheme and WJS as set similarity function. All tests were performed on an Intel Xeon Quad Core 3350 2,66 GHz, about 2.5 GB of main memory. Finally, we conduct our experiments in the context of an XML DBMS platform called *XML Transaction Coordinator* (XTC) [4].

#### 4.1 Accuracy Results

In our first experiment, we evaluated and compared the accuracy of similarity functions for ordered and unordered XML trees. For ordered trees, we employ *epq-grams* [6], an extension of the concept of *pq-grams* [7]. For unordered trees, we can exploit the fact that PCIs are already used to represent similar paths. For this, we simply use the PCIs corresponding to the set of paths of a tree to generate its profile: PCIs of a tree appearing in  $PC_t$  are appended to each *q-gram* generated from the corresponding text node and the remaining PCIs are used to directly represent structural tokens. We did not apply node-swapping operations when generating the dirty datasets; our comparison between similarity functions for unordered and ordered trees is fair. To evaluate accuracy, we used our join algorithms as selection queries, i.e., as the special case where one of the join partners has only one entry. We proceeded as follows. Each dataset was generated by first randomly selecting 500 subtrees from the original dataset and then generating 4500 duplicates from them (i.e., 9 fuzzy copies per subtree, total of 5000 trees). As the query workload, we randomly selected 100 subtrees from the generated dataset. For each queried input subtree  $T$ , the trees  $T_R$  in the result are ranked according to their calculated similarity with  $T$ ; the relevant trees are those generated from the same source tree as  $T$ .

We report the *non-interpolated Average Precision* (AP), which is given by  $AP = \frac{1}{\#relevanttrees} \times \sum_{r=1}^n [P(r) \times rel(r)]$ , where  $r$  is the rank,  $n$  the number





**Fig. 4.** MAP values for different similarity functions on differing datasets

of subtrees returned.  $P(r)$  is the number of *relevant* subtrees ranked before  $r$ , divided by the total number of subtrees ranked before  $r$ , and  $rel(r)$  is 1, if the subtree at rank  $r$  is relevant and 0 otherwise. This measure emphasizes the situation, where more relevant documents are returned earlier. We report the mean of the AP over the query workload (MAP).

Figure 4 shows the results. Our first observation is that both similarity functions obtain near perfect results on low-error datasets. This means that duplicates are properly separated from non-duplicates and positioned on top of the ranked list. Even on dirty datasets, the MAP values are above 0.7 on *SwissProt* and 0.9 on *PSD*. In this connection, we observe that the results on *SwissProt* degrade more than those of *PSD* as the error extent increases. The explanation for this behavior lies on the flip side of structural heterogeneity: while providing good identifying information, structural heterogeneity severely complicates the selection of textual information and, thus, the set  $PC_t$  is more likely to contain spurious PCIs, especially on dirty datasets. Indeed, a closer examination on the dirty dataset of *SwissProt* revealed that  $PC_t$  contained, in fact, several unrelated paths. On the other hand, the results are quite stable on *PSD*, i.e., MAP values do not vary too much on a dataset and no similarity function experienced drastic drop in accuracy along differing datasets. Finally, PCI has overall better accuracy than EPQ (the only exception is on the dirty dataset of *PSD*).

## 4.2 Runtime Performance and Scalability Results

In this experiment, we report the runtime results for fetching the input trees (SCAN), Profile Generation and Weighting steps (collectively reported as SETGEN), set collection sorting (SORT), and set similarity join (JOIN). Note that PCI and EPQ are abbreviated by P and E, respectively. We generated datasets varying from 20k to 100k, in steps of 20k. Finally, we fixed the threshold at 0.75.

The results are shown in Fig. 5. On both datasets, SCAN, SETGEN, and SORT perfectly scale with the input size. Especially for SCAN, this fact indicates that we achieved seamless integration of similarity operators with regular XQuery processing operators of XTC. SCAN is about 80% faster on *PSD* (Fig. 5(b)) as compared to *SwissProt* (Fig. 5(a)), because characteristics of the *PSD* dataset lead to better compression rates of the storage representation. As a result, fewer disk blocks

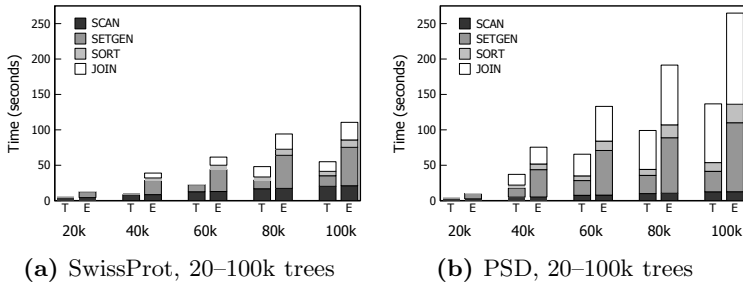


Fig. 5. TSJ execution steps on an increasing number of trees

need to be read during the tree scan operation. On the other hand, SETGEN is about 2x slower on *PSD* as compared to *SwissProt* for both similarity functions. The text data of *PSD* defined by the path queries is larger than those of *SwissProt*, which results in larger sets and, in turn, higher workload for sorting and weighting operations. SETGEN is more than 3x faster on PCI as compared to EPQ. Because paths are provided for free by the path-oriented storage model, PCI-based profile generation simply consists of accessing the PCR-PCI table and splitting strings into sets of  $q$ -grams. On both datasets and for both similarity functions, SORT consumes only a small fraction of the overall processing time. In comparison to the other TSJ components, JOIN takes only up to 20% of the overall processing time on *SwissProt*, whereas it takes up to 60% on *PSD*; on the other hand, JOIN exhibits worst scalability.

## 5 Related Work

There is an extensive research literature on XML similarity, in which a large part is based on the concept of *tree edit distance* [5]. Despite its popularity, the tree edit distance is computationally expensive—worst case complexity of  $O(n^3)$ , where  $n$  is the number of nodes [8]—and, therefore, impractical for use in datasets with potentially large trees. Guha et al. [9] proposed a framework where expensive distance computations are limited by using filters and a pivot-based approach to map trees into a vector space. Augsten et al. presented the concept of  $pq$ -grams to efficiently approximate the tree edit distance on ordered [7] and unordered trees [10]. Neither Guha et al. [9] nor Augsten et al. [7,10] considers textual similarity. Weis and Naumann [11] proposed a framework for XML fuzzy duplicate identification. As in our approach, users can specify the textual information using selection queries. However, the framework only supports structured queries, which prevents its use on heterogeneous datasets.

Dalamagas et al. [12] exploited structural summaries to cluster XML documents by structure. Joshi et al. [13] employed the bag-of-tree-paths model, which represents tree structures by a set of paths. Our aim is completely different from these two previous approaches. We do not cluster XML documents directly; rather, we cluster paths to derive compact structural representations

that can be, afterwards, combined with textual representations to calculate the overall tree similarity. Finally, this paper builds upon our own previous work on similarity for ordered trees [6], unordered trees [3], and set similarity joins [2].

## 6 Conclusion

We presented the design and evaluation of an approach to XML similarity joins. Focusing on token-based similarity functions, we decomposed the original problem into three components suitable for pipelined evaluation, namely, *profile generation*, *weighting*, and *set similarity*. XML tree profiles comprise tokens that capture structural information only and tokens representing structure together with user-defined pieces of textual information. In this context, we identified the need for similarity-based selection of text nodes to evaluate similarity joins on heterogeneous XML datasets — a so far neglected aspect — and proposed an approach that implicitly embeds similarity selection into join processing. We experimentally demonstrated that our approach can be applied flexibly to assess the similarity of ordered and unordered trees, deal with large datasets, and deliver accurate results even when the data quality decreases.

## References

1. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. *TKDE* 19(1), 1–16 (2007)
2. Ribeiro, L.A., Härder, T.: Generalizing prefix filtering to improve set similarity joins. *Information Systems* 36(1), 62–78 (2011)
3. Ribeiro, L.A., Härder, T., Pimenta, F.S.: A cluster-based approach to xml similarity joins. In: *IDEAS*, pp. 182–193 (2009)
4. Mathis, C.: Storing, Indexing, and Querying XML Documents in Native XML Database Systems. PhD thesis, Technische Universität Kaiserslautern (2009)
5. Tai, K.C.: The tree-to-tree correction problem. *Journal of the ACM* 26(3), 422–433 (1979)
6. Ribeiro, L., Härder, T.: Evaluating performance and quality of XML-based similarity joins. In: Atzeni, P., Caplinskas, A., Jaakkola, H. (eds.) *ADBIS 2008*. LNCS, vol. 5207, pp. 246–261. Springer, Heidelberg (2008)
7. Augsten, N., Böhlen, M.H., Gamper, J.: The pq-gram distance between ordered labeled trees. *TODS* 35(1) (2010)
8. Demaine, E.D., Mozes, S., Rossman, B., Weimann, O.: An optimal decomposition algorithm for tree edit distance. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) *ICALP 2007*. LNCS, vol. 4596, pp. 146–157. Springer, Heidelberg (2007)
9. Guha, S., Jagadish, H.V., Koudas, N., Srivastava, D., Yu, T.: Integrating xml data sources using approximate joins. *TODS* 31(1), 161–207 (2006)
10. Augsten, N., Böhlen, M.H., Dyreson, C.E., Gamper, J.: Approximate joins for data-centric xml. In: *ICDE*, pp. 814–823 (2008)
11. Weis, M., Naumann, F.: Dogmatix tracks down duplicates in xml. In: *SIGMOD*, pp. 431–442 (2005)
12. Dalamagas, T., Cheng, T., Winkel, K.J., Sellis, T.K.: A methodology for clustering xml documents by structure. *Information Systems* 31(3), 187–228 (2006)
13. Joshi, S., Agrawal, N., Krishnapuram, R., Negi, S.: A bag of paths model for measuring structural similarity in web documents. In: *SIGKDD*, pp. 577–582 (2003)

# Twig Pattern Matching: A Revisit

Jiang Li<sup>1</sup>, Junhu Wang<sup>1</sup>, and Maolin Huang<sup>2</sup>

<sup>1</sup> School of Information and Communication Technology  
Griffith University, Gold Coast, Australia

Jiang.Li@griffithuni.edu.au, J.Wang@griffith.edu.au

<sup>2</sup> Faculty of Engineering and Information Technology  
The University of Technology, Sydney, Australia  
maolin@it.uts.edu.au

**Abstract.** Twig pattern matching plays a crucial role in XML query processing. In order to reduce the processing time, some existing holistic one-phase twig pattern matching algorithms (e.g., `HolisticTwigStack` [3], `TwigFast` [5], etc) use the core function `getNext` of `TwigStack` [2] to effectively and efficiently filter out the useless elements. However, using `getNext` as a filter may incur other redundant computation. We propose two approaches, namely *re-test checking* and *forward-to-end*, which can avoid the redundant computation and can be easily applied to both holistic one-phase and two-phase algorithms. The experiments show that our approach can significantly improve the efficiency by avoiding the redundant computation.

## 1 Introduction

The importance of fast processing of XML data is well known. *Twig pattern matching*, which is to find all matchings of a query tree pattern in an XML data tree, lies in the center of all XML processing languages. Therefore, finding efficient algorithms for twig pattern matching is an important research problem.

Over the last few years, many algorithms have been proposed to perform twig pattern matching. Bruno et al [2] proposed a two-phase holistic twig join algorithm called `TwigStack`, which breaks the query tree into root-to-leaf paths, finds individual root-to-path solutions, and merges these partial solutions to get the final result. One vivid feature of `TwigStack` is the efficient filtering of useless partial solutions through the use of function `getNext`. Later on several one-phase holistic algorithms (e.g., [3], [5]) also use `getNext` to filter out useless elements. Using `getNext` as a filter can efficiently discard useless elements. However, `getNext` may incur other redundant computation. Li et al [4] try to resolve the redundant computation and propose `TJEssential`, but their approach involves much overhead and can not avoid some important types of redundant computation.

In this paper, we propose a different approach to avoid redundant computation, and this approach imposes less overheads and can be easily applied to both holistic one-phase and two-phase twig pattern matching algorithms that

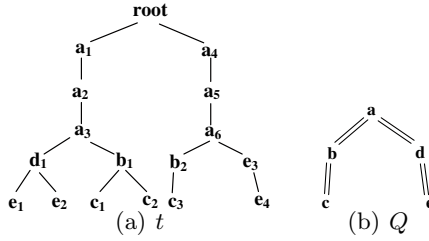


Fig. 1. Example data tree  $t$  and tree pattern  $Q$

are based on `TwigStack`. We present the algorithms `TwigFast*` and `TwigStack*` which extend `TwigFast` and `TwigStack` respectively by applying our proposed approach.

The rest of the paper is organized as follows. Section 2 provides background knowledge and recalls the major features of `getNext` and `TwigFast`. Redundant computation in `getNext` is explained in Section 3. Our approach for resolving redundant computation is presented in Section 4. The experiment results are reported in Section 5. Finally, Section 6 concludes this paper.

## 2 Terminology and Notation

An XML document is modeled as a node-labeled tree, referred to as the *data tree*. A *twig pattern* is also a node-labeled tree, but it has two types of edges:  $/$ -edge and  $//$ -edges, which represent parent-child and ancestor-descendent relationships respectively. The *twig matching* problem is to find all occurrences of the twig pattern in the data tree. For data trees, we adopt the region coding scheme [2]. Each node  $v$  is coded with a tuple of three values:  $(v.start, v.end, v.level)$ .

Below, we will use *elements* to refer to nodes in a data tree, and *nodes* to refer to nodes in a twig pattern. For each node  $n$ , there is a stream,  $T_n$ , consisting of all elements with the same label as  $n$  arranged in ascending order of their *start* values. For each stream  $T_n$ , there exists a pointer  $PT_n$  pointing to the current element in  $T_n$ . The function  $Advance(T_n)$  moves the pointer  $PT_n$  to the next element in  $T_n$ . The function  $getElement(T_n)$  retrieves the current element of  $T_n$ . The function  $isEnd(T_n)$  judges whether  $PT_n$  points to the position after the last element in  $T_n$ . In addition, for node  $n$ , the functions  $isRoot(n)$  (resp.  $isLeaf(n)$ ) checks whether node  $n$  is the root (resp. leaf), and  $parent(n)$  (resp.  $children(n)$ ) returns the parent (resp. set of children) of  $n$ .  $ancestors(n)$  (resp.  $descendants(n)$ ) returns the set of ancestors (resp. set of descendants) of  $n$ .

## 3 Deficiencies in Previous Algorithms

Many previous twig pattern matching algorithms use `getNext` of `TwigStack` to filter out useless elements. However, `getNext` may bring other redundant computation. In this section, we explain where the redundant computation comes

**Table 1.** Example of redundant calls of *getNext*

Step	getNext(a)	getNext(b)	getNext(c)	getNext(d)	getNext(e)
1	a( $a_1$ )	b	c	d	e
2	a( $a_2$ )	b	c	d	e
3	a( $a_3$ )	b	c	d	e
4	d( $d_1$ )	b	c	d	e
5	e( $e_1$ )	b	c	e	e
6	e( $e_2$ )	b	c	e	e
7	e( $e_3$ )	b	c	e	e
8	e( $e_4$ )	b	c	e	e
9	b( $b_1$ )	b	c	d	e
10	c( $c_1$ )	c	c		
11	c( $c_2$ )	c	c		
12	b( $b_2$ )	b	c	d	e
13	c( $c_3$ )	c	c		

from. For ease of understanding, we use the query and data tree in Fig. 1 to exemplify the redundant computation of *getNext*. We present each step of calling *getNext* over the root of the query tree (i.e.,  $a$ ) in Table 1.

Basically, the redundant computation mainly comes from the following *redundant test* and *late end*.

**Redundant test.** *redundant test* is making redundant calls of *getNext* over some nodes in the query tree. The current elements of these nodes did not change in the previous step. Consider the data tree  $t$  and query  $Q$  in Fig. 1.  $a_1$ - $a_3$  are self-nested nodes. After we found  $a_1$  has a solution extension in step 1, it is unnecessary to call *getNext* over the query trees rooted at the nodes  $b$  and  $d$  when testing whether  $a_2$  has a solution extension. This is mainly because the current elements of the nodes  $b$ ,  $c$ ,  $d$  and  $e$  do not change during and after step 1. This also happens on  $a_3$  when checking if  $a_3$  has a solution extension. Therefore, the calls of *getNext* over the nodes  $b$ ,  $c$ ,  $d$  and  $e$  in step 2-3 are redundant, and they are grayed in Table 1. For the similar reason, it is unnecessary to call *getNext* over the nodes  $b$ ,  $c$ ,  $d$  and  $e$  in step 4. This also happens in step 9 and 12, and the calls of *getNext* over  $d$  and  $e$  are redundant. Furthermore, during step 5-7, the calls of *getNext* over the subtree rooted at node  $b$  are redundant.

**Late end.** *late end* is wasting time on the elements that will not contribute to any solutions when some cursors of the streams reach ends. Suppose there are no elements to be processed in the stream of node  $q$ , it is possible to skip all the rest of the elements in the streams of nodes *ancestors*( $q$ ) and *descendants*( $q$ ). This can avoid some calls of *getNext* and the time spent on scanning the elements in some streams. For the example above, when there are no elements left in the stream  $T_d$  after step 4, we can directly set  $PT_a$  to the end because the remaining elements in stream  $T_a$  will not contribute to any solutions. Then, when we found the rest of the elements in the streams of *descendant*( $q$ ) will not contribute any solutions, we can set the  $PT$  pointers of these streams to the ends. In Table 1, calls of *getNext* in step 8 and 13 are redundant and can be pruned.

<sup>1</sup> In this paper, a step is a call of *getNext* over the root of a query tree including the recursive calls.

**Algorithm 1.**  $getNext^*(q)$ 


---

```

1: if  $isLeaf(q)$  then
2:   return  $q$ 
3: for  $q_i \in children(q)$  do
4:   if  $q_i.retest = true$  then
5:      $n_i = getNext^*(q_i)$ 
6:     if  $n_i \neq q_i$  then
7:        $q_i.retest = true$ 
8:       return  $n_i$ 
9:  $n_{min} = \min \arg_{q_i \in children(q)} nextL(T_{q_i})$ 
10:  $n_{max} = \max \arg_{q_i \in children(q)} nextL(T_{q_i})$ 
11: while  $(nextR(T_q) < nextL(T_{n_{max}}))$  do
12:   Advance( $T_q$ )
13: if  $nextL(T_q) < nextL(T_{n_{min}})$  then
14:    $q.retest = false$ 
15:   return  $q$ 
16: else
17:    $n_{min}.retest = true$ 
18:   return  $n_{min}$ 

```

---

## 4 Approach for Avoiding Redundant Computation

### 4.1 Re-test Checking

Our solution for *redundant test* is called *re-test checking* and is mainly based on the following observation:

**Observation.**  $getNext(n)$  is used for testing whether a solution extension can be found for the current element of node  $n$ . Suppose  $getNext$  has been called over the node  $n$  before. If the current element of any node in the query tree rooted at  $n$  changes, it is necessary to call  $getNext(n)$  again for re-testing. Otherwise,  $getNext(n)$  does not need to be called.

We introduce an extra value *retest* for each query node to record whether  $getNext$  need to be called on this node in the next step. The initial value of *retest* is *true*, and this value is dynamic during computation. The new version of  $getNext$  is presented in Algorithm [1](#).

$getNext^*$  has the following properties: (1) Given a query rooted at  $Q$ ,  $getNext^*$  is only called over the nodes whose value of *retest* is *true*, including the nodes that have not been tested by  $getNext^*$  before and the nodes have been tested by  $getNext^*$  but need to be tested again. (2) Suppose  $getNext^*$  has been called over each node at least once. If  $getNext^*(Q)$  returns a node  $n$  in a step,  $getNext^*$  will only be called over the nodes on the path from  $Q$  to  $n$  in the next step. The number of times  $getNext^*$  is called will be bounded by the maximal depth of the query tree in the following steps.

With the properties above, the number of times  $getNext^*$  is called can be significantly reduced, particularly when the query tree has many branches.

### 4.2 Forward-to-End

When the pointer  $PT_n$  of the stream  $T_n$  reaches the end, the rest of the elements in the streams of  $n$ 's ancestors and descendants may become useless. Therefore,

**Algorithm 2.** Forward-to-end

---

```

1: procedure FORWARDANSTOEND( $n$ )
2:   for each  $p$  in  $ancestors(n)$  do
3:      $ForwardtoEnd(T_p)$ 

4: procedure FORWARDDESTOEND( $n$ )
5:   for each  $d$  in  $descendants(n)$  do
6:      $ForwardtoEnd(T_d)$ 

```

---

**Algorithm 3.** TwigFast\*( $Q$ )

---

```

1: initialize the list  $L_{n_i}$  as empty, and set  $n_i.tail = 0$ , for all  $n_i \in Nodes(Q)$ ;
2: while  $\neg end(Q)$  do
3:    $n_{act} = getNext*(root(Q))$ 
4:    $v_{act} = getElement(n_{act})$ 
5:   if  $\neg isRoot(n_{act})$  then
6:      $SetEndPointers(parent(n_{act}), v_{act}.start)$ 
7:   if  $isRoot(n_{act}) \vee parent(n_{act}).tail \neq 0$  then
8:     if  $\neg isLeaf(n_{act})$  then
9:        $SetEndPointers(n_{act}, v_{act}.start)$ 
10:      for  $n_k \in children(n_{act})$  do
11:         $v_{act}.start_{n_k} = length(L_{n_k}) + 1$ 
12:       $v_{act}.ancestor = n_{act}.tail$ 
13:       $n_{act}.tail = length(L_{n_{act}}) + 1$ 
14:      append  $v_{act}$  into list  $L_{n_{act}}$ 
15:    else if  $isEnd(T_{parent(n_{act})}) = true$  then
16:       $ForwardDestoEnd(parent(n_{act}))$ 
17:       $Advance(T_{n_{act}})$ 
18:    if  $isEnd(T_{n_{act}}) = true$  then
19:       $ForwardAnstoEnd(n_{act})$ 
20:  $SetRestEndPointers(Q, \infty)$ 

```

---

we need to find a solution to skip these useless elements. In our approach *forward-to-end*, we select two time points for skipping. Consider a query tree rooted at  $Q$ . Suppose  $getNext(Q)$  returns node  $n$  in a step, and  $PT_n$  reaches the end after calling  $Advance(T_n)$ . The two time points for skipping the rest of the elements in the streams of  $n$ 's ancestors and descendants are as follows:

**Time point 1.** We immediately skip the rest of the elements in the streams of  $n$ 's ancestors after calling  $Advance(T_n)$  because we can not find any solution extension for them in the following steps.

**Time point 2.** We *can not* immediately skip the rest of the elements in the streams of  $n$ 's descendants after calling  $Advance(T_n)$  because they are still potential elements that may contribute to final solutions. We have to wait until there are no elements in the stack  $S_n$  for the two-phase algorithms that use stacks for storing intermediate results and all the end positions in the list  $L_n$  have been set for the one-phase algorithms that use lists for storing final results.

The pseudocode of skipping the rest of the elements in the streams of the node  $n$  and  $n$ 's ancestors and descendants is shown in Algorithm [2](#).



**Algorithm 4.** TwigStack\*(Q)

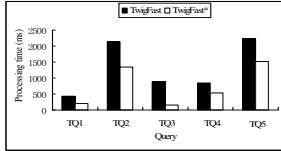
---

```

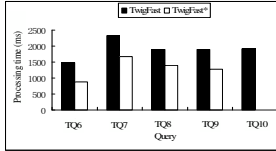
1: while  $\neg \text{end}(Q)$  do
2:    $n_{act} = \text{getNext}^*(\text{root}(Q))$ 
3:    $v_{act} = \text{getElement}(n_{act})$ 
4:   if  $\neg \text{isRoot}(n_{act})$  then
5:      $\text{CleanStack}(\text{parent}(n_{act}), v_{act}.start)$ 
6:   if  $\text{isRoot}(n_{act}) \vee \neg \text{empty}(S_{\text{parent}(n_{act})})$  then
7:      $\text{CleanStack}(n_{act}, v_{act}.start)$ 
8:      $\text{MoveStreamtoStack}(T_{q_{act}}, S_{q_{act}}, \text{pointer to top}(S_{\text{parent}(S_{q_{act}})}))$ 
9:     if  $\neg \text{isLeaf}(n_{act})$  then
10:       $\text{ShowSolutionswithBlocking}(S_{n_{act}}, 1)$ 
11:       $\text{Pop}(S_{n_{act}})$ 
12:   else if  $\text{isEnd}(T_{\text{parent}(n_{act})}) = \text{true}$  then
13:      $\text{ForwardDestoEnd}(\text{parent}(n_{act}))$ 
14:      $\text{Advance}(T_{n_{act}})$ 
15:     if  $\text{isEnd}(T_{n_{act}}) = \text{true}$  then
16:        $\text{ForwardAnstoEnd}(n_{act})$ 
17:  $\text{mergeAllPathSolutions}()$ 

```

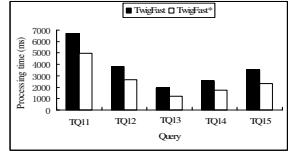
---



(a) low frequency on non-leaf nodes



(b) low frequency on leaf nodes



(c) all nodes with high frequencies

**Fig. 2.** Processing time of queries with different characteristics**4.3 TwigFast\* and TwigStack\***

Algorithm 3 and 4 extend the algorithm TwigFast and TwigStack respectively by applying re-test checking and forward-to-end.

**5 Experiments**

In this section, we present the experiment results on the performance of TwigFast\* against TwigFast [5] and TwigStack\* against TwigStack [2] and TJEssential [4], with both real-world and synthetic data sets. The algorithms are evaluated with the metrics of processing time. We selected the queries with different characteristics for more accurate evaluation.

We implemented TwigFast\*, TwigFast, TwigStack\*, TwigStack and TJEssential in C++. All the experiments were performed on 1.7GHz Intel Pentium M processor with 1G RAM. The operating system is Windows 7. We used the following three data sets for evaluation: TreeBank [1], DBLP [1] and XMark [6]. The queries for evaluation are listed in Table 2, which contain ‘//’ and ‘/’ edges.

**Performance of answering the queries with different characteristics.**In order to make the experiments more objective, we selected the queries with differ-

Table 2. Queries over TreeBank, DBLP and XMark

Data set	Query	XPath expression
TreeBank	TQ1	//V//S
TreeBank	TQ2	//ADV//S//PP//NP
TreeBank	TQ3	//A//S//VP
TreeBank	TQ4	//ADJ//NN//DT
TreeBank	TQ5	//VP//ADV//VP//NP//S
TreeBank	TQ6	//S//NP//CONJ
TreeBank	TQ7	//NP//NP//PP//_NL_
TreeBank	TQ8	//S//VP//NP//_HASH_
TreeBank	TQ9	//S//ADV//PP//NP
TreeBank	TQ10	//VP//NP//PP//FILE
TreeBank	TQ11	//VP//NP//NP//S//PP//VP//NN
TreeBank	TQ12	//NP//S//VP//NP//PP//NP//VBN
TreeBank	TQ13	//S//VP//PP//NP//VBN//IN
TreeBank	TQ14	//S//VP//PP//NP//NP//CD//VBN//IN
TreeBank	TQ15	//S//VP//PP//NP//S//PP//JJ//VBN//PP//NP//_NONE_
TreeBank	TQ16	//S//VP//NP//VP//NP//ADJ//NP//PP//VBN//DT//NN
TreeBank	TQ17	//S//VP//NP//ADV//VBN//VP
TreeBank	TQ18	//S//VP//NP//JJ//NP//PP//ADJ
TreeBank	TQ19	//S//VP//NP//JJ//PP//NN//V
TreeBank	TQ20	//S//VP//NP//PP//A//ADJ
DBLP	DQ1	//dblp/inproceedings/title/author
DBLP	DQ2	//dblp/article/author//title/year
DBLP	DQ3	//dblp/inproceedings/cite//title/author
DBLP	DQ4	//dblp/article/author//title//url//ee/year
DBLP	DQ5	//article//volume//cite//journal
XMark	XQ1	//item/location/description//keyword
XMark	XQ2	//people/person//address/zipcode/profile/education
XMark	XQ3	//item/location//mailbox/mail/emph//description//keyword
XMark	XQ4	//people/person//address/zipcode//id//profile//age//education
XMark	XQ5	//open_auction//annotation//parlist//bidder//increase

ent characteristics over the TreeBank dataset. For the queries TQ1-TQ5, there is at least one non-leaf node with low frequency in each query. On the contrary, the nodes with low frequencies appear on leaf nodes in the queries TQ6-TQ10. For the queries TQ11-TQ15, all the nodes have high frequencies. We compare **TwigFast\*** with **TwigFast** on these three types of queries. The results are shown in Fig. 2. As shown in this figure, **TwigFast\*** achieves better performance than **TwigFast** on all these three types of queries, and is more than 30% faster than **TwigFast** on most queries. The better performance of **TwigFast\*** on the queries TQ1-TQ10 suggests that the *forward to end* approach can avoid the redundant computation brought by *late end*. On the other hand, **TwigFast\*** achieves better performance than **TwigFast** on the queries TQ11-TQ15 mainly because *re-test checking* approach avoids a large amount of unnecessary calls of *getNext*.

**Performance of answering the queries over different datasets.** We first compare **TwigFast\*** with **TwigFast** over the datasets TreeBank, DBLP and XMark. The queries TQ16-TQ20 over TreeBank dataset mix different characteristics we mentioned above. The results are shown in Fig. 3. As shown in this figure, **TwigFast\*** has better efficiency than **TwigFast** on all of the queries over the three datasets. Then we compare **TwigStack\*** with **TwigStack** and **TJEssential** over the datasets TreeBank, DBLP and XMark. The results are shown in Fig. 4.

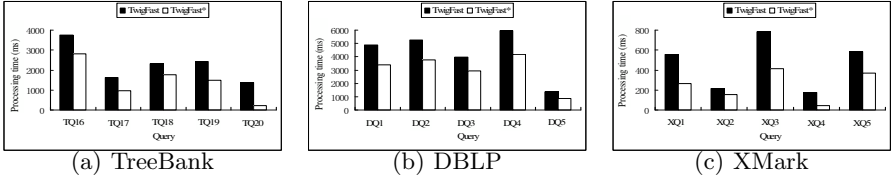


Fig. 3. TwigFast vs TwigFast\*

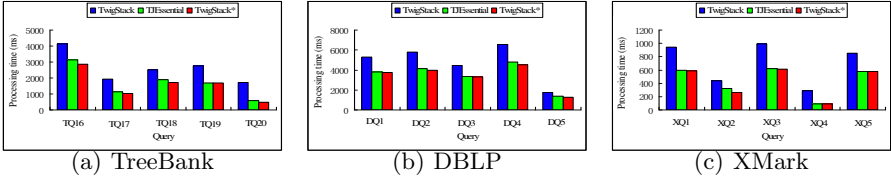


Fig. 4. TwigStack, TJEssential vs TwigStack\*

From the results, we can see that both TJEssential and TwigStack\* achieves better performance than TwigStack by resolving the redundant computation even though they use different approaches. Additionally, TwigStack\* is a littler faster than TJEssential because TwigStack\* can avoid some redundant computation that TJEssential can not avoid and TwigStack\* imposes less overheads.

## 6 Conclusion

We presented the approaches *re-test checking* and *forward-to-end*, which can be easily applied to both holistic one-phase and two-phase twig pattern matching algorithms that are based on TwigStack, to resolve the redundant computation in *getNext*. Two algorithms TwigFast\* and TwigStack\* were presented. The better performance of our algorithms has been verified in our experiments.

## References

1. <http://www.cs.washington.edu/research/xmldatasets/>
2. Bruno, N., Koudas, N., Srivastava, D.: Holistic twig joins: optimal XML pattern matching. In: SIGMOD Conference, pp. 310–321 (2002)
3. Jiang, Z., Luo, C., Hou, W.-C., Zhu, Q., Che, D.: Efficient processing of XML twig pattern: A novel one-phase holistic solution. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 87–97. Springer, Heidelberg (2007)
4. Li, G., Feng, J., Zhang, Y., Zhou, L.: Efficient holistic twig joins in leaf-to-root combining with root-to-leaf way. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 834–849. Springer, Heidelberg (2007)
5. Li, J., Wang, J.: Fast matching of twig patterns. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 523–536. Springer, Heidelberg (2008)
6. Schmidt, A., Waas, F., Kersten, M., Florescu, D., Manolescu, I., Carey, M., Busse, R.: The XML benchmark project. Technical Report INS-R0103, CWI (April 2001)

# Boosting Twig Joins in Probabilistic XML

Siqi Liu and Guoren Wang

School of Information Science and Engineering, Northeastern University, China

**Abstract.** In practice, uncertainty of data is inherent. Probabilistic XML has been proposed to manage semistructured uncertain data. In this paper, we study twig query evaluation over probabilistic XML with probability thresholds. First we propose an encoding scheme for probabilistic XML. Then we design a novel streaming scheme which enables us to prune off useless inputs. Based on the encoding scheme and streaming scheme, we develop an algorithm to evaluate twig queries over probabilistic XML. Finally, we conduct experiments to study the performance of our algorithm.

## 1 Introduction

Data are inherently uncertain in many applications. The semistructured XML model is more flexible and natural to represent uncertain data compared with the relational model. Various probabilistic XML models have been proposed and studied [1–5]. Among the studied models, there is a tradeoff between the expressiveness and the efficiency of query evaluation [2].

Evaluating twig queries over probabilistic XML, we need not only the query answers but also the probability of each answer. Since we are not interested in the answers with low probabilities in most situations, a probability threshold is usually given, requiring only returning the answers satisfying it. In [2, 6], Kimelfeld et al. defined several semantics of twig queries and studied the evaluation of twig queries with projection. They focused on the evaluation of probabilities and left the work of finding the matches to traditional algorithms, which are not efficient enough. Therefore, algorithms specifically designed for twig query evaluation over probabilistic XML have been proposed [7, 8].

In this paper, we focus on the problem of evaluating twig queries with probability thresholds over probabilistic XML (p-documents). The data model we adopt is similar to  $PrXML^{\{ind, mux\}}$  [4]. For p-documents, we propose an encoding scheme, pDewey, in Sect. 3, and a streaming scheme, *Tag+Probability*, in Sect. 4. In Sect. 5 we develop an algorithm, pTJFastTP, for twig query evaluation over p-documents with probability thresholds. In Sect. 6 we conduct experiments to compare our algorithm with the algorithm in [8].

## 2 Preliminaries

### 2.1 Data Model

An XML document is a tree  $d$ .  $V(d)$  denotes the set of nodes.  $E(d)$  denotes the set of edges. Figure 1(b) is an XML document. Additional numbers are used to

distinguish different nodes with the same tag. A p-document is a weighted tree  $P$  with ordinary nodes (o-nodes) and distributional nodes (d-nodes). An o-node is same as a node in XML documents. A d-node defines a random selection over its children. The weight of an edge  $e$ , denoted as  $Pr(e)$ , specifies the probability that the parent selects the child. If  $e$  emanates from a d-node, it is a positive real in  $[0, 1]$ ; otherwise, it is 1. Figure 1(a) is a p-document.

A p-document generates an XML document randomly by: First, each d-node randomly selects its children. The selections of an *ind* (*mux*) node are independent (mutually exclusive). Second, all d-nodes are removed. If the parent of any node is removed, it becomes the child of its lowest ancestor.  $D$  denotes the set of all possible documents. For each  $d \in D$ ,  $Pr(d)$  denotes the probability that  $P$  generates  $d$ .  $Pr(d) = \prod_{e \in V(d)} Pr(e)$ . For each o-node  $v \in V(P)$ ,  $Pr(v)$  denotes the probability that  $v$  appears in a random document. It equals the product of  $Pr(e)$  for all edges  $e$  in the path from the root to  $v$ .

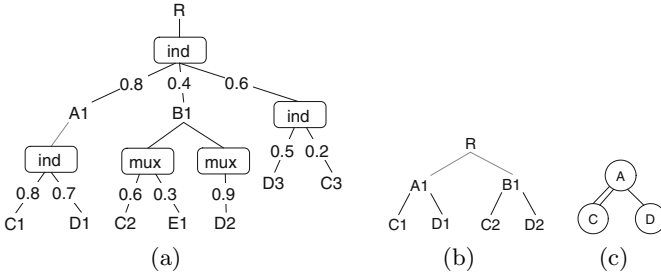


Fig. 1. Data model samples: (a) p-document, (b) XML document and (c) twig pattern

### 2.2 Twig Patterns and Answers

A twig pattern  $Q$  is a tree with parent-child edges and ancestor-descendant edges. A predicate  $pred_n(\cdot)$  is associated with each node  $n$ . A match of  $Q$  in an XML document  $d$  is a mapping  $\phi : V(Q) \rightarrow V(d)$  such that (i) for all  $n \in V(Q)$ ,  $pred_n(\phi(n))$  is true, (ii) the relationship on the edge between any two nodes  $n, m \in Q$  is satisfied by  $\phi(n), \phi(m)$ . The image of  $\phi$  is called an answer.  $Q(d)$  denotes the set of all answers. Figure 1(c) is a twig pattern. A double line represents a descendant edge. The predicates are matching of tags.

For a p-document  $P$ ,  $Q(P) = \bigcup_{d \in D} Q(d)$ . For an answer  $t \in Q(P)$ ,  $Pr(t)$  denotes the probability that  $t$  appears in a random document generated by  $P$ .  $Pr(t) = \prod_{e \in E(r)} Pr(e)$  where  $r$  is the minimal subtree of  $P$  that has the same root of  $P$  and contains all the nodes in  $t$ . We define  $Pr(e_1) \cdot Pr(e_2) = 0$  where  $e_1$  and  $e_2$  emanate from the same *mux* node and  $e_1 \neq e_2$ . In this paper, we are only interested in the answers of which the probabilities are above a given threshold  $PT$ .

## 3 Encoding Scheme: pDewey

Inspired by PEDewey [9] we propose a new encoding scheme pDewey based on extended Dewey [10]. Two types of tag sets are necessary for pDewey: (i) For

each tag  $t$  in a p-document,  $CT(t)$  contains the tags of all the children of nodes with tag  $t$ . (ii)  $DT = \{ind, mux\}$  contains all tags of d-nodes. In the process of getting  $CT(t)$ , we ignore all d-nodes. A pDewey label is composed of a Dewey label (DLabel) and a probabilistic label (PLabel). A DLabel is an integer vector. For each node  $v$ ,  $DLabel(v) = DLabel(p).x$  where  $p$  is the parent of  $v$ . According to the type of  $v$  we assign the value of  $x$  as follows.

1. For an o-node, suppose that  $a$  is the lowest ordinary ancestor of  $v$ ,  $tag(v)$  is the  $k$ -th tag in  $CT(tag(a))$  and  $|CT(tag(a))| = l$ . Ignoring d-nodes,
  - (a) if  $v$  is the first ordinary child of  $a$ , then  $x = k$ ;
  - (b) otherwise, suppose that the last integer assigned to the left ordinary sibling of  $v$  is  $y$ , then

$$x = \begin{cases} \lfloor \frac{y}{l} \rfloor \cdot l + k & \text{if } (y \bmod l) < k, \\ (\lfloor \frac{y}{l} \rfloor + 1) \cdot l + k & \text{otherwise.} \end{cases}$$

2. For a d-node, suppose that  $tag(v)$  is the  $i$ -th tag in  $DT$  and  $|DT| = m$ ,
  - (a) if  $v$  is the first distributional child of  $p$ , then  $x = -m$  if  $i=0$ , otherwise  $x = -i$ .
  - (b) otherwise, suppose that the last integer assigned to the left distributional sibling of  $v$  is  $y$ , then

$$x = \begin{cases} \lfloor \frac{y}{m} \rfloor \cdot m - i & \text{if } ((-y) \bmod m) < i, \\ (\lfloor \frac{y}{m} \rfloor - 1) \cdot m - i & \text{otherwise.} \end{cases}$$

For each node, we assign a PLabel along with the DLabel. The PLabel is a float vector. For each node  $v$ ,  $PLabel(v) = PLabel(p).x$  where  $p$  is the parent of  $v$ . Suppose that the last component of  $PLabel(p)$  is  $y$ , then  $x = y \cdot Pr(e(p, v))$ . If  $v$  is an o-node, we can infer that  $x = Pr(v)$ . Figure 2 shows the pDewey labels of the nodes in the p-document in Fig. 1(a).

From the pDewey label of each node, we can get the labels, tags and probabilities of all its ancestors. For two nodes  $a$  and  $b$ ,  $a$  is the ancestor of  $b$  if  $DLabel(a)$  is a prefix of  $DLabel(b)$ . Additionally if ignoring all negative integers  $DLabel(a).length = DLabel(b).length - 1$ , then  $a$  is the parent of  $b$ .

## 4 Streaming Scheme

In our algorithm, the inputs are streams consisting of the nodes satisfying the predicates on the leaves of the twig. In previous algorithms, nodes with the same tag are partitioned into the same stream. We propose a novel streaming scheme *Tag + Probability* for p-documents using both tags and probabilities of nodes to partition them. First, we introduce probability evaluation as preliminaries.

## 4.1 Probability Evaluation

Evaluating twig queries in p-documents, we must return the probability along with each answer. With pDewey encoding it is possible to evaluate the probability of an answer only from its leaves.

For an answer  $t = (n_1, n_2, \dots, n_m)$ , suppose that  $f_1, f_2, \dots, f_l$  are all the leaves in  $t$ . Since the existence of a leaf guarantees the existence of all its ancestors,

$$Pr(t) = Pr(\bigwedge_{i=1}^m n_i) = Pr(\bigwedge_{j=1}^l f_j) = Pr(f_1) \cdot \prod_{j=2}^l Pr(f_j | \bigwedge_{k=1}^{j-1} f_k) \quad (1)$$

## 4.2 Tag + Probability Streaming Scheme

In ordinary tag stream, each input stream consists of all nodes of the same tag sorted in ascending lexicographical order. *Tag + Probability* streaming scheme partitions every tag stream into several parts according to the probabilities of the nodes in the stream.

First, we partition the range of probability,  $[0, 1]$ , into  $k$  intervals,  $r_1, r_2, \dots, r_k$ , where  $r_i$  is  $[lower_i, upper_i)$  ( $r_k$  is  $[lower_k, 1]$ ). Then we partition each tag stream  $t$  into  $k$  streams,  $t^1, t^2, \dots, t^k$ , such that for each  $t^i$ ,  $\forall v \in t^i, Pr(v) \in r_i$  and all nodes preserve their order. The streams can be generated directly from the p-document without the tag streams.

## 4.3 Pruning Streams

Using *Tag + Probability* streaming scheme, we associate each leaf  $f$  of twig with a set of streams  $T_f$  which contains  $k$  streams  $t_f^1, t_f^2, \dots, t_f^k$  satisfying the predicate of  $f$ . At the beginning of query evaluation, we prune off streams by: (i) Find  $i$  such that  $PT \in r_i$ . (ii) Remove all  $t_f^j$  such that  $j < i$ . The correctness follows from (II). We use a simple example to show the benefits of *Tag + Probability* streaming scheme and the pruning technique.

*Example 1.* The top of Fig. 3 shows a tag stream of A. The probability is below each node. In the bottom,  $[0, 1]$  is partitioned into four equal-sized intervals. The original stream is partitioned into corresponding four streams. Each node is selected into the stream such that its probability is in the corresponding interval. Note that their order is preserved. When evaluating a query with a threshold 0.5 we prune off the streams below the threshold, which in the figure is the two streams below the dotted line.

## 5 Algorithm pTJFastTP

In this section, we describe the algorithm pTJFastTP which evaluates twig queries with probability thresholds over p-documents. First, we introduce some notations and operations.

Node	DLabel	PLabel
R	(0)	(1)
A1	(0),(-1),(0)	(1),(1),(0.8)
C1	(0),(-1),(0),(-1),(0)	(1),(1),(0.8),(0.8),(0.64)
D1	(0),(-1),(0),(-1),(1)	(1),(1),(0.8),(0.8),(0.56)
B1	(0),(-1),(1)	(1),(1),(0.4)
C2	(0),(-1),(1),(-2),(0)	(1),(1),(0.4),(0.4),(0.24)
E1	(0),(-1),(1),(-2),(2)	(1),(1),(0.4),(0.4),(0.12)
D2	(0),(-1),(1),(-4),(4)	(1),(1),(0.4),(0.4),(0.36)
D3	(0),(-1),(-1),(3)	(1),(1),(0.6),(0.3)
C3	(0),(-1),(-1),(6)	(1),(1),(0.6),(0.12)

Fig. 2. pDewey encoding

A1	A2	A3	A4	A5	A6	A7	A8
0.58	0.46	0.90	0.76	0.25	0.12	0.66	0.33
$r_4$ [0.75,1.00]				A3 A4			
$r_3$ [0.50,0.75]				A1 A7			
----- PT=0.50 -----							
$r_2$ [0.25,0.50]				A2 A5 A8			
$r_1$ [0.00,0.25]				A6			

Fig. 3. Tag + Probability streaming scheme and pruning technique

## 5.1 Notations and Operations

For a twig pattern  $q$ ,  $leaves(q)$  is the set of leaves. With each leaf  $f$ , we associate a stream set  $T_f$  consisting of the streams satisfying the predicate of  $f$ . The following operations are defined on  $T_f$ :  $head(T_f)$  returns the node with the minimal encoding in the heads of all the streams.  $min(T_f)$  returns the stream containing  $head(T_f)$ .  $advance(T_f)$  advances  $min(T_f)$  till its head matches  $p_f$ .  $eof(T_f)$  checks if all the streams end. Each branch node  $b$  is associated with a set  $S_b$  containing the roots of answers to the sub-twig rooted at  $b$ . For each leaf  $f$  in twig patterns and p-documents,  $p_f$  denotes the path from the root to  $f$ .

## 5.2 pTJFastTP

Algorithm 1 evaluates a twig pattern  $q$  with a probability threshold  $PT$  and outputs answers that match  $q$  and satisfy  $PT$ . The algorithm consists of two phases. In the first phase, it outputs solutions that match the path patterns. In the other phase, all the path solutions are joined into complete answers and the probabilities are evaluated. We describe the details of this phase in the next subsection.

At the beginning, pTJFastTP prunes off the streams that cannot satisfy  $PT$ . In the main loop (line 2 to 6), it iteratively get the next node to process and output the path solutions derived from it according to the associated sets of branch nodes. When all streams end, it joins the path solutions and evaluates the probabilities.

Function  $getNext(q)$  traverses the twig  $q$  top-down recursively. For each branch node  $b$ , it identifies for the sub-twig rooted at  $b$  all the answers that can be formed by  $p_{head(T_f)}$  for all  $f \in leaves(b)$ . Then, it updates  $S_b$  with the nodes matching  $b$  in the answers. Finally, it returns the leaf  $f$  of  $b$  such that  $head(T_f)$  is minimal for all  $f \in leaves(b)$ .

## 5.3 Pruning in Merge-Join

In merge-join, we incrementally build answers by joining the existing partial answers. Suppose that two existing partial answers are  $t_1$  and  $t_2$ , which can be



**Algorithm 1.** pTJFastTP(q,PT)

---

```

1: Prune streams in  $T_f$  for each  $f \in leaves(q)$ 
2: while  $\exists f \in leaves(q) : \neg eof(T_f)$  do
3:    $n \leftarrow getNext(q)$ 
4:   Output path solutions  $\{s | s \subseteq p_{head(T_n)} \wedge s \text{ matches } p_n \wedge \forall e \in s, e \text{ matches a branch node } b \rightarrow e \in S_b\}$ 
5:    $advance(T_n)$ 
6: end while
7: Merge-join all the path solutions and evaluate the probabilities

```

---

joined to  $t_3$ . For probability evaluation, we need join them as in the p-document instead of as in the twig. That is, we iteratively join each path in  $t_2$  to  $t_1$ . We join a path  $p_f$  with a tree (partial answer)  $t$  by: First, find out the join point  $NLCA(f, t)$ , which is defined as the nearest one to  $f$  in  $\{LCA(f, f') | f' \in leaves(t)\}$  ( $LCA(f, f')$  is the lowest common ancestor of  $f$  and  $f'$ ). Then, compute the probability of the result  $t'$  as:  $Pr(t') = Pr(t) \cdot Pr(f) / Pr(NLCA(f, t))$  except that if  $NLCA(f, t)$  is of type *mux* it is 0.

We propose a pruning strategy in merge-join. We maintain  $Pr_{min}(t) = \min\{Pr(f) | f \in leaves(t)\}$  for each tree  $t$ . Before joining  $t$  and  $p_f$ , we check  $Pr(f) \cdot Pr(t) / Pr_{min}(t) < PT$ . If it is true, we are guaranteed that  $Pr(t') < PT$  (because  $Pr(NLCA(f, t)) > Pr_{min}(t)$ ) and thus abandon this join. Additionally, after the computation of  $Pr(t')$  we discard  $t'$  if  $Pr(t')$  cannot satisfy  $PT$ .

## 6 Experimental Evaluation

*Experimental setup.* We implemented pTJFastTP and ProTwig in JDK 1.6. All experiments were running on a PC with 3.20GHz Intel Core i3 CPU and 3.18GB RAM running Windows 7. We allocated 1GB of memory for the Java virtual machine.

We implemented an algorithm to randomly convert an XML document into a p-document. It traverses the original document in preorder and randomly inserts d-nodes between a node and its children. We used both real-world (DBLP) and synthetic (XMark) datasets for our experiments.

In the following experiments, pTJFastTP adopts a default *Tag + Probability* stream scheme partitioning  $[0, 1]$  into ten equal-sized intervals and each tag stream into ten corresponding streams. The default probability threshold is 0.5. For lack of space we only show some representative experiments.

*Varying threshold.* We compare their performance over DBLP with different thresholds. From Fig. 4 we see that, pTJFastTP always outperforms ProTwig, and the gap between the two algorithms becomes wider with the increase of the probability threshold. The reason is that with a greater probability threshold pTJFastTP prunes off more inputs while ProTwig has to read them all.

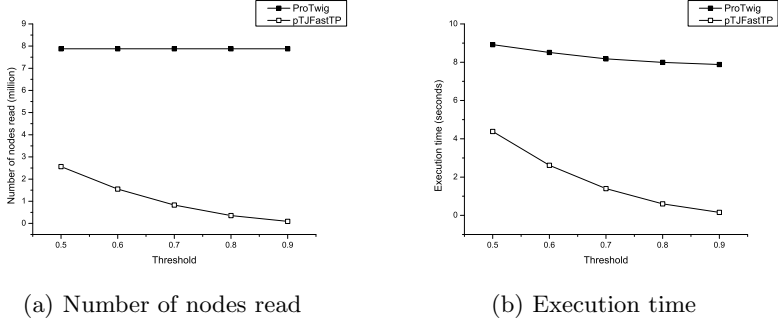


Fig. 4. Vary probability threshold

*Scalability.* We compare the scalability of the two algorithms. We use seven different sizes of data generated by XMark with varying factors from 1 to 7. From Fig. 5 we see that pTJFastTP is far more scalable than ProTwig. The reason is that though the size of total inputs increases, the size of useless inputs pruned off by pTJFastTP also increases. So the I/O cost remains low.

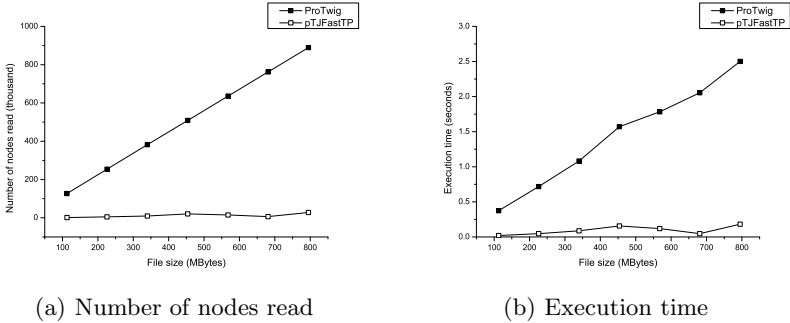


Fig. 5. Vary data size

## 7 Related Work

For evaluating twig queries over probabilistic XML, Kimelfeld et al. developed an algorithm, EvalDP, in [6], which was optimized and extended in [2]. Their work focused on the probability evaluation. In [7] Li et al. proposed a holistic method based on TwigStack [11]. In [8] Ning et al. proposed ProTwig based on TJFast [10]. These two algorithms focused on the query evaluation.

The extended Dewey encoding scheme was proposed in [10] for ordinary XML. In [9], Ning et al. extended it to PEDewey for p-documents. Our encoding scheme, pDewey, has two main differences from PEDewey: (i) In PEDewey, negative constant is used to represent d-nodes, which lead to that the different d-nodes having the same parent cannot be distinguished. So it cannot join two path correctly. For example, it cannot join  $p_{C1}$  and  $p_{D2}$  in the p-document in

Fig. 1(a), because it cannot distinguish the *mux* nodes in the two paths. Whereas pDewey does not have this flaw. (ii) In PEDewey, getting the probability of a node costs  $O(d)$  where  $d$  is the depth of the p-document. Whereas in pDewey, it costs  $O(1)$ .

The concept of streaming scheme was proposed in [12] to boost holism in TwigStack. They developed two streaming schemes for ordinary XML.

## 8 Conclusion

In this paper, we propose an algorithm pTJFastTP to evaluate twig queries with probability thresholds over p-documents. It adopts the encoding scheme pDewey, which incorporates information of distributional nodes and probabilities into extended Dewey. Thus we can efficiently get the tags and probabilities of the ancestors of any node. We propose *Tag + Probability* streaming scheme, with which pTJFastTP can prune off useless inputs to reduce I/O cost. In merge-join, a pruning strategy is proposed to prune off useless partial answers. The results of experiments show the advantages of pTJFastTP in various aspects.

## References

1. Abiteboul, S., Kimelfeld, B., Sagiv, Y., Senellart, P.: On the expressiveness of probabilistic XML models. *VLDB Journal* 18(5), 1041–1064 (2009)
2. Kimelfeld, B., Kosharovskiy, Y., Sagiv, Y.: Query evaluation over probabilistic XML. *VLDB Journal* 18(5), 1117–1140 (2009)
3. Kimelfeld, B., Kosharovskiy, Y., Sagiv, Y.: Query efficiency in probabilistic XML models. In: *SIGMOD 2008*, pp. 701–714. ACM, Canada (2008)
4. Kimelfeld, B., Sagiv, Y.: Modeling and querying probabilistic XML data. *SIGMOD Record* 37(4), 69–77 (2008)
5. Senellart, P., Abiteboul, S.: On the complexity of managing probabilistic XML data. In: *PODS 2007*, pp. 283–292. ACM, China (2007)
6. Kimelfeld, B., Sagiv, Y.: Matching twigs in probabilistic XML. In: *VLDB 2007*, pp. 27–38. VLDB Endowment (2007)
7. Li, Y., Wang, G., Xin, J., Zhang, E., Qiu, Z.: Holistically twig matching in probabilistic XML. In: *ICDE 2009*, pp. 1649–1656. Inst. of Elec. and Elec. Eng. Computer Society, China (2009)
8. Ning, B., Li, G., Zhou, X., Zhao, Y.: An efficient algorithm for twig joins in probabilistic XML. In: *PIC 2010*, vol. 1, pp. 622–626. IEEE Computer Society, China (2010)
9. Ning, B., Liu, C., Yu, J.X., Wang, G., Li, J.: Matching top-k answers of twig patterns in probabilistic XML. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) *DASFAA 2010*. LNCS, vol. 5981, pp. 125–139. Springer, Heidelberg (2010)
10. Lu, J., Ling, T.W., Chan, C.Y., Chen, T.: From region encoding to extended dewey: On efficient processing of XML twig pattern matching. In: *VLDB 2005*, vol. 1, pp. 193–204. ACM, Norway (2005)
11. Bruno, N., Koudas, N., Srivastava, D.: Holistic twig joins: Optimal XML pattern matching. In: *SIGMOD 2002*, pp. 310–321. ACM, United states (2002)
12. Chen, T., Lu, J., Ling, T.: On boosting holism in XML twig pattern matching using structural indexing techniques. In: *SIGMOD 2005*, pp. 455–466. ACM, New York (2005)

# Prediction of Cerebral Aneurysm Rupture Using Hemodynamic, Morphologic and Clinical Features: A Data Mining Approach

Jesus Bisbal<sup>1,2</sup>, Gerhard Engelbrecht<sup>1,2</sup>, Mari-Cruz Villa-Uriol<sup>1,2</sup>,  
and Alejandro F. Frangi<sup>1,2,3</sup>

<sup>1</sup> Center for Computational Imaging and Simulation Technologies in Biomedicine (CISTIB), Universitat Pompeu Fabra (UPF), Barcelona, Spain

<sup>2</sup> Networking Biomedical Research Center on Bioengineering, Biomaterials and Nanomedicine (CIBER-BBN)

<sup>3</sup> Institució Catalana de Recerca i Estudis Avançats (ICREA), Barcelona, Spain  
`jesus.bisbal@upf.edu`

**Abstract.** Cerebral aneurysms pose a major clinical threat and the current practice upon diagnosis is a complex, lengthy, and costly, multi-criteria analysis, which to date is not fully understood. This paper reports the development of several classifiers predicting whether a given clinical case is likely to rupture taking into account available information of the patient and characteristics of the aneurysm.

The dataset used included 157 cases, with 294 features each. The broad range of features include basic demographics and clinical information, morphological characteristics computed from the patient's medical images, as well as results gained from personalised blood flow simulations.

In this premiere attempt the wealth of aneurysm-related information gained from multiple heterogeneous sources and complex simulation processes is used to systematically apply different data-mining algorithms and assess their predictive accuracy in this domain. The promising results show up to 95% classification accuracy. Moreover, the analysis also enables to confirm or reject risk factors commonly accepted or suspected in the domain.

**Keywords:** Data mining, complex data, classifiers, association rules, feature discretization, feature selection, decision support, aneurysm rupture, biomedicine.

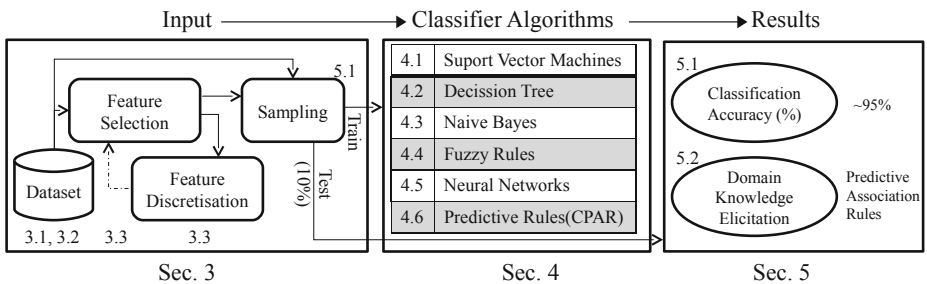
## 1 Introduction

An aneurysm is a localized, blood-filled dilation of a blood vessel caused by disease or weakening of the vessel wall. If left untreated, it can burst leading to severe haemorrhage and sudden death [2]. The current practice involves a complex and multi-criteria analysis, which aims to assess the risk of undertaking no action confronted with the inherent risk of performing an invasive treatment. However, the complexity of the potentially influencing factors leading to rupture

of a cerebrovascular aneurysm is, to a large extent, still unknown and consequently preventive treatment is offered to almost all patients. Moreover, despite the misgivings of the clinical practitioners, there is still insufficient evidence to support a non-interventional treatment. Also the economic dimension of unnecessary interventional or surgical procedures is significant with estimated costs of several hundred million Euros per annum in Europe alone [12,20].

This paper aims to contribute towards a more complex understanding of the associations among the potential risk factors of a cerebral aneurysm. The approach aims to exploit the wealth of aneurysm-related information gained from multiple heterogeneous sources and complex simulation processes by systematically applying different state-of-the-art data mining techniques. This resulted in several classifiers predicting whether a given clinical case is likely to rupture, with classification accuracy as high as 95%. The broad range of features considered in the prediction include basic demographics and clinical information of the patient, morphological characteristics about the aneurysm computed from the patient’s medical images, as well as results gained from personalised blood flow simulations. Especially morphological, flow, and structural characteristics are reported to be different for ruptured and unruptured aneurysms [24,26].

The basic objective of this paper is to provide decision support predicting whether a given clinical case is likely to rupture or not. Therefore, different prediction models, by means of six different classifier algorithms, are utilized in order to automatically differentiate between ruptured and unruptured cases. Figure 1 presents an overview of the prediction process including the input processing, the used classifier algorithms and the obtained results. The extracted set of strong predictive rules which, according to our dataset, associates specific feature values with aneurysm rupture status contributes to underpin clinical assumptions and to the accepted knowledge of relationships between certain characteristics of the patient and/or aneurysm, which to date are not fully understood.



**Fig. 1.** Flowchart summarising the paper organisation and the prediction process

The dataset used as a starting point for the prediction process was taken from the @neurIST project [9]. It contains a large amount of potential features to be used, but the actually considered features have been reduced to those more likely to be relevant to differentiate between ruptured and unruptured cases. To the best of the authors’ knowledge, this is the first attempt to systematically

apply these data mining algorithms to this specific and complex clinical domain, including such heterogeneous multimodal data. Conclusions are also drawn from a population significantly larger than used in previous studies in this application domain.

The remainder of this paper is organized as follows: Section 2 summarizes related work. Sections 3 to 5 follow the process of the prediction as shown in Figure 1, where the numbers indicate the sections and subsections. Finally, the main conclusions are summarised in Section 6, and Section 7 presents a number of future directions for this research.

## 2 Related Work

The application of data mining techniques for decision support in the clinical domain is receiving increased scientific attention with the general objective to predict severe conditions (e.g. cancer, heart failure or aneurysm rupture) from the vast and growing amounts of data generated in the biomedical domain. Data mining techniques have clearly been applied more extensively to specific biomedical domains, such as breast cancer [1,19] and cardiology [14,21].

**Table 1.** Summary of previous contributions related to the objectives of this paper

Ref.	Domain	Feature Types	Techniques Used	Results
[1]	Breast Cancer	Image mean, variance, skewness and kurtosis	Association Rules Neural Networks	Accuracy 69.11% Inconsistent, 89.2%
[19]	Breast Cancer	as [1], entropy, invariant moments	Association Rules	Accuracy 96.7%
[14]	Coronary Heart Disease	Thickness carotid artery, ECG calc.	SVM, CPAR, Bayes, C4.5, MDA	Accuracy 90% SVM and CPAR
[13]	Coronary Calcium Detection	Location, texture-based features	SVM hierarchical multiple ensemble	Accuracy 92.6%
[23]	Cerebral aneurysms	Only morphological	Linear discriminant analysis (LDA)	Accuracy 77.9%
[6,7]	Cerebral Aneurysms	Only hemodynamics		Good discriminants
[26]	Cerebral Aneurysms	Morphological and hemodynamics	Multivariate regression analysis	No classifiers Good discriminants
[8,10]	Cerebral Aneurysms	Clinical, literature, genetics	Decision tree, text mining	Association studies personal assessment

Related prediction approaches and association studies of cerebral aneurysm rupture are typically based on other techniques than data mining and utilize very specific data about the hemodynamics [7,6] and/or morphology of an aneurysm [23,26]. Data mining algorithms in the context of cerebral aneurysms have been applied by the application suites of the @neurIST project, i.e. @neuLink [10] to link clinical and genetic information to public literature and use the linked knowledge for an individual risk assessment in @neuRisk [8].

Table 1 summarises the related work by showing the specific biomedical application domain, the features considered for the prediction or analysis, the techniques used and the most relevant results achieved. The approach presented

in this paper goes beyond previous works, by applying different data-mining technologies in the cerebral aneurysm domain and by considering a larger and broader set of descriptors (i.e. clinical information along with computed hemodynamic and morphologic data).

### 3 Data Collection and Pre-processing

A central objective of this work was to automatically categorise cerebral aneurysms according to their rupture status using different state-of-the-art data mining techniques. In line with the processing flow shown in Figure 1, classification models have been created and applied with a selected set of features and then the accuracy of the classifiers has been tested with new aneurysm instances. The potential set of features is extremely large and inherently multimodal, since they are derived from heterogeneous sources:

**Clinical Information.** Data traditionally used for clinical care, such as demographics (e.g. age, gender), lifestyle (e.g. smoker status), and aneurysm anatomical location and type.

**Morphological Analysis.** Image processing techniques are applied to obtain morphological descriptors about an aneurysm, such as aspect ratio, non-sphericity index, volume, or Zernike moment invariants [17].

**Hemodynamic Analysis.** Blood flow simulations using computational fluid dynamics are used to compute characteristics of the blood flow and pressure, as well as their impact on the vessel wall.

The heterogeneity of the considered data underpins the complexity involved in discovering correlations and dependencies among seemingly unrelated factors. This paper represents an initial attempt to systematically exploit the data through data mining techniques.

#### 3.1 Experimental Dataset

The dataset used for experimental evaluation of this work originated in the @neurIST project focused on advancing the management of cerebral aneurysms using information technology. All the clinical information relevant for this purpose was captured in a large virtual patient information system federated across Europe. The data representation followed the Clinical Reference Information Model (CRIM) [11] and was captured in the clinical context either by a research personnel or drawn directly from hospital electronic records. The CRIM exposed up to 1286 clinical attributes for each subject including basic demographics and lifestyle, clinical history including all examinations (e.g. blood analysis), treatment details, medications and all basic information available about aneurysms (e.g. location or type).

In total data of 1420 subjects suspected aneurysms has been collected, but fortunately only 1000 subjects actually had one or more aneurysms. A total of 900 image studies were collected for subjects with confirmed aneurysms, and 550

of these belonged to an image modality (3DRA) which enables the computation of hemodynamic and morphological descriptors of an aneurysm. These features were derived using complex and time-consuming analyses [24]. Clinical information as well as hemodynamic and morphological characteristics of almost 200 aneurysms were available from the project. Unfortunately, missing clinical data items (e.g. patient age, weight, or the clinical confirmation of aneurysm’s location) in some of the records lead to a total of 157 cases with complete information used for this work.

Although the considerable wealth of data contained in the @neurIST information system is not yet fully exploited, the number of instances described above still make it a larger dataset than those reported in previous studies of cerebral aneurysms [6,7,26].

### 3.2 Feature Extraction

As outlined in Section 1, a wide variety of aneurysm characteristics are currently believed to influence rupture. This, in turn, demands a significant heterogeneity and multimodality of the data needed to create the features included in the prediction models.

**Table 2.** Clinical Features

Num.	Feature Name	Description/Values
1	Gender	Male, Female
2	Age	Positive integer $\in [22,84]$
3	Smoking Status	No, Yes-still, Yes-quit
4	Hypertension Status	No, Yes-controlled, Yes-poorly controlled
5	Height Width	Used to compute Body Mass Index (BMI)
6	Location of aneurysm	24 categorical values
7	Laterality of aneurysm	Left, Right, Midline
8	Status of aneurysm	Ruptured, UnRuptured
9	Aneurysm Type	Saccular-Bifurcation, -Side Wall Fusiform-Dissecting, -Non Dissecting
10	Aneurysm Aspect	Rough, Smooth

Regarding clinical information, Table 2 shows the set of features used as input for the models. Some of these features apply to the patient, and some others specifically to each aneurysm.

The collection of morphological descriptors used in our experiments has been detailed and computed following the workflow described in [24]. Table 3 shows this set of morphological features. Those include two large feature vectors generated by the extraction of so-called Zernike-moment invariants (ZMI) [17], which can be computed for the surface and for the volume of an aneurysm.

Finally, the hemodynamic features, which were also computed with the workflow detailed in [24], are extracted through two coupled analyses. The first one

<sup>1</sup> Three-dimensional morphological descriptors which, up to a level of detail, the shape can be recovered.



**Table 3.** Morphological Features

Num.	Feature Name	Description
11	Volume	Volume of the aneurysm dome
12	Surface	Surface area of the aneurysm dome
13	Neck Surface	Surface area of the neck of an aneurysm
14	Neck Width	Width of the neck of an aneurysm
15	Depth	Depth of an aneurysm
16	Aspect Ratio	Depth/neck width
17	Non-Sphericity Index	Quantifies the difference between the aneurysm shape and a perfect circumscribed sphere
18-138	Surface ZMI	Surface-based Zernike Moments Invariants
139-259	Volume ZMI	Volume-based Zernike Moments Invariants

is a 1D model of the human systemic circulation, and predicts the flow rate and pressure waveforms at predefined points in the affected vessel. The second is a 3D analysis, which used the results from the first as boundary conditions. Table 4 highlights the types of available features. For the sake of clarity, not all features are listed here. We omitted (called 'other calculations') those that represent the values of features at different time points during the cardiac cycle (systole or diastole), or the maximum, minimum, and average of these values. Further specific examples are shown in Table 5 and Table 7. The reader is referred to [6,7,24,26] for a detailed description of such features.

**Table 4.** Hemodynamic Features

Num.	Feature Name	Description
260	Wall Shear Stress (WSS)	Frictional tangential force exerted by blood flow on endothelial layer
261	Oscillatory Shear Index (OSI)	Measure of oscillation of shear forces on endothelial layer over cardiac cycle
262	Velocity at Peak Systole	Mean speed of blood-flow inside the aneurysm at peak systole
263	Flow Type	Shear-driven, momentum-driven or mixed type of blood-flow
264	Flow Stability	Stable or unstable blood-flow inside the aneurysm
265	Viscous Dissipation	Rate at which mechanical energy is converted into heat inside the aneurysm
266	Area Elevated WSS	Area in which the WSS is above average
267	Velocity in Aneurysm	Mean speed of blood-flow in the aneurysm during the cardiac cycle
268	Influx Area at Neck	Area of the neck surface in which blood flows into the aneurysm
269	Number Vortexes	Number of vortexes of the blood-flow in the aneurysm
270	Energy Flux	Kinetic energy created by the blood-flow through the aneurysm during the cardiac cycle
271-294	(other calculations related to these features)	

The description above indicates the interdisciplinary nature of the effort required to incorporate the full wealth of data in our experiments. Although the main outcomes reported here are related to the data mining field, feature extraction was certainly a large undertaking and required collaboration between medical image processing, computerised fluid dynamics, clinicians, and data modelling experts.

### 3.3 Feature Selection

The set of potentially useful features are in the order of hundreds for medical images alone [19]. In the context of our work, the situation is further complicated by the introduction of hemodynamic features, and a large clinical schema. However, considering the whole feature set can lead to the *curse of dimensionality* problem, where the significance of each feature decreases [19]. This, in turn, reduces the accuracy of classification algorithms.

Feature selection aims at identifying the subset of features that are more relevant for the task at hand. In this case, differentiating between ruptured and unruptured aneurysms. All the continuous features described in Section 3.1 were tested for statistical significance, following [18]. This algorithm essentially considers, for each feature independently, the values for all the ruptured cases, and the values for the unruptured cases. Then it applies a t-test to compare whether there is sufficient evidence that these two sets of values belong to different populations. At a 90% confidence level, the set of relevant features is shown in Table 5. In this table, ‘ZMI’ stands for the above-mentioned Zernike moment invariants [17].

**Table 5.** Statistically Significant Features (sorted by ‘p-value’)

Feature Name	p-value
Non-Sphericity Index	0,00002
Relative Area Elevated Press Gauge at Peak	0,00027
ZMI Surface 8	0,00064
Absolute Area Elevated Press Gauge at Peak	0,00087
Max OSI	0,00185
Number Vortexes in Aneurysm at Peak	0,00628
Avg Velocity in Aneurysm Time Avg	0,00811
Relative Area Elevated WSS at Peak	0,00885
ZMI Surface 1	0,01587
Relative Area WSS Above Threshold Time Avg	0,03136
Relative Influx Areaatneckatpeak	0,06821
Max Velocity in Aneurysm Time Avg	0,07619
Absolute Influx Area at Neck at Peak	0,08289
Absolute Area Elevated OSI	0,09356
Relative Area WSS Below Threshold Time Avg	0,09746
ZMI Volume	45 different features

This process produced an 80% dimensionality reduction. The resulting relevant features are consistent with a recent study which focuses on identifying good discriminant features for intracranial aneurysm rupture [26]. Our results are a superset of those obtained in this other study, except for the *non-sphericity index* which is highly relevant in our dataset. It must also be noted that [26] did not include moment invariants [17] in their study, which accounts for a large part of the dimensions to be dealt with in our work. Finally, all categorial features shown in Table 2 were also included when building the classifiers in this paper. In total, 71 different features were used.

### 3.4 Feature Discretisation

The features shown in Table 5 all take continuous values. Not all prediction models reported in Section 4 can be applied to continuous values, but but require these

features to be previously discretised. Discretisation algorithms have played an important role in data mining and knowledge discovery. They not only produce a concise summarisation of continuous attributes to help the experts understand the data more easily, but also make learning more accurate and faster [22].

The discretisation process applied the OMEGA algorithm [19]. This scheme does not create an equidistant discretisation, by which all intervals of a given feature would have the same size. In contrast, it computes an inconsistency rate for each interval, which is directly proportional to the number of different classes (in our case, 'ruptured' or 'unruptured') of the instances that belong to the interval. The algorithm defines the intervals so that the inconsistency rate falls below a user-defined threshold (in our case tuned to 35%, [19] used 30%).

The OMEGA algorithm can also be used to complement the feature selection process described in Section 3.3. If the global inconsistency rate resulting from discretising a given feature falls above the user-defined threshold, it is recommended not to use such feature in the training process, as it is likely to reduce the classifier's accuracy [19].

## 4 Prediction Model

This section describes the set of classifiers<sup>3</sup> we have experimented with and compared in order to develop a prediction model for the rupture of an aneurysm.

**SVM (Support Vector Machines).** SVM is a collection of methods for classification of both linear and non-linear data [3][14][25]. It transforms the original data into a high dimensional space, from where a hyper-plane is computed to separate the data according to the class of each instance. A new instance can be classified by mapping it into a point into the same dimensional space, which is already divided (classified) by the hyper-plane.

**Decision Tree.** A decision tree is a classifier that recursively partitions the training dataset until each partition belongs to (mainly) one class. Deciding which feature to use at each partitioning step determines the accuracy of the classifier. Information gain measures are used in this context [25].

**Bayesian Classifier.** This classifier assigns the highest posterior probability class using the prior probability computed from the observed instances. This approach usually performs well in practice [25], even if it makes two assumptions that do not necessarily hold. Firstly, all features are assumed to be independent. Secondly, numeric values are assumed to follow a normal distribution for each feature.

---

<sup>2</sup> Features 'Number Vortexes in Aneurysm at Peak' and 'Relative Area WSS Below Threshold Time Avg' of Table 5 would have been removed this way. However, the results reported in this paper did not apply this possibility, taking a conservative approach not to exclude too many features.

<sup>3</sup> Prototyped using KNIME, <http://www.knime.org/>, except for CPAR

**Fuzzy Rules.** This algorithm generates rules which are fuzzy intervals in higher dimensional spaces. These hyper-rectangles are defined by trapezoid fuzzy membership functions for each dimension. Each rule consists of one fuzzy interval for each dimension plus the target classification columns along with a number of rule measurements node [4].

**Neural Networks.** Neural networks are often used for predicting numerical quantities [25]. The networks are composed of non-linear computational elements arranged in patterns reminiscent of biological neural networks. Computational elements are arranged in several layers via weights that are adapted during the training phase. Probabilistic neural networks [5] were used in our experiments, as they do not require the network topology to be defined in advance, produce more compact representations, and show better performance than traditional multilayer perceptions.

**Predictive Association Rules.** Classification based on Predictive Associative Rules (CPAR) [27]<sup>4</sup> is a rule induction algorithm that avoids generating large item sets collections, as is the case of more traditional association rule algorithms such as FP-growth. Rules are created for each of the classes in the training set. These rules are incrementally constructed by adding attributes to the antecedent. A measure of ‘gain’ in the classifier’s accuracy is computed in order to select which attribute to add to the antecedent of a rule.

The input data to this algorithm must be binary. This is achieved by first discretising the continuous features, see Section 3.4. Then, a new feature is created for each interval, and value 1 is assigned to the feature representing the appropriate interval each instance belongs to.

**Table 6.** Results for Classifiers’ Performance (in %)

Classifier	TP	FP	Precision	Recall	Specificity	Class
SVM Binary	95.2	4.8	95.2	96.3	94.7	Unruptured
	95.9	4.1	95.9	94.7	96.3	Ruptured
SVM	75.3	24.7	75.3	70.7	74.7	Unruptured
	70.0	30.0	70.0	74.7	70.7	Ruptured
Decision Tree	60.0	40.0	60.0	58.5	57.3	Unruptured
	55.8	44.2	55.8	57.3	58.5	Ruptured
Naive Bayes	68.7	31.3	68.7	69.5	65.3	Unruptured
	66.2	33.8	66.2	65.3	69.5	Ruptured
Fuzzy Rules	63.6	36.4	63.6	72.1	54.8	Unruptured
	64.2	35.8	64.2	54.8	72.1	Ruptured
Neural Networks	64.6	35.4	64.6	75.6	54.7	Unruptured
	67.2	32.8	67.2	54.7	75.6	Ruptured
CPAR	75.4	24.6	75.4	86.0	68.0	Unruptured
	48.6	51.4	48.6	48.6	58.6	Ruptured

<sup>4</sup> Implementation modified from

[http://www.csc.liv.ac.uk/~sim\\$frans/KDD/Software/](http://www.csc.liv.ac.uk/~sim$frans/KDD/Software/)

## 5 Experimental Results

All classifiers discussed in Section 4 have been used to develop a prediction model of aneurysm rupture, and compare their respective accuracies, using a *ten-fold cross-validation* [25] approach.

### 5.1 Classification Performance

Table 6 summarises the performance of the different classifiers. Column ‘true positives’ (TP) is the number of instances that have been correctly classified. Analogously for ‘false positives’ (FP). *Precision* is the proportion of instances in a given class which are correctly classified. The *recall* (or *sensitivity*) measures the proportion of true positives which are correctly identified as such. The *specificity* is the proportion of negatives which are correctly identified.

It should be noted that SVM is designed to operate on continuous values. However, the discretised dataset, described in Section 3.3, was used also in these experiments with SVM, and the resulting accuracy outperformed that obtained without discretization. This approach is referred to as ‘SVM Binary’.

Fig. 2 shows the overall classification accuracy achieved by each classifier. Clearly, the Supporting Vector Machines (SVM) approach, with the discretised and binary input, outperforms all other methods.

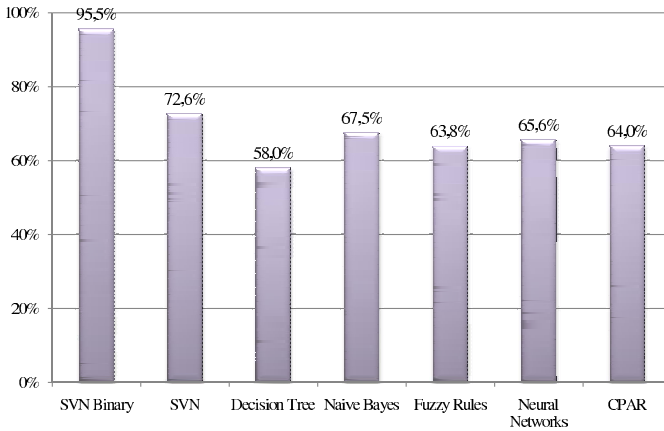


Fig. 2. Summary of Classifiers’ Accuracy

### 5.2 Association Rules and Risk Factors

The outcome of a classification based on predictive association rules (CPAR), in addition to showing an accuracy comparable to other methods (besides SVM), has also another essential added value. The rules generated to construct the classifier explicitly identify features and feature values that strongly influence the aneurysms rupture status. As justified in Section 1, this domain is not yet sufficiently understood as to justify treatments other than invasive intervention.

**Table 7.** List of Aneurysm Rupture Predictive Association Rules

Num.	Accuracy	Status	Rule Body (feature values)
1	92%	Unruptured	← Location: Intracavernous internal carotid
2	90%	Unruptured	← Location: Ophthalmic segment carotid, Side: Left
4	88%	Unruptured	← Location: Intracavernous internal carotid, Number Vortexes in Aneurysm at Peak = 1
22	83%	Unruptured	← NON Sphericity Index $\in$ 17 [0.20634225,0.213549286]
27	80%	Unruptured	← NON Sphericity Index $\in$ 26 [0.287403159,0.291000392]
33	80%	Ruptured	← NON Sphericity Index $\in$ 8 [0.14925828,0.155347234]
13	86%	Unruptured	← Female, Side: null
15	85%	Ruptured	← Number Vortexes in Aneurysm at Peak = 2, Side: Right , HyperTension: No, Aneurysm Type: Saccular - Bifurcation
30	80%	Ruptured	← Number Vortexes in Aneurysm at Peak $\in$ [4,0,7,0]
6	88%	Unruptured	← Absolute Area Elevated OSI $\in$ 9 [5.8E-7,6.77E-7]
10	87%	Unruptured	← Absolute Area Elevated OSI $\in$ 5 [2.19E-7,3.93E-7]
23	83%	Unruptured	← Absolute Area Elevated OSI $\in$ 18 [1.72E-6,2.09E-6]
29	80%	Unruptured	← Absolute Area Elevated OSI $\in$ 12 [7.05E-7,7.95E-7]
26	81%	Ruptured	← Absolute Area Elevated OSI 25 [4.23E-6,5.53E-6]
8	88%	Ruptured	← ZMI Surface 8 $\in$ 26 [0.137805704,0.143486214]
35	80%	Ruptured	← ZMI Surface 8 $\in$ 14 [0.099828474,0.101293597]
21	84%	Ruptured	← ZMI Surface 1 $\in$ 20 [3.4E-9,6.49E-9]
36	75%	Unruptured	← Avg Velocity in Aneurysm Time $\in$ Avg 26 [0.04449,0.06722]
3	90%	Ruptured	← Avg Velocity in Aneurysm Time $\in$ Avg 25 [0.03688,0.04449]
18	85%	Ruptured	← Avg Velocity in Aneurysm Time $\in$ Avg 8 [0.01281,0.01382]
31	80%	Ruptured	← Avg Velocity in Aneurysm Time Avg $\in$ 10 [0.0152,0.01653]
39	71%	Ruptured	← Avg Velocity in Aneurysm Time Avg $\in$ 29 [0.07305,0.09617]
28	80%	Unruptured	← Absolute Area Elevated Press Gauge at Peak $\in$ 6 [5.8E-7,7.33E-7]
37	75%	Unruptured	← Absolute Area Elevated Press Gauge at Peak $\in$ 2 [7.93E-8,1.57E-7]
34	80%	Ruptured	← Absolute Area Elevated Press Gauge at Peak $\in$ 8 [8.3E-7,9.55E-7]
5	88%	Unruptured	← Relative Area Elevated WSS at Peak $\in$ 33 [17.7,42.6000000001], Smoker: No
38	75%	Unruptured	← Relative Area Elevated WSS at Peak $\in$ 12 [1.6,1.8]
7	88%	Ruptured	← Relative Area Elevated WSS at Peak $\in$ 13 [1.8,2.3]
25	83%	Ruptured	← Relative Area Elevated WSS at Peak $\in$ 19 [3.9,4.5]
24	83%	Ruptured	← Relative Area Elevated Press Gauge at Peak $\in$ 11 [2.3,3.3]
9	87%	Unruptured	← Relative Area WSS Below Threshold Time Avg $\in$ 8 [4.3,7.3]
20	85%	Ruptured	← Relative Area WSS Below Threshold Time Avg $\in$ 10 [11.3,16.7]
11	87%	Ruptured	← Relative in Flux Area at Neck at Peak $\in$ 6 [33.5,37.2]
19	85%	Ruptured	← Relative in Flux Area at Neck at Peak $\in$ 30 [58.8,63.7]
16	85%	Ruptured	← Relative in Flux Area at Neck at Peak $\in$ 8 [37.8,39.0]
14	85%	Unruptured	← Absolute in Flux Area at Neck at Peak $\in$ 29 [1.04E-5,1.11E-5]
32	80%	Ruptured	← Absolute in Flux Area at Neck at Peak $\in$ 25 [7.6E-6,8.09E-6]
12	87%	Ruptured	← Max Velocity in Aneurysm Time Avg $\in$ 27 [0.8205,0.9355]
17	85%	Ruptured	← Max Velocity in Aneurysm Time Avg $\in$ 32 [1.326,1.643]

It is expected that this type of rules can contribute to structure the accepted knowledge in the domain.

Table 7 shows the set of 39 rules generated by this classifier. It includes the *Laplace accuracy* of each rule, calculated as follows:

$$Accuracy = \frac{N_c + 1}{N_{tot} + Number\ of\ Classes}$$

being,

$N_c$  = Number of records in training set with all attributes in the rule, i.e. support.

$N_{tot}$  = Number of records in training set with all the attributes in the rule's body.

This table groups together all those rules that contain the same features in their body, so that the influence of these features in predicting rupture is

highlighted. The rule number would result from sorting all rules according to their accuracy. For example, rule number 1 states that a given aneurysm location (Intracavernous internal carotid), is very strongly correlated to aneurysms that do not rupture. Also, rules number 22, 27, and 33 correlate the non-sphericity index [24] to the rupture status of an aneurysm, so that a low value of such index often appears with a ruptured aneurysm. For discretised features, the rules show the interval number the value would belong to, plus the actual limits of the interval. For example, the body: NON Sphericity Index  $\in$  8 [0.14,0.15], indicates that the value belongs to the eight interval generated by the discretisation algorithm. Intervals with smaller limits have lower internal numbers.

Conversely, it is also interesting to observe how some features have surprisingly not been identified as strongly correlated to ruptured status. Specifically, 'smoking status' does not appear associated to ruptured aneurysms. This would suggest that, according to our dataset, there is no evidence to consider this a risk factor. A similar conclusion can be drawn about 'hypertension status'. These are two examples of risk factors, commonly accepted in this domain, which can not be confirmed by our dataset, which is, as stated above, of significant size as compared to other studies in the same domain.

## 6 Conclusions

This paper has presented some initial results of applying data mining techniques to a complex biomedical domain, with the goal of predicting the rupture of cerebral aneurysms. The features used in this context are of very different nature, and exploit the latest advances in medical image processing, and hemodynamic simulations, together with the state-of-the-art clinical knowledge of this domain.

It has described how 6 well-known classifiers were used to create such prediction models based on these features. The results show that SVM (supporting vector machines) offer the best results (as was expected from related studies applied to other clinical conditions [14]), with accuracy rates as high as 95%.

The CPAR algorithm used in the discussion of Section 5.2 was run several times. The predictive association rules shown in Table 7, however, are those mined in one of these executions, for which the reported accuracy of the classifier was 86%. These set of predictive association rules represents the first attempt to explicitly capture the strong evidence and causal relationship between feature values and aneurysm ruptured status, according to our dataset.

## 7 Future Work

Future research will address mainly two lines of work, the data pre-processing step and the prediction model. Section 3.1 pointed out how missing values significantly reduced the size of the dataset available in practice. The instances with missing values were simply eliminated. Mining over incomplete data is still a research challenge, and some approaches exist, such as *expectation maximisation* [15] and *conceptual reconstruction* [16]. These need to be evaluated in the

context of this complex application domain. The main goal is to estimate the most appropriate values for these missing features, so that they do not alter the role of the feature in the classification model, but the other feature values that have indeed been informed can be used to strengthen the accuracy and support of the prediction models. This would be the case if the models could be built from close to 500 instances, instead of the current 157 (see Section 3.1).

The feature discretisation being used [19] defines the so-called *cut-points* in the feature values every time a change in the category (ruptured/unruptured) of the instances is observed. Typically near the interval limits there are several instances of different categories and this could lead to many different small intervals, which are likely to confuse the classifiers. In case of discrete features, furthermore, this could cause that the inconsistency rate calculated by the algorithm differs from the one actually present in the dataset used to build the prediction model.

The feature selection approach used in this paper is essentially based on statistical significance, which considers each feature in isolation. The outcomes are comparable to those obtained in [26], which used *multivariate logistic regression analysis*. However, since the outcomes of the selection process are contradicting the somewhat accepted clinical knowledge in the field (e.g. age and body mass index of the patient, and aspect ratio of an aneurysm are not statistically significant), further work is needed in this direction.

Regarding the prediction model, the explanation power of association rules cannot be underestimated, particularly for a domain which still lacks sufficient evidence on the best course of treatment (see Section II). Further work is needed in order to produce a set of predictive rules which are common over all runs, thus not dependent on the actual sampling of the instances used for training.

Finally, the data used in this work is of very different nature (clinical, hemodynamics and morphological), and even of different quality, in the sense that a clinical observation is likely to be more reliable than a value derived from a simulation, which relies on a number of assumptions (e.g. boundary conditions which are themselves results of a simulation, material properties). It would be necessary to quantify the uncertainty introduced by these differences, for example, in the prediction models, and even in the accuracy of each individual predictive association rule (depending on which features it includes in its body).

**Acknowledgments.** This work was partially funded by the Integrated Project @neurIST (FP6-IST-027703), which was co-financed by the European Commission, and by the Spanish Ministry of Innovation and Science (MICINN) through the cvREMOD project (CEN-20091044) under the CENIT programme of the Industrial and Technological Development Center.

## References

1. Antonie, M., Zaiane, O., Coman, A.: Application of data mining techniques for medical image classification. In: Proceedings of the Second International Workshop Multimedia Data Mining, with ACM SIGKDD, pp. 94–101 (2001)



2. Benkner, S., Arbona, A., Berti, G., Chiarini, A., Dunlop, R., Engelbrecht, G., Frangi, A.F., et al.: @neurIST: Infrastructure for advanced disease management through integration of heterogeneous data, computing, and complex processing services. *IEEE Transactions on Information Technology in Biomedicine* 14, 126–131 (2010)
3. Bennett, K.P., Campbell, C.: Support vector machines: Hype or hallelujah? *SIGKDD Explorations Newsletter* 2, 1–13 (2000)
4. Berthold, M.R.: Mixed fuzzy rule formation. *International Journal of Approximate Reasoning* 32(2-3), 67–84 (2003)
5. Berthold, M.R., Diamond, J.: Constructive training of probabilistic neural networks. *Neurocomputing* 19(1-3), 167–183 (1998)
6. Cebal, J., Mut, F., Weir, J., Putman, C.: Association of hemodynamic characteristics and cerebral aneurysm rupture. *American Journal of Neuroradiology* 32, 264–270 (2011)
7. Chien, A., Castro, M., Tateshima, S., Sayre, J., Cebal, J., Vinuela, F.: Quantitative hemodynamic analysis of brain aneurysms at different locations. *American Journal of Neuroradiology* 30, 1507–1512 (2009)
8. Dunlop, R., Arbona, A., Rajasekaran, H., Lo Iacono, L., Fingberg, J., Summers, P., Benkner, S., Engelbrecht, G., Chiarini, A., Friedrich, C.M., Moore, B., Bijlenga, P., Iavindrasana, J., Hose, R.D., Frangi, A.F.: @neurIST - chronic disease management through integration of heterogeneous data and computer-interpretable guideline services. *Stud. Health Technol. Inform.* 138, 173–177 (2008)
9. Frangi, A.F., Hose, R., Ruefenacht, D.: The @neurIST project: Towards understanding cerebral aneurysms (2007)
10. Friedrich, C.M., Dach, H., Gattermayer, T., Engelbrecht, G., Benkner, S., Hofmann-Apitius, M.: @neurIST - chronic disease management through integration of heterogeneous data and computer-interpretable guideline services. *Stud. Health Technol. Inform.* 138, 165–172 (2008)
11. Iavindrasana, J., Depeursinge, A., Ruch, P., Spahn, S., Geissbuhler, A., Müller, H.: Design of a decentralized reusable research database architecture to support data acquisition in large research projects. *Stud. Health Technol. Inform.* 129, 325–329 (2007)
12. Johnston, S., Wilson, C.B., Halbach, V., Higashida, R., Dowd, C., McDermott, M., Applebury, C., Farley, T., Gress, D.: Endovascular and surgical treatment of unruptured cerebral aneurysms: comparison of risks. *Annals of Neurology* 48, 11–19 (2000)
13. Kurkure, U., Chittajallu, D., Brunner, G., Le, Y., Kakadiaris, I.: A supervised classification-based method for coronary calcium detection in non-contrast CT. *International Journal of Cardiovascular Imaging* 26, 9817–9828 (2010)
14. Lee, H.G., Nohand, K.Y., Ryu, K.H.: A data mining approach for coronary heart disease prediction using HRV features and carotid arterial wall thickness. In: *Proceedings of the 2008 International Conference on BioMedical Engineering and Informatics*, pp. 200–206. IEEE Computer Society, Los Alamitos (2008)
15. Little, R.J.A., Rubin, D.B.: *Statistical Analysis with Missing Data*, 2nd edn. Wiley, Chichester (2002)
16. Parthasarathy, S., Aggarwal, C.: On the use of conceptual reconstruction for mining massively incomplete data sets. *IEEE Transactions on Knowledge and Data Engineering* 15(6), 1512–1521 (2003)
17. Pozo, J.M., Villa-Uriol, M.C., Frangi, A.F.: Efficient 3D geometric and Zernike moments computation from unstructured surface meshes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 471–484 (2011)

18. Ribeiro, M., Balan, A., Felipe, J., Traina, A., Traina, C.: Mining Complex Data, Studies in Computational Intelligence. In: Mining Statistical Association Rules to Select the Most Relevant Medical Image Features, vol. 165, pp. 113–131. Springer, Heidelberg (2009)
19. Ribeiro, M., Traina, A.M., Traina, C., Rosa, N., Marques, P.: How to improve medical image diagnosis through association rules: The IDEA method. In: Proceedings of the 21st IEEE International Symposium on Computer-Based Medical Systems, pp. 266–271. IEEE Computer Society, Los Alamitos (2008)
20. Roos, Y.B., Dijkgraaf, M.G., Albrecht, K.W., Beenen, L.F., Groen, R.J., de Haan, R.J., Vermeulen, M.: Direct costs of modern treatment of aneurysmal subarachnoid hemorrhage in the first year after diagnosis. *Stroke* 33, 1595–1599 (2002)
21. Tan, X., Han, H.P.Q., Ni, J.: Domain knowledge-driven association pattern mining algorithm on medical images. In: Proceedings of the 2009 Fourth International Conference on Internet Computing for Science and Engineering, pp. 30–35 (2009)
22. Tsai, C., Lee, C., Yang, W.: A discretization algorithm based on class-attribute contingency coefficient. *Information Sciences* 731, 714–731 (2008)
23. Valencia, C., Villa-Uriol, M.C., Pozo, J.M., Frangi, A.F.: Morphological descriptors as rupture indicators in middle cerebral artery aneurysms. In: EMBC, Buenos Aires, Argentina, pp. 6046–6049 (September 2010)
24. Villa-Uriol, M.C., Berti, G., Hose, D.R., Marzo, A., Chiarini, A., Penrose, J., Pozo, J., Schmidt, J.G., Singh, P., Lycett, R., Larrabide, I., Frangi, A.F.: @neurIST complex information processing toolchain for the integrated management of cerebral aneurysms. *Interface Focus* (2011)
25. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco (2000)
26. Xiang, J., Natarajan, S.K., Tremmel, M., Ma, D., Mocco, J., Hopkins, L.N., Siddiqui, A.H., Levy, E.I., Meng, H.: Hemodynamic-morphologic discriminants for intracranial aneurysm rupture. *Stroke* 42, 144–152 (2011)
27. Yin, X., Han, J.: CPAR: Classification based on predictive association rules. In: Proceedings SIAM International Conference on Data Mining, pp. 331–335 (2003)

# Semantic Translation for Rule-Based Knowledge in Data Mining

Dejing Dou, Han Qin, and Haishan Liu

Computer and Information Science Department  
University of Oregon  
Eugene, Oregon 97403, USA  
{dou,qinhan,ahoyleo}@cs.uoregon.edu

**Abstract.** Considering data size and privacy concerns in a distributed setting, it is neither desirable nor feasible to translate data from one resource to another in data mining. Rather, it makes more sense to first mine knowledge from one data resource and then translate the discovered knowledge (models) to another for knowledge reuse. Although there have been successful research efforts in knowledge transfer, the *knowledge translation* problem in the semantically heterogeneous scenario has not been addressed adequately. In this paper, we first propose to use Semantic Web ontologies to represent rule-based knowledge to make the knowledge computer “translatable”. Instead of an inductive learning approach, we treat knowledge translation as a deductive inference. We elaborate a translation method with both the forward and backward chaining to address the *asymmetry of translation*. We show the effectiveness of our knowledge translation method in decision tree rules and association rules mined from sports and gene data respectively. In a more general context, this work illustrates the promise of a novel research which leverages ontologies and Semantic Web techniques to extend the knowledge transfer in data mining to the semantically heterogeneous scenario.

## 1 Introduction

Information resources distributed across the Internet present structurally and semantically heterogeneous data that are hard to process automatically for knowledge acquisition. These resources include online databases, web services and the Semantic Web [6]. They provide a unique and challenging opportunity for knowledge acquisition in new and meaningful ways. Although standards such as SQL, XML, and OWL [1] reduce the *syntactic* diversity, it is unreasonable to expect schemas or ontologies that describe the *structure* and *semantics* of data to be few in number [7]. A variety of heterogeneity has been observed in different data analysis tasks [17]. It will be extremely helpful for data analysts to have a system that can automatically reuse the knowledge mined from one data resource to another. The traditional data mining solution to the semantic heterogeneity is to first apply data translation or data integration as a pre-processing step to either translate data from one schema to another or integrate data from several

resources into a centralized location (e.g., a data warehouse). However, the data translation or integration approach is not ideal because of following two reasons: (i) communication and storage costs make transferring huge volumes of data infeasible; (ii) individual sites want to maintain privacy (e.g., SSN) and do not want to share this information with other parties.

*Knowledge translation* is a method to overcome semantic heterogeneity by translating knowledge from one data resource (source) to another (target) to make the knowledge reusable or comparable. It is closely related to *knowledge transfer* or *transfer learning* which focuses on applying knowledge gained in solving one problem to a different but related problem. However, to make existing knowledge transfer algorithms work, it is necessary that the target domain has enough or at least auxiliary training data. It may not be true in the real world problems. For example, a new credit card company without historical data wants to use the classification model generated by its collaborative credit card company to determine whether applicants to this new company are qualified or not. The new company and its collaborative company may use different schemas to store their applicants' data. One way is to translate the applicants' data of the new company to the schema of its collaborator's and use the collaborator's classification model to conduct prediction. However, due to communication costs and privacy issues, it is plausible to translate only knowledge described using its collaborator's schema to the new company's schema without carrying out data translation. It is surprising that little research in knowledge transfer has been done in the semantically heterogeneous scenario where two data resources have different but semantically related representations. For example, the semantic heterogeneities between the attributes describing two credit company databases may include synonyms (e.g., salary vs. pay), subsumption (e.g., graduate\_student\_status vs. student\_status) or functional (e.g., concatenation of customer\_firstname and customer\_lastname vs. customer\_fullname) and so forth. It is hard to say that we can directly apply knowledge transfer algorithms to solve the *knowledge translation* problem in the semantically heterogeneous scenario.

To solve the knowledge translation problem, we first identify two critical theoretical challenges need to be addressed: i) there is no standard formal language to represent semantics of mined knowledge. Previous data mining research in exploring mined knowledge mainly focuses on visualization for aiding human comprehension. Although some XML-based languages, e.g., Predictive Model Markup Language (PMML), have been developed to represent a standard syntax for mined knowledge, few work has been done for helping computers "understand" the semantics of knowledge automatically. We need to represent the knowledge in a way that is computer "translatable" from the source to the target if the source and target use different but semantically related schemas. And ii) what criteria can justify a *correct* translation remains a theoretically hard problem when we try to design algorithms for knowledge translation.

To address these two challenges, in this paper, after introducing more related work (Section 2), we propose to use the standard Semantic Web ontology languages to formally represent the rule-based knowledge, such as decision tree

rules and association rules (Section 3). Ontologies, which have been used for formal specification of conceptualization in traditional knowledge engineering and the emerging Semantic Web, will be used for representing IF-THEN data mining rules. We then give definition of correctness and completeness of knowledge translation based on the soundness of formal inference. We point out the asymmetry of translation based on our definitions. The conditions of rules can be translated from the source to the target by backward chaining with generalized modus ponens. The conclusions of rules can be translated from the source to the target by forward chaining (Section 4). We show the effectiveness of our knowledge translation method through two case studies in real world data (Section 5). We discuss the uncertainty issues in knowledge translation and its extension to distributed data mining as our future work (Section 6). We finally conclude the paper with our contributions in Section 7.

## 2 Related Work

Knowledge transfer and distributed data mining are major research areas which also deal with heterogeneous data and knowledge resources in data mining.

Knowledge transfer focuses on applying knowledge gained in solving one problem to a different but related problem. It is also treated as one kind of knowledge reuse. Gao *et al.* [15] proposed a locally weighted ensemble framework to combine multiple models. Gupta *et al.* [16] presented an algorithm for leveraging heterogeneous data resources and algorithms using different algorithms for each knowledge resource (e.g., Wikipedia). Eaton [14] proposed a method on learning across multiple resolutions of input and applied this technique to the problem of knowledge transfer in multitask learning. However, most knowledge transfer researches in data mining focus on the source and target data resources with different models, structures or distributions. None of them has discussed the semantic heterogeneities, such as synonyms, subsumption and functions, between the source and target. Also, previous knowledge transfer research does not handle the scenario that there is no data in the target resource.

In distributed data mining (DDM), the heterogeneity that has been studied is mainly focused on the scenario where only incomplete knowledge can be observed at each local site [19]. It is also termed as *vertical fragmentation* in DDM literature [8,20]. Although vertical fragmentation is not the same as the semantically heterogeneous scenario we focus on in this paper, some previous experience is helpful. Of particular interest, Caragea and colleagues [8] used attribute value taxonomy (AVT) to represent the semantics of local data resources. A user ontology and a set of simple interoperation constraints (e.g., equivalence and subsumption) between the data resource ontology and user ontology are first specified manually. Then mappings between local ontologies and the user ontology can be derived and used to answer statistical queries. In the paper, we prefer to use mappings to translate the generated knowledge instead of only translating the statics of queries.

### 3 Formal Representation of Rule-Based Knowledge

To formally represent mined knowledge, it is important to understand the semantics of the knowledge. We basically borrow the ideas of the Semantic Web [6] which targets at making the web data computer “understandable” and sharable among software agents.

Different data mining tools (e.g., Weka [22], XLMiner [4]) may use different syntax and semantics to represent results (e.g., decision trees). The previous data mining research has focused on the visualization of mined knowledge to help human understanding. However, to enable automatic knowledge translation, we represent the knowledge in a formal way that computers can “understand” and process. To that end, we choose ontologies, which are formal specifications of conceptualization, to formally describe the mined knowledge (e.g., decision tree rules and association rules). Specifically, we leverage the research progress in the Semantic Web on Web Ontology Language (OWL [1]) and one of its related rule languages, SWRL [3], to help formally represent data mining knowledge. Both OWL and SWRL are based on fragments of first order logic (i.e., Description Logics and Horn Logic).

Ontologies can be designed by domain experts manually or be mined from data resources semi-automatically [10]. Once the ontologies are constructed, the classes, properties and axioms of ontologies will be used to describe the knowledge discovered from that data resource. For example, one decision tree rule from a credit card company data can be represented in SWRL. The SWRL uses OWL/RDF syntax which is space consuming. To save the space, we use the general first order logic (FOL) syntax to represent SWRL rules in the paper:

$$\forall x, y \text{ Applicant}(x) \wedge \text{age}(x, y) \wedge (y > 30) \wedge \text{grad\_student}(x, \text{“Y”}) \rightarrow \text{credit}(x, \text{“Good”})$$

where “Applicant” is an OWL class (i.e., unary predicate), “age”, “>”, “graduate\_student”, and “credit” are OWL properties (i.e., binary predicates). This rule means: “IF an applicant’s age is larger than 30 AND the applicant is a graduate student, THEN his or her credit is good”.

The semantic heterogeneities between two data resources are represented in a formal way with ontological concepts as well. For example, if one credit card company uses the “age” concept but another one uses “birth\_year” to record the applicants’ information, the heterogeneity or mapping between them can also be represented in SWRL:

$$\begin{aligned} \forall x, y \text{ birth\_year}(x, y) \rightarrow \text{age}(x, \text{Current\_year} - y) \\ \forall x, y \text{ age}(x, y) \rightarrow \text{birth\_year}(x, \text{Current\_year} - y) \end{aligned}$$

where “Current\_year” is a constant of number (e.g., 2011). We notice that RIF [2] is the rule interchange format for the Semantic Web currently under discussion at W3C. As long as RIF becomes a W3C standard, we will change our representation and implementation from SWRL to RIF. The major point of this

paper is that we need an ontology-based rule language to represent data mining knowledge to enable formal reasoning and translation.

## 4 Knowledge Translation

After we use OWL and SWRL (or RIF) to represent rule-based knowledge and mappings, we can discuss the knowledge translation problem in a formal way.

### 4.1 Formal Definitions

We first formally define the correctness and completeness of knowledge translation (KT).

**Definition 1. Knowledge Translation (KT):** Let  $K_s$  be the knowledge mined from the source resource with ontology  $O_s$ ,  $O_t$  be the target ontology, and the set of mapping rules between  $O_s$  and  $O_t$  be  $\Sigma$ . Knowledge translation (KT) is the process to translate  $K_s$  represented by  $O_s$  to the knowledge  $K_t$  represented by  $O_t$  according to  $\Sigma$ . We use the symbol  $\rightsquigarrow_K$  to indicate the process of knowledge translation.

Since  $K_s$  and  $K_t$  can be described by ontology languages, they are a set of logical true statements.  $\Sigma$  are also a set of logical true statements as rules.

**Definition 2. Correctness of KT:** Let  $K_t$  be the translated knowledge determined by some algorithm. The  $K_t$  is considered as a correct translation from  $K_s$  with  $\Sigma$  only if  $K_t$  is a semantic consequence of  $\Sigma$  and  $K_s$ :  $(K_s; \Sigma) \rightsquigarrow_K K_t$  only if  $(K_s; \Sigma) \models K_t$ , where  $\models$  means the semantic consequence (i.e., a logical entailment). It can be implemented using inference as long as the inference is sound:  $(K_s; \Sigma) \vdash K_t \Rightarrow (K_s; \Sigma) \models K_t$ , where  $\vdash$  means an inference.

The above definition for correctness means that if  $K_s$  and  $\Sigma$  are true, a correct translation will guarantee that  $K_t$  is also true. A correct translation can be implemented by a sound inference. On the contrary, a simple rewriting of rules cannot guarantee to be sound. What kind of inference algorithms can be considered as sound will be further discussed later in this section. Also, if the  $\Sigma$  is 100% true, we expect that  $K_t$  has the same accuracy as  $K_s$  according to the translated data if it is a correct translation. If the mappings in  $\Sigma$  are not 100% true (i.e., with some uncertainty), the accuracy of  $K_t$  depends on the quality of mappings as well. We consider the later a harder problem and further discuss it in Section 6.

**Definition 3. Completeness of KT:** Let  $K_t$  be the translated knowledge generated by some algorithm.  $K_t$  is considered as a complete translation from  $K_s$  according to  $\Sigma$  only if all statements in  $K_s$  can be translated to  $K_t$  correctly.

However, a knowledge translation may not be complete if not all concepts in  $O_s$  can be mapped to concepts in  $O_t$ . It is normally the case for real world data resources. Therefore, we do not focus on getting a fully complete knowledge translation in this research but we will propose a way to handle the uncertainty of mappings in Section 6.

## 4.2 Asymmetry of Knowledge Translation

Continue with the decision tree rule from a credit card company (e.g.,  $C1$ ) data:

$$\forall x, y \text{ Applicant}(x) \wedge \text{age}(x, y) \wedge (y > 30) \wedge \text{grad\_student}(x, \text{"Y"}) \rightarrow \text{credit}(x, \text{"Good"})$$

If another credit card company (e.g.,  $C2$ ) uses different but semantically related attributes to describe its applicants: “birth\_year”, “student” and “credit\_ranking”. The mapping rules between corresponding attributes look like:

$$\begin{aligned} \forall x \text{ Applicant}(x) &\leftrightarrow \text{applicant}(x) \\ \forall x, y \text{ birth\_year}(x, y) &\rightarrow \text{age}(x, \text{Current\_year} - y) \\ \forall x, y \text{ age}(x, y) &\rightarrow \text{birth\_year}(x, \text{Current\_year} - y) \\ \forall x \text{ grad\_student}(x, \text{"Y"}) &\rightarrow \text{student}(x, \text{"Y"}) \\ \forall x, y \text{ credit}(x, y) &\leftrightarrow \text{credit\_ranking}(x, y) \end{aligned}$$

A simple rewriting (e.g., “age”  $\Rightarrow$  “Current\_year” minus “birth\_year”, “graduate\_student”  $\Rightarrow$  “student”, and “credit”  $\Rightarrow$  “credit\_ranking”) based on the mappings from the source  $C1$  to the target  $C2$  will get:

$$\forall x, y \text{ Applicant}(x) \wedge \text{student}(x, \text{"Y"}) \wedge \text{birth\_year}(x, y) \wedge (\text{Current\_year} - y > 30) \rightarrow \text{credit\_ranking}(x, \text{"Good"})$$

Careful readers will find that this rule for  $C2$  based on the simple rewriting is not necessarily correct nor semantically equivalent to the original rule in  $C1$  because that an applicant is a student does not necessarily mean that he or she is a graduate student. However, if  $C2$  uses “PhD\_student”, “MS\_student” and “undergraduate\_student” to describe their student applicants, the mappings are:

$$\begin{aligned} \forall x \text{ grad\_student}(x, \text{"Y"}) &\leftarrow \text{PhD\_student}(x, \text{"Y"}) \\ \forall x \text{ grad\_student}(x, \text{"Y"}) &\leftarrow \text{MS\_student}(x, \text{"Y"}) \end{aligned}$$

And the translated rules will be:

$$\forall x, y \text{ Applicant}(x) \wedge \text{PhD\_student}(x, \text{"Y"}) \wedge \text{birth\_year}(x, y) \wedge (\text{Current\_year} - y > 30) \rightarrow \text{credit\_ranking}(x, \text{"Good"})$$

$$\forall x, y \text{ Applicant}(x) \wedge \text{MS\_student}(x, \text{"Y"}) \wedge \text{birth\_year}(x, y) \wedge (\text{Current\_year} - y > 30) \rightarrow \text{credit\_ranking}(x, \text{"Good"})$$

Both translated rules are correct. Therefore, given our formal definition for the correctness of knowledge translation, translation exhibits certain asymmetries that one must be aware of. The translation of condition is different from the translation of conclusion. Assume mined IF-THEN rules are in the form of  $L \rightarrow R$  where  $L$  is the condition (left side) as the conjunctions of literals and  $R$  is the conclusion (right side) as the conjunctions of literals. Consider a general



IF-THEN rule:

$$\forall x_1, x_2 \dots P_1(X) \wedge \dots \wedge P_n(X) \rightarrow Q_1(X) \wedge \dots \wedge Q_m(X)$$

where  $X$  is the set of quantified variables  $x_1, x_2 \dots$  and constants, the translation of the condition  $(P_1(X) \wedge \dots \wedge P_n(X))$  is not the same process as the translation of the conclusion  $(Q_1(X) \wedge \dots \wedge Q_m(X))$ . We will subscript the symbol  $\rightsquigarrow$  with a “L” to indicate the condition translation ( $\rightsquigarrow_L$ ), and with a “R” to indicate the conclusion translation ( $\rightsquigarrow_R$ ) in the rest of the paper.

If we transform the IF-THEN rule to the conjunctive normal form (CNF), it becomes:

$$\forall x_1, x_2 \dots \neg P_1(X) \vee \dots \vee \neg P_n(X) \vee (Q_1(X) \wedge \dots \wedge Q_m(X))$$

Instead (not surprisingly), negation ends up involving the same asymmetry as the condition and conclusion translations. Assume that  $R$  is an expression which can be derived from  $\Sigma$  and  $\neg P$  by inference. Using the deduction theorem in first-order logic and considering that  $\neg P \rightarrow R$  is equivalent to  $\neg R \rightarrow P$ , we know that

$$(\Sigma; \neg P) \vdash R \Leftrightarrow \Sigma \vdash (\neg P \rightarrow R) \Leftrightarrow \Sigma \vdash (\neg R \rightarrow P) \Leftrightarrow (\Sigma; \neg R) \vdash P$$

This gives us a way to translate negations. We can think of  $P$  as a “ground condition” ( $\theta(P) = P$ ): Given  $P$ , try to find a  $P'$ , which satisfies  $(\Sigma; P') \vdash P$ . But this is just the problem of translating the condition  $P$ :  $(\Sigma; P) \rightsquigarrow_L P'$ .

Therefore, if the condition translation of  $P$  is  $P'$ ,  $\neg P'$  can be derived from  $\Sigma$  and  $\neg P$  by the conclusion translation and vice versa:

$$\begin{aligned} (\Sigma; P) \rightsquigarrow_L P' &\Rightarrow (\Sigma; \neg P) \rightsquigarrow_R \neg P' \\ (\Sigma; P) \rightsquigarrow_R P' &\Rightarrow (\Sigma; \neg P) \rightsquigarrow_L \neg P' \end{aligned}$$

A similar discussion and a proof on asymmetry of translation are in [\[11,12\]](#).

### 4.3 Design and Implementation

To address the asymmetry of translation for rule-based knowledge, we extended our open source inference engine, OntoEngine [\[13\]](#), to conduct both condition translation ( $\rightsquigarrow_L$ ) and conclusion translation ( $\rightsquigarrow_R$ ) for IF-THEN rules. The basic idea for the translation is:

For each IF-THEN rule in the source, we conduct the condition translation using backward chaining with generalized modus ponens and conduct the conclusion translation using forward chaining with generalized modus ponens. Then we combine the results from backward chaining and forward chaining to a new translated rule in the target. Our contribution is to design a method by combing both backward and forward chaining to address the asymmetry of translation for data mining rules. The knowledge translation algorithm is described in Algorithm [1](#). The generalized modus ponens is a well known sound

inference procedure. We will illustrate the detail of how our method works with the case studies in Section 5.

OntoEngine is a free downloadable inference engine from SemWebCentral.org. It mainly handles the translation of Semantic Web data (RDF assertions) by forward chaining and the translation of Semantic Web queries by backward chaining. To translate the data mining rules (e.g., decision tree rules and association rules), the major extension we have made in the implementation is to make OntoEngine be able to handle the translation of numeric comparisons (i.e., “>”, “<”, “=”, “>=” and “<=”) as binary predicates and the inference with arithmetic operators (i.e., “+”, “-”, “×” and “÷”) which are not trivial for reasoning.

---

**Algorithm 1.** Knowledge Translation (KT)

---

**Input:** : Rule  $P_1(X) \wedge \dots \wedge P_n(X) \rightarrow Q_1(X) \wedge \dots \wedge Q_m(X)$  in the source ontology.

The mapping rules  $\Sigma$  between the source and target.

**Output:** : Rule in the target ontology:  $P'_1(X) \wedge \dots \wedge P'_u(X) \rightarrow Q'_1(X) \wedge \dots \wedge Q'_v(X)$

$P_T = \text{null}$

**for** all predicate  $P_i$ ,  $1 \leq i \leq n$  **do**

  Query  $P_t = \text{BackwardChaining}(P_i)$

**if**  $P_T == \text{null}$  **then**

$P_T = P_t$

**else**

**if**  $P_t != \text{null}$  **then**

$P_T = P_T \wedge P_t$

**end if**

**end if**

**end for**

$Q_T = \text{null}$

**for** all predicate  $Q_j$ ,  $1 \leq j \leq m$  **do**

  Fact  $Q_t = \text{ForwardChaining}(Q_j)$

**if**  $Q_T == \text{null}$  **then**

$Q_T = Q_t$

**else**

**if**  $Q_t != \text{null}$  **then**

$Q_T = Q_T \wedge Q_t$

**end if**

**end if**

**end for**

**if**  $P_T != \text{null}$  and  $Q_T != \text{null}$  **then**

$P'_1(X) \wedge \dots \wedge P'_u(X) \Leftarrow P_T$

$Q'_1(X) \wedge \dots \wedge Q'_v(X) \Leftarrow Q_T$

**end if**

---

**Function** BackwardChaining (Query  $Q$ )Query  $Q_r = \text{null}$ **if**  $Q$ 's predicate is in the target ontology **then** $Q_r = Q$ **else****while**  $\exists M (\forall x_1 \dots x_k, Pm_1 \wedge \dots \wedge Pm_i \dots \wedge Pm_w \rightarrow \exists z_1 \dots z_l, Qm_1 \wedge \dots \wedge Qm_j \dots \wedge Qm_r)$  in  $\Sigma$ ,  $Q$ 's predicate is the same as  $Qm_j$  in  $M$  **do**New Query  $Q_N = \text{ModusPonens}(Q, M)$  $Q_r = \text{BackwardChaining}(Q_N)$ **end while****end if**Return  $Q_r$ **Function** ForwardChaining (Fact  $F$ )Fact  $F_r = \text{null}$ **if**  $F$ 's predicate is in the target ontology **then** $F_r = F$ **else****while**  $\exists M (\forall x_1 \dots x_k, Pm_1 \wedge \dots \wedge Pm_i \dots \wedge Pm_w \rightarrow \exists z_1 \dots z_l, Qm_1 \wedge \dots \wedge Qm_j \dots \wedge Qm_r)$  in  $\Sigma$ ,  $F$ 's predicate is the same as  $Pm_i$  in  $M$  **do**New Fact  $F_N = \text{ModusPonens}(F, M)$  $F_r = \text{ForwardChaining}(F_N)$ **end while****end if**Return  $F_r$ **Function** ModusPonens (Object  $O$ , Mapping  $M$ )Object  $O_r = \text{null}$ **if**  $O$  is a Query **then**Query  $Q_r = O$ ;  $Substitutions = \{ \}$ **if**  $Q_r$  is  $Qm_j(?x_j, ?y_j)$  and one predicate in the conclusion of  $M$  is  $Qm_j(x_j, y_j)$  **then** $Substitutions = Substitutions + \{x_j/?x_j, y_j/?y_j\}$ **end if****if**  $Substitutions$  is not empty **then** $O_r = \text{Substitute the variables in the condition (i.e., } Pm_1 \wedge \dots \wedge Pm_i \dots \wedge Pm_w)$  of  $M$ .**end if****end if****if**  $Q$  is a Fact **then**Fact  $F_r = O$ ;  $Substitutions = \{ \}$ **if**  $F_r$  is  $Pm_i(x_i, y_i)$  and one predicate in the condition of  $M$  is  $Pm_i(?x_i, ?y_i)$  **then** $Substitutions = Substitutions + \{?x_i/x_i, ?y_i/y_i\}$ **end if****if**  $Substitutions$  is not empty **then** $O_r = \text{Substitute the variables in the conclusion (i.e., } Qm_1 \wedge \dots \wedge Qm_j \dots \wedge Qm_r)$  of  $M$ .**end if****end if**Return  $O_r$

## 5 Case Studies

### 5.1 Translation of NBA Classification Rules

In the first case study, which we focused on the translation of decision tree rules, we first extracted data from two popular sports web sites about NBA: the NBA official site<sup>1</sup> and the Yahoo Sports NBA site<sup>2</sup> and put them into two databases which we called *NBA* and *NBAYahoo*. We then generated two ontologies by transforming relations to OWL classes and attributes to data type properties. Therefore the decision tree rules mined from each database can be represented in OWL and SWRL.

Since both sites collect the data about the same specific domain (i.e., NBA), the data are highly overlapping. However, these two databases use different but semantically related attributes to describe NBA players and teams. For example, in *NBA* the unit of player height is meter but in *NBAYahoo* it is foot. *NBA* uses “position” but *NBAYahoo* uses “playerposition” to describe the positions of players. Therefore the mappings between *NBA* and *NBAYahoo* look like:

- 1  $\forall x @NBA:Player(x) \leftrightarrow @NBAYahoo:Player(x)$
- 2  $\forall x, y @NBA:height(x, y) \rightarrow @NBAYahoo:height(x, y/0.3048)$
- 3  $\forall x, y @NBAYahoo:height(x, y) \rightarrow @NBA:height(x, y * 0.3048)$
- 4  $\forall x, y @NBA:weight(x, y) \leftrightarrow @NBAYahoo:weight(x, y)$
- 5  $\forall x, y @NBA:position(x, y) \leftrightarrow @NBAYahoo:playerposition(x, y)$

where we use “@NBA:” and “@NBAYahoo:” as namespaces to distinguish the concepts which are from two databases or ontologies but have the same names (e.g., “Player”, “height”, and “weight”).

We ran the C4.5 decision tree learning algorithm (J48 in WEKA [22]) in the *NBA* database and got 11 rules with high accuracy to classify the positions of players based on the “player” table or to classify whether a team will play in playoff based on the “team” table. The overall accuracy of the 11 classification rules is 85.7% (342/399). To show the translation process, we take one example:

$$\forall x, y, z @NBA:Player(x) \wedge Float(y) \wedge Float(z) \wedge @NBA:height(x, y) \wedge @NBA:weight(x, z) \wedge (y < 1.96) \wedge (z \leq 218) \rightarrow @NBA:position(x, 'Guard')$$

This rule means if a player’s height is less than 1.96 meters and weight is less than or equal to 218 pounds, the player is very likely to have the position as “Guard.” The accuracy of this rule is 96.3%. To translate this rule, the first step is conducting the condition translation ( $\rightsquigarrow_L$ ) as backward chaining from *NBA* to *NBAYahoo*. Note,

$$@NBA:Player(x) \wedge Float(y) \wedge Float(z) \wedge @NBA:height(x, y) \wedge @NBA:weight(x, z) \wedge (y < 1.96) \wedge (z \leq 218)$$

<sup>1</sup> <http://www.nba.com>

<sup>2</sup> <http://sports.yahoo.com/nba>

is the condition of the rule in the *NBA* ontology. Our inference engine does the backward chaining with mapping rule 1, 3, and 4 and generates the condition of the rule in the *NBAYahoo* ontology:

$$\begin{aligned} & @NBAYahoo:Player(x) \wedge Float(y) \wedge Float(z) \wedge @NBAYahoo:height(x, y) \\ & \wedge @NBAYahoo:weight(x, z) \wedge (y < 6.42) \wedge (z \leq 218) \end{aligned}$$

Similarly, our inference engine conducts the conclusion translation ( $\rightsquigarrow_R$ ) as forward chaining from *NBA* to *NBAYahoo*. The conclusion of the rule (i.e.,  $@NBA:position(x, 'Guard')$ ) is translated to  $@NBAYahoo:playerposition(x, 'Guard')$  with mapping rule 5. Then the inference engine combines the results from both the condition translation and conclusion translation with quantifiers:

$$\begin{aligned} \forall x, y, z @NBAYahoo:Player(x) \wedge Float(y) \wedge Float(z) \wedge @NBAYahoo:height(x, y) \wedge \\ @NBAYahoo:weight(x, z) \wedge (y < 6.42) \wedge (z \leq 218) \\ \rightarrow @NBAYahoo:playerposition(x, 'Guard') \end{aligned}$$

This is the translated classification rule in *NBAYahoo*. We tested the accuracy of the translated rule in the *NBAYahoo* data and got a 97.2% accuracy. We ran J48 in WEKA in the *NBAYahoo* data directly, the most similar rule to the above translated rule is

$$\begin{aligned} \forall x, y, z @NBAYahoo:Player(x) \wedge Float(y) \wedge Float(z) \wedge @NBAYahoo:height(x, y) \wedge \\ @NBAYahoo:weight(x, z) \wedge (y < 6.42) \wedge (z \leq 213) \\ \rightarrow @NBAYahoo:playerposition(x, 'Guard') \end{aligned}$$

in which only the splitting point of weight is different (213 vs. 218) and this rule has a 98.5% accuracy. Figure 1 shows the accuracy of all 11 rules mined from *NBA*, 11 translated rules to *NBAYahoo* and 11 most similar rules mined directly from *NBAYahoo*. For most cases translated rules are as accurate as rules mined directly from *NBAYahoo*. Rule 1-7 are about “position” of “player” and rules 8-11 are about “playoff” for “team”.

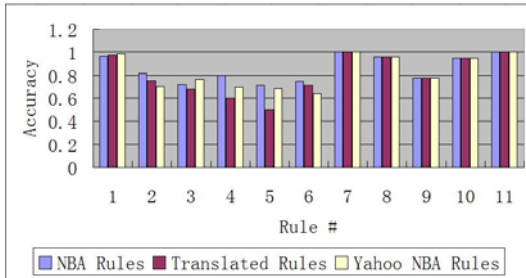


Fig. 1. Accuracy of Classification Rules of the NBA Domain

We tested all 11 translated rules in the *NBAYahoo* data. The overall accuracy is 82.1% (431/525). Compared with the overall accuracy of rules we mined directly from the *NBAYahoo* data as 80.9% (425/525), translated rules have similar overall accuracy as directly mined rules.

We also tested the scalability of our knowledge translation system. For all “player”, “team” and “scores” data tables, we selected different attributes as classification labels and got a large number of classification rules using WEKA. Although many of them are not meaningful or with low accuracy, they are useful for the scalability test. Figure 2 shows that when the number of the rules increases, the processing time of the translation process increases linearly. The testing process was performed on a regular PC with an AMD Athlon Dual-Core Processor 1.90 GHz and 4.00GB memory.

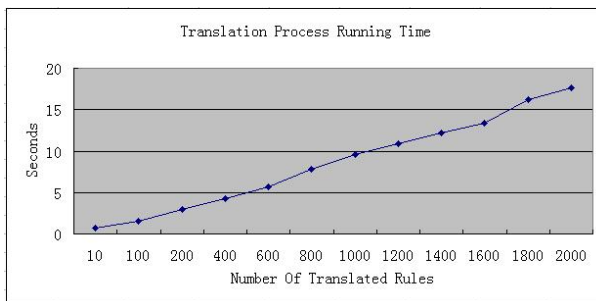


Fig. 2. Scalability of Classification Rule Translation

## 5.2 Translation of ZFIN and MGI Gene Association Rules

In the second case study, which we focused on the translation of association rules, we obtained the data and schemas from NIH model organisms which support different online gene databases, such as ZFIN<sup>3</sup> (the zebrafish gene) and MGI<sup>4</sup> (the mouse gene). Genetic researchers normally gather knowledge across different species because the genes from different species are potentially related to each other although their data are not overlapping. Comparing the patterns mined from different gene databases are meaningful to domain experts. However, different gene databases use different table names and attributes which make the comparison hard. For both gene databases we tried association rule mining. Then we translated association rules from MGI to the ZFIN schema (ontology) and compared them with the rules mined from ZFIN directly. Domain experts helped us to specify some mappings between MGI and ZFIN so that we could process the translation. The motivation for knowledge translation is that domain experts can compare the association rules from different databases involving shared concepts.

<sup>3</sup> <http://www.zfin.org>

<sup>4</sup> <http://www.informatics.jax.org>

In this case study, we mainly focused on MGI’s “Marker\_list” and ZFIN’s “Marker” tables and their attributes. These two tables both describe the information about gene expressions. We got 20 meaningful association rules from the MGI data by using Apriori in WEKA and successfully translated 10 of them to ZFIN. It means 10 of 20 meaningful rules from MGI have corresponding rules as a subset of all rules mined from ZFIN directly. The reason why the rest 10 rules were not translated to ZFIN is that one attribute (i.e, MGI’s “cytogeneticOffset”) has no counterpart in the ZFIN database. This is because that the ZFIN group did not collect these information.

For the 10 rules related to MGI’s “chromosome” and “cM\_position” that indicate the position of the marker on the chromosome in mouse genes, it is straightforward to translate them to ZFIN in the similar way as our inference engine does for NBA classification rules. The original data for MGI’s “cM\_position” are numbers. We took one preprocessing step to categorize them by selecting some intervals based on its range. For example, one association rule mined from MGI,

$$\forall x @MGI:Marker\_list(x) \wedge @MGI:cM\_position(x, 16) \rightarrow @MGI:chromosome(x, '1')$$

means if the Marker position on the chromosome is 16, this corresponds to chromosome ‘1’ in the MGI database. To translate this rule, three mappings,

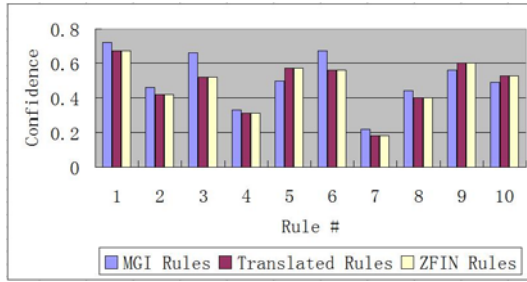
$$\begin{aligned} 6 \quad & \forall x, y @MGI:Market\_list(x) \leftrightarrow @ZFIN:Marker(x) \\ 7 \quad & \forall x, y @MGI:cM\_position(x, 16) \leftrightarrow @ZFIN:lg\_location(x, 20) \\ 8 \quad & \forall x, y @MGI:chromosome(x, '1') \leftrightarrow @ZFIN:or\_lg(x, '22') \end{aligned}$$

need to be applied. Note that the ZFIN group uses *or\_lg(x, ‘22’)* (linkage groups) rather than chromosome numbers to identify chromosomes. This is because unlike most species, it is hard to readily distinguish one zebrafish chromosome from another using visual techniques. Therefore one linkage group in a zebrafish model includes a set of potential chromosomes and is comparable to a chromosome in a mouse model. The “or” prefix stands for “Oregon” since there are many groups working on linkage groups and the ZFIN group is in Oregon.

The backward chaining for the condition with mapping rule 6 and 7 generates:  $@ZFIN:Marker(x) \wedge @ZFIN:lg\_location(x, 20)$  and the forward chaining for the conclusion with mapping rule 8 generates:  $@ZFIN:lg\_location(x, 20)$ . Finally the combination generates the translated rule:

$$\forall x @ZFIN:Marker(x) \wedge @ZFIN:lg\_location(x, 20) \rightarrow @ZFIN:or\_lg(x, '22')$$

The confidence of the original rule in MGI is 67% and the confidence of the translated rule in ZFIN is 56%. Considering the large size of data instances in ZFIN and MGI, although both confidences are not high, this translation and comparison actually shows that there are some similar patterns (associations) with similar confidences in both of the ZFIN and MGI data. From domain



**Fig. 3.** Confidence of Association Rules of the Gene Domain

experts’ view, this rule also shows there is a marker cluster at a specific location of linkage group 22. Figure 3 shows the confidences of all 10 rules mined from the MGI data, the translated rules from MGI to ZFIN, and the corresponding rules mined from the ZFIN data.

Since the association rule translation process is basically the same as the classification rule translation, we did not perform scalability tests in this case study. Instead, it is more interesting for us to discuss the situation where 10 out of 20 rules are not translatable because MGI’s “cytogeneticOffset” has no counterpart in the ZFIN database. For example, one association rule:

$$\forall x \ @MGI:Marker\_List(x) \wedge \ @MGI:Organism\_key(x, '1') \wedge \\ \ @MGI:cytogeneticOffset(x, 'p') \rightarrow \ @MGI:chromosome(x, '5')$$

means that if one “Marker” has an organism key as ‘1’ and cytogeneticOffset as ‘p’, the corresponding chromosome will be ‘5’. For MGI’s  $\_Organism\_key(x, '1')$  it should be translated to ZFIN’s  $organism(x, 'mouse')$  and MGI’s  $chromosome(x, '5')$  should be translated to ZFIN’s  $or\_lg(x, '24')$ . However for MGI’s  $cytogeneticOffset(x, y)$  we could not do any translation. Cytogenetics is a set of genetics which describes the structure and function of the chromosome. For the MGI group cytogenetics offset is a very interesting and important attribute while the ZFIN group does not study it because zebrafish genes are not as complex as mouse genes and also too small to collect cytogenetics offset data. But domain experts in ZFIN believe rules with this attribute are interesting and meaningful. The suggestion was to create a “cytogeneticOffset” attribute in ZFIN and the rest 10 rules can be translated successfully. For example, one translated rule is:

$$\forall x \ @ZFIN:Marker(x) \wedge \ @ZFIN:organism(x, 'mouse') \wedge \\ \ @ZFIN:cytogeneticOffset(x, 'p') \rightarrow \ @ZFIN:or\_lg(x, '24')$$

## 6 Discussion and Future Work

In our work, we have specified the mappings among data resources manually. Although it is a one-time job compared with potential applications for translating



large number of various data mining rules, mapping specification is still time consuming. Some automatic or semi-automatic mapping discovery tools, such as schema mapping [5,18] and ontology mapping tools [21], will be helpful. However, there must be some uncertainty with the automatically discovered mappings. It also happens for the mappings specified by human experts, because sometime it is hard to say what exact mappings are among attributes.

To handle mappings with uncertainty, one promising way is to extend Semantic Web ontologies with Markov logic [9], which combines first-order logic with Markov random fields, to represent knowledge and mappings as Markov logic networks (MLNs). The knowledge translation with uncertain mappings can be a process using both logic inference and probabilistic inference.

We also plan to apply knowledge translation algorithms for distributed data mining (DDM) systems in a client-server model. The clients will be data analysts and a DDM server will connect to local data resources. The data mining tasks will run on local resources. The output from local data resources is the mined knowledge based on local ontologies. Given the mappings between local data resources and the user site, the system will apply appropriate knowledge translation algorithms to first translate the knowledge to the user ontology, then the knowledge from multiple resources can be combined as if in the homogeneous scenario (i.e., the same user ontology).

## 7 Conclusions

Major contributions of this research to data mining and the Semantic Web are:

- It is novel to research how to translate the mined knowledge from one data resource to another semantically heterogeneous one. This work can be applied to knowledge transfer and potentially to distributed data mining in the semantically heterogeneous scenario.
- The formal representation of mined knowledge leverages the ideas of the Semantic Web to make the knowledge computer “understandable” and “translatable”. It is a key step to make the mined knowledge sharable among software agents.
- The general nature of our approach makes it applicable to any domain, especially to biomedical sciences, where large amounts of data are already publicly available from different labs but are semantically heterogeneous.

In our future work, we need to consider the uncertainty of mappings. We also plan to extend our knowledge translation methods to distributed data mining.

**Acknowledgment.** We thank Xiang Shao, Sridhar Ramachandran, and Tom Conlin in the ZFIN group for providing domain knowledge on genetic data, database mappings, and valuable comments.

## References

1. OWL Web Ontology Language, <http://www.w3.org/TR/owl-ref/>
2. Rule Interchange Format (RIF), <http://www.w3.org/2005/rules/>
3. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, <http://www.w3.org/Submission/SWRL/>
4. XLMiner: Data Mining in Excel, <http://www.resample.com/xlminer/index.shtml>
5. An, Y., Borgida, A., Miller, R.J., Mylopoulos, J.: A semantic approach to discovering schema mapping expressions. In: ICDE, pp. 206–215 (2007)
6. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American* 284(5) (May 2001)
7. Bruijn, J.D., Polleres, A.: Towards an Ontology Mapping Specification Language for the Semantic Web. Technical report, DERI (June 2004)
8. Caragea, D., Zhang, J., Bao, J., Pathak, J., Honavar, V.G.: Algorithms and software for collaborative discovery from autonomous, semantically heterogeneous, distributed information sources. In: Jain, S., Simon, H.U., Tomita, E. (eds.) ALT 2005. LNCS (LNAI), vol. 3734, pp. 13–44. Springer, Heidelberg (2005)
9. Domingos, P., Lowd, D.: *Markov Logic: An Interface Layer for AI*. Morgan & Claypool, CA (2009)
10. Dou, D., Frishkoff, G., Rong, J., Frank, R., Malony, A., Tucker, D.: Development of NeuroElectroMagnetic Ontologies (NEMO): A Framework for Mining Brainwave Ontologies.. In: KDD, pp. 270–279 (2007)
11. Dou, D., McDermott, D.: Deriving Axioms Across Ontologies. In: AAMAS, pp. 952–954 (2006)
12. Dou, D., McDermott, D.: Towards theory translation. In: Baldoni, M., Endriss, U. (eds.) DALT 2006. LNCS (LNAI), vol. 4327, pp. 16–28. Springer, Heidelberg (2006)
13. Dou, D., McDermott, D.V., Qi, P.: Ontology Translation on the Semantic Web. *Journal on Data Semantics* 2, 35–57 (2004)
14. Eaton, E.: Multi-resolution learning for knowledge transfer. In: AAAI (2006)
15. Gao, J., Fan, W., Jiang, J., Han, J.: Knowledge transfer via multiple model local structure mapping. In: KDD, pp. 283–291 (2008)
16. Gupta, R., Ratinov, L.-A.: Text categorization with knowledge transfer from heterogeneous data sources. In: AAAI, pp. 842–847 (2008)
17. Liu, H., Dou, D.: An Exploration of Understanding Heterogeneity through Data Mining. In: Proceedings of KDD 2008 Workshop on Mining Multiple Information Sources, pp. 18–25 (2008)
18. Miller, R.J., Hernández, M.A., Haas, L.M., Yan, L.-L., Ho, C.T.H., Fagin, R., Popa, L.: The clio project: Managing heterogeneity. *SIGMOD Record* 30(1), 78–83 (2001)
19. Park, B.-H., Kargupta, H.: Distributed data mining: Algorithms, systems, and applications. In: Ye, N. (ed.) *Data Mining Handbook* (2002)
20. Provost, F.J., Buchanan, B.G.: Inductive policy: The pragmatics of bias selection. *Machine Learning* 20(1-2), 35–61 (1995)
21. Qin, H., Dou, D., LePendu, P.: Discovering Executable Semantic Mappings Between Ontologies. In: ODBASE, pp. 832–849 (2007)
22. Witten, I.H., Frank, E.: *Data Mining: Practical machine learning tools and techniques*, 2nd edn. Morgan Kaufmann, San Francisco (2005)

# Mining Frequent Disjunctive Selection Queries

Inès Hilali-Jaghdam<sup>1,2</sup>, Tao-Yuan Jen<sup>1</sup>, Dominique Laurent<sup>1</sup>, and Sadok Ben Yahia<sup>2</sup>

<sup>1</sup> ETIS - CNRS - ENSEA, University of Cergy Pontoise, F-95000, France

<sup>2</sup> Computer Sc. Dept, Faculty of Sciences of Tunis, University Campus, 1060 Tunis, Tunisia  
{ines.hilali-jaghdam, jen, dlaurent}@u-cergy.fr,  
sadok.benyahia@fst.rnu.tn

**Abstract.** In this paper, we address the issue of mining frequent disjunctive selection queries in a given relational table. To do so, we introduce a level-wise algorithm to mine such queries whose selection condition is minimal. Then, based on these frequent minimal queries, and given any disjunctive selection query, we are able to decide whether its frequent or not. We carried out experiments on synthetic and real data sets that show encouraging results in terms of scalability.

**Keywords:** Level-wise algorithms, frequent query, disjunctive query mining.

## 1 Introduction

The extraction of frequent patterns from data sets has motivated many research efforts during the last two decades. However, the extraction of *disjunctive* queries grasped little attention despite the interest that applications could benefit from this kind of queries. As examples of such applications, we mention the different explanations of symptoms observed in an incorrect diagnosis, the ambiguities in the comprehension of the natural language, biological inheritance that consists in knowing if an offspring cell becomes predisposed to such or such characteristic of its parents, etc. In this paper, we study the mining of all minimal disjunctive frequent queries from a relational table  $\Delta$  defined over a set of attributes  $U$ . In addition, we show that these queries constitute a concise representation, in the sense that, given a disjunctive selection query, we are able to say whether it is frequent according to a support threshold.

The remainder of this paper is organized as follows. Section 2 discusses related work, and Section 3 states basic definitions and properties of disjunctive selection queries. Section 4 focuses on mining frequent minimal disjunctive selection queries, through a level-wise algorithm, called DISAPRIORI. Experimental results are reported in Section 5, and Section 6 concludes the paper and discusses future work.

## 2 Related Work

In the literature, a wealthy number of approaches were interested in the extraction of frequent objects based on conjunctive criteria. In the transactional database case, among well known approaches, we mention frequent itemsets [3], frequent closed patterns [4], minimal generator patterns [5], and frequent maximal patterns [6]. In the relational

database case, some pioneering work addressing the extraction of frequent conjunctive queries are [7–9]. In [7], the authors consider conjunctive projection-selection queries extracted from a relational table. The approach in [8] is an extension of [7], since [8] considers conjunctive projection-selection-join queries where joins are performed along keys and foreign keys in the presence of functional and inclusion dependencies in a star schema. In [9], the authors consider the general case of any relational database, from which frequent conjunctive projection-selection-join queries are mined. The main difference between [8] and [9] is that, in [8], functional and inclusion dependencies holding in a star schema are explicitly used to optimize the computations, whereas in [9], any database schema can be handled, but no dependency is taken into account.

On the other hand, the extraction of objects according to disjunctive criteria was mainly studied for transactional databases. In [10], inclusion-exclusion identities are introduced to derive disjunctive supports, based on conjunctive ones. In [11], a closure operator was presented in order to map many disjunctive itemset to a unique one called the disjunctive closed itemset. However, to the best of our knowledge, the issue of mining disjunctive queries from a relational table has not been addressed in the literature.

### 3 Disjunctive Selection Queries

In this section, we introduce the basic definitions and properties of our approach. We assume that the reader is familiar with standard notation of the relational model of databases (see [12]). We assume that we are given a relational table  $\Delta$  defined over the attribute set  $U$ , also called the universe. Every attribute  $A$  in  $U$  is associated with a set of values, called the domain of  $A$  and denoted by  $dom(A)$ . The notion of *disjunctive selection query* is defined as follows.

**Definition 1.** *Given a relational table  $\Delta$  defined over  $U$ , a disjunctive selection query is a relational query of the form  $\sigma_{(A_1=a_1) \vee \dots \vee (A_n=a_n)}(\Delta)$ , where  $n \geq 1$  and for every  $i$  in  $\{1, \dots, n\}$ ,  $A_i$  is in  $U$  and  $a_i$  in  $dom(A_i)$ .*

*Given a disjunctive selection query  $q$ , the support of  $q$  in  $\Delta$ , denoted by  $sup_{\Delta}(q)$ , or simply by  $sup(q)$  when  $\Delta$  is understood, is the cardinality of the answer to  $q$  in  $\Delta$ . Given a support threshold  $minsup$ ,  $q$  is said to be frequent in  $\Delta$ , if  $sup_{\Delta}(q) \geq minsup$ .*

**Example 1.** *Considering the relational table  $\Delta$  shown in Table 1 we have  $U = \{Cid, Cname, Caddr, Pid, Ptype, Qty\}$ , where attributes have the following intuitive meaning:*

- *Cid, Cname and Caddr represent respectively Identifier, Name, and Address of customers,*
- *Pid and Ptype represent respectively the Identifier and Type of products, and*
- *Qty represents the quantity sold for a given customer and a given product.*

*Let us consider the queries  $q_1 = \sigma_{(Cname=John)}(\Delta)$ ,  $q_2 = \sigma_{(Cid=C_2)}(\Delta)$  and  $q_3 = \sigma_{(Cname=John) \vee (Cid=C_2)}(\Delta)$ . These queries are clearly disjunctive selection queries, and it is easy to check from Table 1 that  $sup(q_1) = 2$ ,  $sup(q_2) = 2$  and  $sup(q_3) = 4$ . Thus, setting  $minsup$  equal to 3,  $q_3$  is frequent while  $q_1$  and  $q_2$  are not.*

Disjunctive selection queries are compared according to the following definition.

**Table 1.** The table  $\Delta$ 

Cid	Pid	Cname	Caddr	Ptype	Qty
$c_1$	$p_1$	John	Paris	milk	10
$c_1$	$p_2$	John	Paris	beer	10
$c_2$	$p_1$	Mary	Paris	milk	1
$c_2$	$p_3$	Mary	Paris	beer	5
$c_3$	$p_3$	Paul	NY	beer	10
$c_4$	$p_4$	Peter	Paris	milk	15

**Definition 2.** For all disjunctive selection queries  $q_1 = \sigma_{(A_{11}=a_{11}) \vee \dots \vee (A_{1n_1}=a_{1n_1})}(\Delta)$  and  $q_2 = \sigma_{(A_{21}=a_{21}) \vee \dots \vee (A_{2n_2}=a_{2n_2})}(\Delta)$ ,  $q_1$  is said to be a sub-query of  $q_2$  if for every  $i$  in  $\{1, \dots, n_1\}$ , there exists  $j$  in  $\{1, \dots, n_2\}$  such that  $A_{1i} = A_{2j}$  and  $a_{1i} = a_{2j}$ .

Given a support threshold  $minsup$ , the disjunctive selection query  $q = \sigma_{(A_1=a_1) \vee \dots \vee (A_n=a_n)}(\Delta)$  is said to be frequent minimal with respect to  $minsup$ , or frequent minimal when  $minsup$  is clear from the context, if  $q$  is frequent and either  $n = 1$  or no sub-query of  $q$  is frequent.

In the context of Example [1](#),  $q_1$  and  $q_2$  are the only sub-queries of  $q_3$ . Since  $q_1$  and  $q_2$  are not frequent while  $q_3$  is frequent,  $q_3$  is frequent minimal with respect to  $minsup$ .

It turns out that frequent minimal queries play a key role in our approach, since mining only these queries is enough to know all frequent disjunctive queries. This result is based on the following monotonicity property, whose easy proof is omitted.

**Proposition 1.** For all disjunctive queries  $q_1$  and  $q_2$ , if  $q_1$  is a sub-query of  $q_2$  then  $sup(q_1) \leq sup(q_2)$ .

As a consequence of Proposition [1](#), if  $q$  is not frequent then any sub-query  $q'$  of  $q$  is not frequent either. In other words, if  $q'$  is frequent and if  $q'$  is a sub-query of  $q$ , then  $q$  is frequent. This explains why all frequent disjunctive selection queries can be obtained from the minimal frequent selection queries.

It is however important to notice that, based on frequent minimal queries and their supports, it is not possible in general to compute the support of any frequent disjunctive selection query. Indeed, let  $q_1 = \sigma_{D_1}(\Delta)$  and  $q_2 = \sigma_{D_2}(\Delta)$  be two frequent minimal queries, and consider the disjunctive selection query  $q = \sigma_{D_1 \vee D_2}(\Delta)$ . Since the answer to  $q$  is the union of those to  $q_1$  and  $q_2$ , we have  $sup(q) = sup(q_1) + sup(q_2) - sup(\sigma_{D_1 \wedge D_2}(\Delta))$ . Thus, computing the support of  $q$  requires to know the support of  $\sigma_{D_1 \wedge D_2}(\Delta)$ , which is not a disjunctive selection query. In the next section, we show a particular case where the support of  $q$  can be inferred from those of  $q_1$  and  $q_2$ .

## 4 Mining Frequent Minimal Selection Queries

In this section, we show how to mine frequent minimal queries, using level-wise strategy as in Apriori ([3](#)). Our algorithm, called DISAPRIORI, proceeds as follows:

1. At level 1, the algorithm computes the supports of every query  $\sigma_{(A=a)}(\Delta)$  for every  $A$  in  $U$  and every  $a$  in  $dom(A)$  appearing in  $\Delta$ . For every  $q = \sigma_{(A=a)}(\Delta)$ , if

**Algorithm 1.** DISAPRIORI Algorithm

---

**Data:** The table  $\Delta$  and a support threshold  $minsup$

**Results:** The set  $Freq$  of all frequent minimal queries

$$F_1 = \{q \mid q = \sigma_{(A=a)}(\Delta) \text{ and } sup(q) \geq minsup\}$$

$$\neg F_1 = \{q \mid q = \sigma_{(A=a)}(\Delta) \text{ and } sup(q) < minsup\}$$

**For**  $k = 2, \neg F_{k-1} \neq \emptyset, k++$  **do**

$D_k = \text{DisApriori-Gen}(\neg F_{k-1})$

**Foreach**  $t \in \Delta$  **do**

**Foreach**  $D \in D_k$  **do**

/\*  $D = (A_1 = a_1) \vee \dots \vee (A_k = a_k)$  \*/ \*/

**If**  $\exists j \in \{1, \dots, k\}$  such that  $t.A_j = a_j$  **then**

$supp(\sigma_D(\Delta))++$

**End**

**end**

**end**

$F_k = \{\sigma_D(\Delta) \mid D \in D_k \text{ and } sup(\sigma_D(\Delta)) \geq minsup\}$

$\neg F_k = \{\sigma_D(\Delta) \mid D \in D_k \text{ and } sup(\sigma_D(\Delta)) < minsup\}$

**end**

**return**  $Freq = \bigcup_k F_k$  and  $\neg Freq = \bigcup_k \neg F_k$

---

$sup(q) < minsup$  then  $q$  and its support are stored in a set denoted by  $\neg F_1$ . Otherwise,  $q$  is a frequent minimal query, in which case it is stored, along with its support, in another set denoted by  $F_1$ .

2. At any level  $i > 1$ , the candidate disjunctions  $D_k$  are generated from those in the queries of  $\neg F_{k-1}$  to form disjunctions of size  $k$ . The corresponding disjunctive selection queries are pruned using Proposition 1; that is, if a sub-query of a query  $q$  appears to be in  $F_{k-1}$ , then  $q$  is frequent but not minimal. As such a query has not to be processed, its disjunction is removed from the candidate set  $D_k$ .
3. For every disjunction  $D$  remaining in  $D_k$ , the support of  $q = \sigma_D(\Delta)$  is computed. If  $sup(q) < minsup$  then  $q$  and its support is put in  $\neg F_k$ , otherwise,  $q$  is frequent minimal, and so, it is put along with its support in  $F_k$ .

The processing above is iterated while  $D_k$  is not empty. In the end of the processing, we know (i) all non frequent disjunctive queries and their supports, and (ii) all frequent minimal queries and their supports. Algorithm 1 gives full details on the steps mentioned above, except for step 2, which generates and prunes candidate disjunctions in exactly the same way as candidate itemsets are generated and pruned in Apriori ([3]).

We recall that, based on the output of Algorithm 1, given a disjunctive selection condition  $q$  is frequent if and only if  $Freq$  contains a query  $q'$  such that  $q'$  is a sub-query of  $q$ . Moreover, if the disjunction in  $q$  involves only one attribute, then  $sup(q)$  can be deduced from those output by Algorithm 1.

Indeed, let  $q = \sigma_D(\Delta)$  where  $D = (A = a_1) \vee \dots \vee (A = a_n)$ . If  $Freq$  contains no query  $q'$  such that  $q'$  is a sub-query of  $q$ , then  $q$  is not frequent and so,  $q$  is stored in  $\neg Freq$  along with its support. Otherwise, we proceed by induction on  $n$  as follows:

- The case  $n = 1$  is trivial, since the supports of all queries  $\sigma_{(A=a_1)}(\Delta)$  are stored in the output of Algorithm 1.

- Assume now that for all  $k \leq n$ , the supports of all queries of the form  $\sigma_{D_k}(\Delta)$ , where  $D_k = (A = a_1) \vee \dots \vee (A = a_k)$ , are known and let  $q = \sigma_D(\Delta)$ , where  $D = (A = a_1) \vee \dots \vee (A = a_n) \vee (A = a_{n+1})$ . In this case, denoting by  $D_n$  the selection condition  $(A = a_1) \vee \dots \vee (A = a_n)$ , we first notice that if  $(A = a_{n+1})$  occurs in  $D_n$ , then  $D$  is equivalent to  $D_n$  and so,  $\text{sup}(q) = \text{sup}(\sigma_{D_n}(\Delta))$ . Otherwise, we have  $\text{sup}(q) = \text{sup}(\sigma_{D_n}(\Delta)) + \text{sup}(\sigma_{(A=a_{n+1})}(\Delta)) - \text{sup}(\sigma_{D_n \wedge (A=a_{n+1})}(\Delta))$ , and we know that the last support is 0 since, for any  $i = 1, \dots, n$ ,  $\Delta$  contains no tuple  $t$  such that  $t.A = a_i$  and  $t.A = a_{n+1}$ . Therefore, we obtain that  $\text{sup}(q) = \text{sup}(\sigma_{D_n}(\Delta)) + \text{sup}(\sigma_{(A=a_{n+1})}(\Delta))$ .

However, it is unknown to the authors if there exist further cases where the supports of disjunctive selection queries can be deduced from the output of Algorithm [1](#).

## 5 Experiments

Algorithm [1](#) has been implemented in C++ and run on a computer with 3 Go main memory running on Linux Ubuntu.

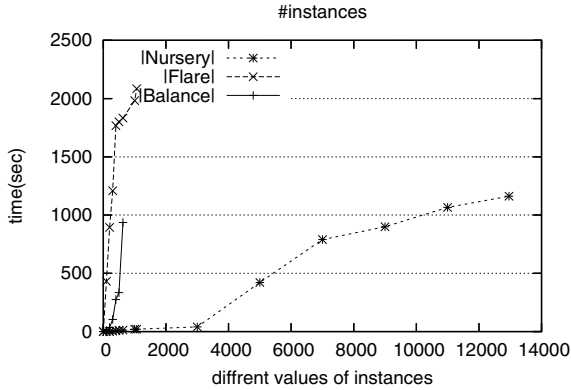
**Table 2.** The database characteristics of the first experiments

Base	$ \Delta $	# Attributes
Nursery	12960	8
Solar Flare	1066	10
Balance Scale	625	4

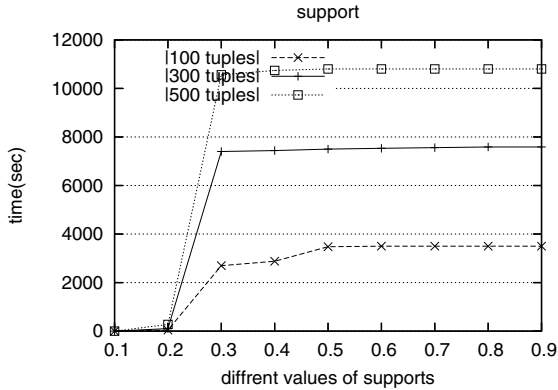
Our first experiments use benchmark databases from <http://archive.ics.uci.edu/ml/datasets.html> whose characteristics are given in Table [2](#). The experiments on these databases report on the runtime over the size of the table  $\Delta$ , which we denote by  $|\Delta|$ , in the case where the minimum support threshold is fixed to  $0.5 \times |\Delta|$ . To do so, we selected increasing numbers of tuples from the underlying table  $\Delta$  in order to plot the evolution of runtime with respect to the size of  $\Delta$ .

As can be seen from Figure [1](#) runtime increases exponentially with respect to the size of  $\Delta$ , except for *Nursery*. This is due to the fact that, in this data set, few values per attribute are present in the table, thus implying that the number of processed queries remains quite low, as compared to the other data sets. These experiments clearly show that runtime increases when the size of the table increases. In fact, due to the particular form of disjunctive selection queries considered in this paper, runtime increases with respect to the overall number of values present in the table  $\Delta$ , whatever their attribute domain.

Our second experiments report on the runtime spent with respect to different values of the minimum support threshold, in the cases where the size of the table is set to 100, 300 and 500. To do this, we used data sets generated by our own data generator, adapted from the one by IBM ([www.almaden.ibm.com](http://www.almaden.ibm.com)). Tables with 3 attributes



**Fig. 1.** Runtime over the size of *Nursery*, *Solar Flare* and *Balance* databases



**Fig. 2.** Runtime over the minimum support threshold value

and random values over these attributes have been generated. Experiments were run on these tables for support thresholds equal to  $\rho \times |T|$ , where  $\rho$  ranges from 0.1 up to 0.9.

Figure 2 shows so the behavior of runtime with respect to these different values of the minimum support threshold. It can be seen that, in all cases, runtime becomes roughly stable, reaching its maximum value, for  $\rho = 0.3$ . This is due to the fact that the supports of most frequent minimal queries are less than  $0.3 \times |\Delta|$ , and so, in this case, increasing the support does not imply any significant runtime increase. More generally, as compared to the case of mining frequent *conjunctive* queries (see [8]), we observe a totally different behavior of runtime with respect to support: in the conjunctive case, runtime decreases with the support, whereas in the disjunctive case, runtime increases with the support. This is so because in the disjunctive case, the higher the support, the more unfrequent disjunctions.



## 6 Conclusion and Further Work

In this paper, we have addressed the problem of mining frequent minimal disjunctive selection queries from a relational table, using a level-wise strategy, and we have shown that given any disjunctive selection query, the output of our algorithm allows to determine whether this query is frequent or not. Experiments on real and synthetic data sets show that our approach is realistic in terms of runtime.

As future work, we will consider constraints, such as functional dependencies for optimizing the computation of frequent disjunctive queries. We also plan to extend our approach from selection to projection-selection-join queries.

## References

1. Weiss, G.M., Zadrozny, B., Saar-Tsechansky, M.: Guest editorial: special issue on utility-based data mining. *Data Mining and Knowledge Discovery* 17(2), 129–135 (2008)
2. Nambiar, U.: Supporting Imprecision in Database Systems. In: Wang, J. (ed.) *Encyclopedia of Data Warehousing and Mining*, pp. 1884–1887 (2009)
3. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: *Advances in Knowledge Discovery and Data Mining*, pp. 307–328 (1996)
4. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. *Information Systems* 24(1), 25–46 (1999)
5. Bastide, Y., Pasquier, N., Taouil, R., Stumme, G., Lakhal, L.: Mining Minimal Non-redundant Association Rules Using Frequent Closed Itemsets. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) *CL 2000. LNCS (LNAI)*, vol. 1861, pp. 972–986. Springer, Heidelberg (2000)
6. Gouda, K., Zaki, M.J.: Efficiently mining maximal frequent itemsets. In: *IEEE International Conference on Data Mining*, pp. 163–170. IEEE Computer Society, Los Alamitos (2001)
7. Jen, T.Y., Laurent, D., Spyratos, N.: Mining all frequent projection-selection queries from a relational table. In: *Int. Conference on Extending Database Technology*, pp. 368–379. ACM Press, New York (2008)
8. Jen, T.Y., Laurent, D., Spyratos, N.: Mining frequent conjunctive queries in star schemas. In: *Int. Database Engineering Applications Symposium*, pp. 97–108 (2009)
9. Goethals, B., Le Page, W., Mannila, H.: Mining association rules of simple conjunctive queries. In: *Int. Conference SIAM-SDM*, pp. 96–107 (2008)
10. Galambos, J., Simonelli, I.: *Bonferroni-type Inequalities with Applications*. Springer, Heidelberg (2000)
11. Hamrouni, T., Yahia, S.B., Nguifo, E.M.: Sweeping the disjunctive search space towards mining new exact concise representations for frequent itemsets. *Data and Knowledge Engineering* 68(10), 1091–1111 (2009)
12. Ullman, J.: *Principles of Databases and Knowledge-Base Systems*, vol. 1. Computer Science Press, Rockville (1988)

# A Temporal Data Mining Framework for Analyzing Longitudinal Data

Corrado Loglisci, Michelangelo Ceci, and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari  
via Orabona, 4 - 70125 Bari - Italy

**Abstract.** Longitudinal data consist of the repeated measurements of some variables which describe a process (or phenomenon) over time. They can be analyzed to unearth information on the dynamics of the process. In this paper we propose a temporal data mining framework to analyze these data and acquire knowledge, in the form of temporal patterns, on the events which can frequently trigger particular stages of the dynamic process. The application to a biomedical scenario is addressed. The goal is to analyze biosignal data in order to discover patterns of events, expressed in terms of breathing and cardiovascular system time-annotated disorders, which may trigger particular stages of the human central nervous system during sleep.

## 1 Introduction

Domains of the real world that evolve over time, such as biomedical processes, human beings behaviours, physical and natural phenomena, can be described by a finite set of variables whose repeated measurement generates a particular class of multidimensional time-series known as *longitudinal data* [10]. Normally, longitudinal data represent the complete evolution or dynamics of a process over time and therefore they can convey relevant information. However, the complexity of longitudinal data makes their interpretation difficult and resorting to automatic techniques of analysis becomes necessary. Traditionally, most of attention has been paid by the classical computational statistics techniques, which anyway can suffer from problems coming from the high dimensionality of the collected data, from heterogeneity of data types and from the need of handling the intrinsic temporal nature of longitudinal data.

In this regard, a relevant role can be played by data mining approaches. One of the first proposed methods is the querying and mining system described in [3] where the authors investigated three different tasks for temporal association rules discovery, namely the discovery of valid time periods during which association rules hold, the discovery of possible periodicities that association rules have, and the discovery of association rules with temporal features, where the analyzed data are represented in the simplified representation of the transactions. Another interesting approach is presented in [8] which reports a methodology to pre-process time-series and discover frequent patterns from the pre-processing

results. In particular, the patterns are organized according to an hierarchical structure built on the temporal concepts of duration, coincidence and partial order.

In this paper, we propose a temporal data mining approach that aims at supporting the tasks of analyzing and interpreting the evolution of a dynamic process. It mines time-varying data and discovers patterns of time-annotated *complex events* which can trigger particular *stages* of the process. A complex event is associated to a variation or change while a stage corresponds to a specific state of the process which holds in a period of time. Two consecutive stages represent two different states and together they depict a transition in the process. So, given two consecutive stages, we assume that whatever happens in the first stage may affect the second one, and, since two consecutive stages are different each other, the events which occur in the first stage and do not occur in the second one can be responsible of the transition of the process towards the second stage. Therefore, the transition can be ascribed to these events.

Patterns are discovered from the events detected on a collection of pairwise stages of interest. Such a collection is properly created in order to consider only pairs of stages which depict similar transitions. The usage of pattern discovery is therefore addressed to find out the most frequent (and maybe significant) complex events which can determine similar transitions and, thus, can trigger analogous stages.

The paper is organized as follows. In next section we define the problem in terms of four sub-problems. The computational solution for them is described in Section 3. An application to the case of a biomedical scenario is presented in Section 4. Finally, conclusions are drawn.

## 2 Problem Formulation

Before formally defining the problem of interest, we introduce some necessary concepts. Let  $P : \{a_1, \dots, a_m\}$  be the finite set of real-valued variables (e.g., *blood oxygen*, *heart rate*, *respiration rate*), longitudinal data form a collection  $Mp$  of time-ordered measurements of the variables in  $P$ .

A stage  $S_j$  is a 4-tuple  $S_j = \langle ts_j, te_j, C_j, SV_j \rangle$ , where  $[ts_j..te_j]$  ( $ts_j, te_j \in \tau$ ,  $ts_j \leq te_j$ )<sup>1</sup> represents the time-period of the stage, while  $C_j : \{f_1, f_2, \dots\}$  is a finite set of *fluents*, namely facts or properties in terms of variables  $P$  that are true during the time-period  $[ts_j..te_j]$ .  $SV_j$  is the set  $\{sv_1, \dots, sv_k, \dots, sv_m\}$  containing  $m$  symbolic values such that  $sv_k$  is a high-level description of the parameters  $a_k \in P$  during  $[ts_j..te_j]$ .

An example of stage is  $S_1 : \langle t_1, t_{10}, \{ \textit{blood oxygen} \in [6500;6700], \textit{heart rate} \in [69;71], \textit{respiration rate} \in [2300;5500] \}, \{ \textit{blood oxygen is INCREASE}, \textit{heart rate is STEADY}, \textit{respiration rate is INCREASE} \} \rangle$  which can be interpreted as follows:  $S_1$  is associated with the period of time  $[t_1, t_{10}]$  and is characterized by the fact (fluent) that the variables *blood oxygen*, *heart rate* and *respiration rate*

<sup>1</sup>  $\tau$  is a finite totally ordered set of time-points. Henceforth, the corresponding order relation is denoted as  $\leq$ .

have values respectively in [6500; 6700], [69; 71], [2300; 5500] and have increasing, steady and increasing trend, respectively.

An event  $e$  is a signature  $e = \langle t_F, t_L, Ea, IEa, SEa \rangle$ , where  $[t_F..t_L]$  is the time-interval when event  $e$  occurs ( $t_F, t_L \in \tau$ ),  $Ea : \{ea_1, \dots, ea_k, \dots, ea_{m'}\}$  is a subset of  $P$  and contains  $m'$  distinct variables which take values in the intervals  $IEa : [inf_1, sup_1], \dots, [inf_k, sup_k], \dots, [inf_{m'}, sup_{m'}]$ , respectively, during  $[t_F..t_L]$ . Finally,  $SEa : \{sv_1, \dots, sv_k, \dots, sv_{m'}\}$  is a set of  $m'$  symbolic values associated to  $Ea$ . In particular,  $IEa$  is a quantitative description of the event, while  $SEa$  is a qualitative representation of the trend of values taken by each  $ea_k$  during  $[t_F..t_L]$ .

Two examples of events are  $e_1 : \langle t_1, t_5, \{bloodoxygen\}, \{[6300; 6800]\}, \{DECREASE\} \rangle$  and  $e_2 : \langle t_6, t_{10}, \{bloodoxygen\}, \{[6600; 7000]\}, \{INCREASE\} \rangle$  which can be interpreted as follows:  $e_1$  ( $e_2$ ) is associated with the time-period  $[t_1, t_5]$  ( $[t_6, t_{10}]$ ) during which the variables *blood oxygen* ranges in [6300; 6800] ([6600; 7000]) and has a decreasing (increasing) trend. Trivially, a sequence  $\langle e_1, e_2 \rangle$  is an ordered list of events when, given  $[t_{F1}..t_{L1}]$ ,  $[t_{F2}..t_{L2}]$  of  $e_1$  and  $e_2$  respectively,  $t_{F1}$  is the immediate predecessor of  $t_{F2}$  in  $\tau$ .

The notions above introduced suggest to resort to representation formalisms able to suitably handle the complex formulation of the events. Indeed, we resort to first-order logic formalism and approaches synthesized in *Inductive Logic Programming* (ILP) [9] which permit us to naturally deal with the intrinsic complexity of the longitudinal data and handle the structural and relational aspects of events and sequences as above defined. The events are modeled in a logical formalism (Datalog language [2]) and represented as *ground atoms*. A ground atom is an  $n$ -ary logic predicate symbol applied to  $n$  constant terms, while a non-ground atom is an  $n$ -ary predicate symbol applied to  $n$  constant and variable terms. For instance the sequence  $e_1, e_2$  before introduced is so represented:

```
sequence(seq1). event(seq1, e1). time_tF(e1, 1). time_tL(e1, 5). parameter_of(e1, p1). is_a(p1, blood_oxygen). value_interval(p1, '[6300;6800]'). symbolic_value(p1, 'DECREASE').
event(seq1, e2). time_tF(e2, 6). time_tL(e2, 10). parameter_of(e2, p2). is_a(p2, blood_oxygen). value_interval(p2, '[6600;7000]'). symbolic_value(p2, 'INCREASE').
```

where *sequence(seq1)* is the atom which identifies the sequence  $seq_1$  through the predicate *sequence()*; *event(seq1, e1)* is the atom which relates the sequence  $seq_1$  to the event  $e_1$  through *event()*; *time\_tF(e1, 1)* is the atom which assigns the specific value 1 to the attribute *time\_tF* of  $e_1$  through *time\_tF()*; *variable\_of(e1, p1)* is the atom which relates the event  $e_1$  to the variable  $p_1$  through *parameter\_of()*; *is\_a(p1, blood\_oxygen)* is the atom which assigns a specific value *blood\_oxygen* to  $p_1$  through *is\_a()*; *value\_interval(p1, '[6300;6800]')* is the atom which assigns a specific interval of values [6300;6800] to  $p_1$  through *value\_interval()* and *symbolic\_value(p1, 'DECREASE')* is the atom which assigns a specific symbolic value DECREASE to  $p_1$  through *symbolic\_value()*.

We can now formally define a temporal pattern: a temporal pattern  $T_P$  is a set of atoms  $p_0(t_0^1), p_1(t_1^1, t_1^2), p_2(t_2^1, t_2^2), \dots, p_r(t_r^1, t_r^2)$ , where  $p_0, p_i, i = 1, \dots, r$ , are logic predicate symbols while  $t_i^j$  are either constants or variables, which identify

sequences, events or variables in  $T_P$ . Among these logic predicates we can have predicates able to express possible temporal relationships between two events  $e_1$ ,  $e_2$  according to the Allen temporal logic [1]. For instance, the temporal pattern

*Tp: sequence(Q), event(Q, E1), event(Q, E2), before(E1, E2), parameter\_of(E1, P1), is\_a(P1, blood\_oxygen), value\_interval(P1, '[6300;7000]'), symbolic\_value(P1, steady), is\_a(P2, respiration\_rate), value\_interval(P2, '[2300;5500]'), symbolic\_value(P2, strong\_increase)*

expresses the fact that, for a subset of sequences, the event  $E_1$  is followed by  $E_2$ , where in  $E_1$  the blood oxygen has steady trend and ranges in [6300;7000] while in  $E_2$  the respiration rate is strongly increasing with values in [2300;5500].

Considering the concepts so far defined, the problem of interest in the proposed framework can be divided in four sub-problems formalised as follows:

1. *Given*: longitudinal data  $Mp : \{Mp_{t1}, Mp_{t2}, \dots, Mp_{tn}\}$ ; *Find*: a finite set  $S : \{S_1, S_2, \dots\}$  of consecutive stages which represent distinct sub-sequences of  $Mp$ .
2. *Given*: a criterion  $CS$  to collect pairwise stages of interest from  $S$ ; *Find*: a collection  $R$  of pairwise stages  $(S_j, S_{j+1})$  which satisfy the criterion  $CS$ .
3. *Given*: the collection  $R$ ; *Find*: a set  $ES$  of sequences  $\langle e_1, e_2, \dots \rangle$  of events for each pair  $(S_j, S_{j+1})$  in  $R$ .
4. *Given*: the set  $ES$  and a user-defined threshold  $minF$ ; *Find*: temporal patterns in  $ES$  whose support exceeds the threshold  $minF$ .

A computational solution to these sub-problems is described in Section 3.

## 3 Temporal Data Mining Framework

### 3.1 Determination of Stages

A stage can be seen as one of the steps of dynamics characterized by numerical ( $C_j$ ), symbolic ( $\{sv_1, \dots, sv_h, \dots, sv_m\}$ ) and temporal ( $[ts_j..te_j]$ ) properties. In other words, a stage corresponds to one of the distinct segments of  $Mp$ . The components  $ts_j, te_j, C_j$  are obtained by resorting to the method we proposed in [5] which is here shortly described. The periods of time  $[ts_j..te_j]$  are obtained by means of a two-stepped technique of temporal segmentation. In particular, it first identifies a series of change-points and recursively partitions  $Mp$  in a succession of multi-variate segments until the variability of each variable  $a_h$  does not exceed a user-defined threshold  $\omega$ . Then, it merges together consecutive segments if the variables in the segments are statistically correlated w.r.t. user-defined maximum threshold  $\rho$  of correlation. The duration  $[ts_j..te_j]$  of each stage is forced to be bigger than a user-defined minimum threshold  $minSD$ . This segmentation produces a sequence of segments of  $Mp$  that differ each other, and it guarantees that two consecutive segments have different fluents: given three consecutive segments,  $[ts_{j-1}..te_{j-1}]$ ,  $[ts_j..te_j]$ ,  $[ts_{j+1}..te_{j+1}]$ , the fluents  $C_j$  associated to  $[ts_j..te_j]$  are conditions which hold in  $[ts_j..te_j]$  but neither in the previous  $[ts_{j-1}..te_{j-1}]$  nor in the next  $[ts_{j+1}..te_{j+1}]$  segments. The generation

of fluents  $C_j$  is solved with the inductive logic programming approach used in [5] which permits to determine  $C_j$  as the set of interval-valued atomic formulae which *characterizes* the measurements included in  $[ts_j.. te_j]$  and *discriminates* them from those of  $[ts_{j-1}.. te_{j-1}]$  and  $[ts_{j+1}.. te_{j+1}]$ . This way, we can determine each stage of dynamics and distinguish it from each other with a rigorous description. Finally, the values of the elements  $SV_j$  of  $S_j$  are derived by means a function  $\Theta : \Pi \rightarrow \Lambda$  which provides an high-level representation  $\lambda \in \Lambda$  of the most relevant features  $\pi \in \Pi$  of data:  $\Theta$  returns, for each variable  $a_k$ , a representation of the slope of the regression line built on the values taken by  $a_k$  in the time interval  $[ts_j..te_j]$ . For instance, the slope values ranging in the interval  $(0.2, 1]$  are described as INCREASE.

### 3.2 Collection of Pairwise Stages

A collection  $R$  of pairwise stages is properly created in order to study the transition between similar pairs of stages through the discovery of the events which most frequently trigger analogous stages.

Pairwise stages appropriate for  $R$  are identified on the basis of a similarity value: pairs whose first stages and second stages have similarity value which exceeds a user-defined numerical threshold  $CS$  ( $CS \in [0; 100]$ ) are considered. For instance, two pairs  $(S_j, S_{j+1}), (S_k, S_{k+1})$  are collected in  $R$  if the similarity between  $S_j$  and  $S_k$  and the similarity between  $S_{j+1}$  and  $S_{k+1}$  exceeds  $CS$ . In this work the similarity between two stages  $S_j$  and  $S_k$  corresponds to the similarity between their fluents  $C_j, C_k$ <sup>2</sup> under the assumption that the symbolic values  $SV_j, SV_k$  are identical. Since the fluents are sets of interval-valued data (section 3.1), the similarity between  $C_j$  and  $C_k$  is so formulated:  $Sim(C_j, C_k) = (\sum_{f_j \in C_j, f_k \in C_k} (1 - Diss(f_j, f_k)) / (|C_j| * |C_k|)) * 100$ , where  $f_j(f_k)$

is a single interval-valued formula of  $C_j$  ( $C_k$ ). To compute  $Diss(f_j, f_k)$  we resort to dissimilarity functions specific for interval-valued data. In particular, we consider the Gowda and Diday's [4] dissimilarity measure defined as:  $Diss(f_j, f_k) = \sum_{h=1..|P|} \delta(f_{j_h}, f_{k_h})$ , where,  $f_{j_h}, f_{k_h}$  are the intervals assumed by

the parameter  $a_h$ ,  $|P|$  is number of intervals (variables), and  $\delta(f_{j_h}, f_{k_h})$  is obtained considering three types of dissimilarity measures incorporating different aspects of similarity, namely  $\delta(f_{j_h}, f_{k_h}) = \delta_\pi(f_{j_h}, f_{k_h}) + \delta_s(f_{j_h}, f_{k_h}) + \delta_c(f_{j_h}, f_{k_h})$ , ( $\delta_\pi, \delta_c, \delta_s \in [0, 1]$ ). It should be noted that several collections of similar transitions can be actually created from the pairs of stages in  $S$ : the resulting collection  $R$  is selected by the user in the set of possibly overlapping collections.

### 3.3 Detection of Complex Events

Once the collection  $R$  of pairwise stages has been identified, for each pair  $(S_j, S_{j+1})$  we look for events which may trigger the transition from  $S_j$  to  $S_{j+1}$ .

<sup>2</sup> The notion of similarity between two stages does not concern the time-periods  $[ts_j..te_j]$ , i.e., two stages can be similar although they are associated to different time-periods.

Events are detected by resorting to the method we proposed in [6] which permits us to exploit the assumption for which events occurring during the time interval  $[ts_j..te_j]$  should not occur in  $[ts_{j+1}..te_{j+1}]$ . The blueprint is to mine first candidate events then to select from these the events deemed *statistically interesting*. The algorithm for mining candidate events  $\{e \mid e = \langle t_F, t_L, Ea, IEa, SEa \rangle\}$  proceeds by iteratively scanning the measurements included in the stages  $S_j$  (i.e.,  $\{Mp_{ts_j}, \dots, Mp_{te_j}\}$ ) and  $S_{j+1}$  (i.e.,  $\{Mp_{ts_{j+1}}, \dots, Mp_{te_{j+1}}\}$ ) with two adjacent time-windows which slide back in time. The candidates are identified by finding variations in the measurements between the windows  $w$  and  $w'$ . At the first iteration, the time-windows  $w, w'$  ( $w'$  immediately follows  $w$ ) correspond to the last part of  $S_j$  and to the complete  $S_{j+1}$ , respectively. If a candidate is found then the next candidate is searched for the pair  $(w'', w)$ , where the new time-window  $w''$  has the same size of  $w$ . Otherwise, the next candidate is searched for the pair  $(w'', w')$ , where  $w''$  is strictly larger than  $w$ . At the end of a single scan a sequence of candidates is obtained.

The intuition underlying the detection of candidate events for a given couple of windows  $(w, w')$  is that the intrinsic dependence of two variables in  $P$  may change between the two adjacent time-windows. This idea is implemented in the following strategy: for each variable  $a_i$  two multiple linear regression models are built on the remaining variables in  $P$  by considering the distinct measurements in  $w$  and  $w'$  respectively:

$$\begin{aligned} a_i &= \beta'_0 + \beta'_1 a_1 + \dots + \beta'_{i-1} a_{i-1} + \beta'_{i+1} a_{i+1} + \dots + \beta'_m a_m, \\ a_i &= \beta''_0 + \beta''_1 a_1 + \dots + \beta''_{i-1} a_{i-1} + \beta''_{i+1} a_{i+1} + \dots + \beta''_m a_m, \end{aligned}$$

The couple of regression models which guarantees the lowest predictive information loss is selected. Let  $a_h$  be the variable for which the lowest predictive information loss is obtained, the set of parameters  $Ea = \{a_k \in P - \{a_h\} \mid |\beta'_k - \beta''_k| \leq \sigma_k\}$ <sup>3</sup> is selected and associated with the time window  $w : [t_F..t_L]$  to form the event  $e : \langle t_F, t_L, Ea, IEa, SEa \rangle$ . The set  $Ea$  is further filtered in order to remove those parameters for which no interval of values which discriminates the measurements in  $w$  from those in  $w'$  can be generated. This permits also to determine the element  $IEa$ . In particular, for each  $a_k \in Ea$  the interval  $[inf_k, sup_k]$  is computed by taking the minimum ( $inf_k$ ) and maximum ( $sup_k$ ) value of  $a_k$  in  $w$ . If  $[inf_k, sup_k]$  is weakly consistent with respect to values taken by  $a_k$  during the time window  $w'$  then  $a_k$  is kept, otherwise it is filtered out. Weak consistency is verified by computing the weighted average of the zero-one loss function on the measurements in  $w'$ , where weights decrease proportionally with the time points in  $w'$ . Finally, the filtered set of  $m'$  variables will be associated with a set of intervals  $\{[inf_1, sup_1], \dots, [inf_k, sup_k], \dots, [inf_{m'}, sup_{m'}]\}$ , which corresponds to the quantitative description  $IEa$  of the event  $e : \langle t_F, t_L, Ea, IEa, SEa \rangle$ . The set  $SEa : \{sv_1, \dots, sv_k, \dots, sv_m\}$  is determined through the same technique of temporal abstraction introduced in the section 3.1. It contains a symbolic value

<sup>3</sup>  $\sigma_k$  is automatically determined and is the standard deviation of the  $k$ -th coefficient of linear regression models computed on non-overlapping time-windows of size  $t_L - t_F$  over  $(S_j, S_{j+1})$ .

for each  $a_k$  and each  $sv_k$  denotes the slope of the regression line built on the data in  $[t_F..t_L]$ .

Once the candidates for each single pair  $(S_j, S_{j+1})$  in  $R$  have been generated, the sequence with the most statistically interesting events is identified by selecting the *most supported* events. An event  $e_u$  is *most supported* if it meets the following two conditions: 1) there exists a set of candidates  $\{e_1, e_2, \dots, e_t\}$  which contains the same information of  $e_u$ , that is:  $\forall e_q, q = 1, \dots, t, e_q \neq e_u$ , the set of parameters  $Ea$  associated to  $e_q$  includes the set of parameters associated to  $e_u$ , the time interval  $[t_F, t_L]$  associated to  $e_q$  includes the time interval associated to  $e_u$ , and, finally, the set of symbolic values  $SEa$  and the intervals  $IEa$  associated to the parameters of  $e_q$  coincide; 2) no event  $e_v$  exists whose information is contained in a set of candidates  $\{e_1, e_2, \dots, e_{t'}\}$  with  $|\{e_1, e_2, \dots, e_{t'}\}| > |\{e_1, e_2, \dots, e_t\}|$ . The *support* of the event  $e_u$  is computed as follows: let  $\{e_1, e_2, \dots, e_z\}$  be the set of candidates such that the time interval associated to each of them contains that of  $e_u$  and  $\{e_1, e_2, \dots, e_t\}$  be the set of candidates as described at the point 1), then the support of  $e_u$  is  $supp(e_u) = (t + 1)/z$ . The sequence of the most supported events for each pair of disease stages  $(S_j, S_{j+1}) \in R$  forms the set  $ES$  of sequences of events.

### 3.4 Discovery of Temporal Patterns

Discovery of temporal patterns from  $ES$  is performed by resorting to the ILP method for frequent patterns mining implemented in SPADA [7]. The sequences generated in the section 3.3 are modeled with the logic predicates introduced in the section 2 and stored as sets of ground atoms in the extensional part  $D_E$  of a deductive database  $D$  [2]. The intensional part  $D_I$  of the database  $D$  is defined with the predicates based on Allen temporal logic [1]:  $D_I$  represents background knowledge on the problem (e.g., precedence relationships between two events through the predicate *before*()) and allows to entail additional atoms by applying these predicates to the extensional part. For example, give two sample sequences  $seq1 : \langle e_1, e_2 \rangle$ ,  $seq2 : \langle e_3, e_4 \rangle$  the extensional part  $D_E$  of  $D$  would include the following ground atoms: *sequence(seq1)*. *sequence(seq2)*.

*event(seq1, e1)*. *event(seq1, e2)*. *event(seq2, e3)*.

*event(seq2, e4)*. *time\_tF(e1, 10)*. *time\_tL(e1, 15)*. *time\_tF(e2, 22)*. *time\_tL(e2, 25)*.

*time\_tF(e3, 90)*. *time\_tL(e3, 110)*. *time\_tF(e4, 170)*. *time\_tL(e4, 190)*.

where the constants  $seq1$  and  $seq2$  denote two distinct sequences, while the constants  $e_1, e_2, e_3, e_4$  identify four events. The intensional part  $D_I$  is formulated as the logic program:

*before(E1, E2) ← event(S, E1), event(S, E2), E1 ≠ E2, time\_tL(E1, T1), time\_tF(E2, T2), T1 < T2, not(event(S, E3), E3 ≠ E1, E3 ≠ E2, time\_tF(E3, T3F), time\_tL(E3, T3L), T1 < T3F, T3L < T2)*

by considering the atoms in  $D_E$  the ground atoms *before(e1, e2)*, *before(e3, e4)* are entailed and added to  $D_E$ .

By following the level-wise method integrated in SPADA, the process of temporal patterns discovery performs a search in the space of patterns and finds



out patterns whose support is greater than the user-defined threshold  $minF$  (frequent patterns) while it prunes those with support less than  $minF$  (infrequent patterns). The support of a pattern  $P$  is the percentage of sequences in  $D$  which covers the pattern  $P$ . The implementation of the anti-monotonicity of the support in the system guarantees the effectiveness of the level-wise method.

## 4 Application to Biomedical Data

In this section we explore the applicability of the proposed framework to a scenario of biomedicine. In particular, we focus on the analysis of data observed during a polysomnography, namely longitudinal data which describe the dynamic process of the human sleep, in order to investigate sleep disorders. Sleep disorders represent an issue of great importance and widely investigated in medicine because some serious diseases are accompanied by typical sleep disturbances. This attracts the interest of several scientific communities and, in this work, it is studied to discover patterns of events, in terms of breathing and cardiovascular system time-annotated disorders, which may trigger particular stages of the human central nervous system during sleep.

**Dataset Description.** The dataset [4] has been created by sampling measurements at 1 second of a patient from 21.30 p.m. to 6.30 a.m. Physiological parameters are *eeg* (electroencephalogram), *leog*, *reog* (electrooculograms), *emg* (electromyogram), *ecg* (electrocardiogram), *airflow*, (nasal respiration), *thorex* (thoracic excursion), *abdoex* (abdominal excursion), *pr* (heart rate) and *saO2* (arterial oxygen saturation). Where, *ecg*, *airflow*, *thorex*, *abdoex*, *pr*, *saO2* describe the cardiovascular and respiratory systems, while *eeg*, *leog*, *reog*, *emg* describe the central nervous system.

**Results.** Different sets  $S$  of disease stages are obtained by tuning  $minSD$  [5]. For each  $S$ , several collections  $R$  are created by setting  $CS$  to 60, 70, 80. Pattern are discovered from these collections by setting the threshold  $minF$  to 5% (Table [1]). As we can see the number of discovered patterns ( $\#patterns$ ) is strongly dependent on the minimum duration of the stages. Indeed, the greater the stages, the higher the dissimilarity between the stages and the lower the number of similar pairwise stages (cardinality of  $R$ ). This can be due to the fact that the fluents of stages with longer duration characterize and discriminate an higher number of physiological measurements. Therefore, they tend to be too specific for the set of data to characterize and very dissimilar from other fluents. In these cases, the cardinality of  $R$  is lower and this produces a set  $ES$  with a small number of sequences where it could be difficult to discover frequent patterns.

A first interesting result is produced when the minimum duration is set to 60 secs and  $CS$  to 60. In this case a set  $ES$  of nine sequences (as many the pairs of stages) of complex events is identified, while 579 frequent patterns are discovered. Among them, the most frequent one, which can trigger the transition depicted by the 9 pairs of stages, is so described:

<sup>4</sup> Accessible at <http://www.physionet.org/physiobank/>

**Table 1.** Patterns and stages discovered by tuning the minimum duration and  $CS$ 

minimal duration (secs)	S	CS	R	#pattern
60	139	60	9 pairwise stages	579
		70	3 pairwise stages	112
		80	0	0
120	126	60	6 pairwise stages	63
		70	3 pairwise stages	34
		80	0	0
300	31	60	3 pairwise stages	7
		70	1 pairwise stages	4
		80	0	0

*sequence(S), event(E1, S), event(E2, S), event(E3, S), before(E1, E2), before(E2, E3), parameter\_of(E1, P1), is\_a(P1, abdoex), value\_interval(P1, [-1.412, 0.722]'), symbolic\_value(P1, 'STRONG\_INCREASE'), parameter\_of(E2, P2), is\_a(P2, airflow), value\_interval(P2, [-2.322, 3.482]'), symbolic\_value(P2, 'STRONG\_DECREASE'), parameter\_of(E3, P3), is\_a(P3, saO2), value\_interval(P3, [94.013, 95.012]'), symbolic\_value(P3, 'DECREASE')* [support = 21.4%]

This pattern involves both temporal predicates (*before()*), structural predicates (e.g., *parameter\_of()*) and properties (e.g., *symbolic\_value()*) and it is supported by a percentage of 21.4% of the total sequences.

Patterns with more predicates but with lower support are rather discovered at higher values of the minimal duration. For instance, one pattern mined when the minimal duration is 120 secs and  $CS=60$  is the following:

*sequence(S), event(E1, S), event(E2, S), event(E3, S), before(E1, E2), before(E2, E3), before(E3, E4), parameter\_of(E1, P11), is\_a(P11, thorex), value\_interval(P11, [-3.984, 3.984]'), symbolic\_value(P11, 'INCREASE'), parameter\_of(E2, P21), is\_a(P21, abdoex), value\_interval(P21, [-1.757, 1.82]'), symbolic\_value(P21, 'STRONG\_INCREASE'), parameter\_of(E2, P22), is\_a(P22, thorex), value\_interval(P22, [-0.91, 2.071]'), symbolic\_value(P22, 'STRONG\_INCREASE'), parameter\_of(E3, P3), is\_a(P3, saO2), value\_interval(P3, [97.010, 98.009]'), symbolic\_value(P3, 'DECREASE'), parameter\_of(E4, P4), is\_a(P4, abdoex), value\_interval(P4, [-1.663, 1.443]'), symbolic\_value(P4, 'STEADY')* [support = 7.14%]

This pattern demonstrates empirically that when the stage duration is higher, then the frequency of temporal pattern is lower. Indeed, a larger value of the minimal duration leads to the generation of wider time-windows and a numerous set of complex events, many of which are so different to reduce the frequency of patterns of events. This observation is also confirmed by the accuracy of the results (Table 2) of the method of event detection (subsection 3.3). Indeed, when the minimal duration is 120 secs (Table 2 right) the number of true positive events (sensitivity) decreases while the number of false positive events increases, and this leads to avoid that the true positive events contribute to form the final set of frequent patterns. True positive events are defined by asking domain experts to manually identify physiological parameters expected to be involved in known events.

**Table 2.** Accuracy of the event detection for minimal duration set to 60 secs (left) and 120 secs (right) and *CS* to 60

$[t_F..t_L]$ width	sensitivity (%)	specificity (%)	$[t_F..t_L]$ width	sensitivity (%)	specificity (%)
10	71	44	20	67	39
15	68	46	30	62	42
20	64	43	40	59	40
25	70	48	50	64	36
30	71	48	60	66	41

## 5 Conclusions

We investigated some issues raising when analyzing longitudinal data and proposed a combined approach driven by only data which does not (necessarily) rely on domain knowledge. Given the characteristic of longitudinal data to represent a dynamic process, the approach can have particular usefulness in the initial or preliminary investigations of the processes, as the experiments empirically prove. As future work we plan to explore the possibilities to integrate other forms of temporal data describing the same process into the several tasks of the framework.

**Acknowledgment.** This work is in partial fulfillment of the research objectives of the project ATENEO-2010: "Modelli e Metodi Computazionali per la Scoperta di Conoscenza in Dati Spazio-Temporal".

## References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11), 832–843 (1983)
2. Ceri, S., Gottlob, G., Tanca, L.: *Logic Programming and Databases*. Springer, Heidelberg (1990)
3. Chen, X., Petrounias, I.: A framework for temporal data mining. In: Quirchmayr, G., Bench-Capon, T.J.M., Schweighofer, E. (eds.) *DEXA 1998*. LNCS, vol. 1460, pp. 796–805. Springer, Heidelberg (1998)
4. Diday, E., Esposito, F.: An introduction to symbolic data analysis and the sodas software. *Intell. Data Anal.* 7(6), 583–601 (2003)
5. Loglisci, C., Berardi, M.: Segmentation of evolving complex data and generation of models. In: *ICDM Workshops*, pp. 269–273. IEEE Computer Society, Los Alamitos (2006)
6. Loglisci, C., Malerba, D.: Discovering triggering events from longitudinal data. In: *ICDM Workshops*, pp. 248–256. IEEE Computer Society Press, Los Alamitos (2008)
7. Malerba, D., Lisi, F.A.: An ILP method for spatial association rule mining. In: *First Workshop on Multi-Relational Data Mining*, pp. 18–29 (2001)
8. Mörchen, F.: Unsupervised pattern mining from symbolic temporal data. *SIGKDD Explorations* 9(1), 41–55 (2007)
9. Muggleton, S.: *Inductive Logic Programming*. Academic Press, London (1992)
10. Singer, J.D., Willet, J.B.: *Applied longitudinal data analysis. Modelling change and event occurrence*. Oxford University Press, Inc., Oxford (2003)

# How to Use "Classical" Tree Mining Algorithms to Find Complex Spatio-Temporal Patterns?\*

Nazha Selmaoui-Folcher and Frédéric Flouvat

University of New Caledonia, PPME, BP R4, F-98851 Nouméa, New Caledonia  
{nazha.selmaoui, frederic.flouvat}@univ-nc.nc

**Abstract.** These last years an increasing amount of spatio-temporal data has been collected to study complex natural phenomena (e.g. natural hazards, environmental change, spread of infectious diseases). Extracting knowledge to better understand the dynamic of these phenomena is a challenging task. Existing works typically use patterns (e.g. sequences, trees, graphs) to model the dynamic of the phenomenon. However, the spatio-temporal properties captured by these patterns are often limited. For example, they hardly capture the spatial and temporal interactions of factors in different districts when studying the spread of a virus. In this paper, we define a new type of pattern, called complex spatio-temporal tree, to better capture the spatio-temporal properties of natural phenomena. Then, we show how a "classical" tree mining algorithm can be used to extract these complex spatio-temporal patterns. We experiment our approach on three datasets: synthetic data, real dengue data and real erosion data. The preliminary results highlighted the interest of our approach.

**Keywords:** spatio-temporal data mining, unordered tree mining algorithms, complex spatio-temporal trees, data pre-processing.

## 1 Introduction

These last years an increasing amount of spatio-temporal data has been collected to study complex natural phenomena (e.g. natural hazards, environmental change, spread of infectious diseases). Extracting knowledge to better understand the dynamic of these phenomena is a challenging task. For example, an epidemic of dengue fever is characterized by a set of interacting factors, causing the spread of the disease in space and time. It is important to know, in this example, how and which factors have an effect on disease spread. Even if the global influence of environmental factors (water points, nearby mangrove, rainfall, humidity etc.) is known, the impact of all the factors together with their interactions stills an open problem. An other example of application having such properties is soil erosion evolution (progression or movement). The study of soil erosion dynamic is an important problem in natural risk. This phenomenon is also impacted by

---

\* This work is funded by French contract ANR-2010-COSI-012 FOSTER.

a set of environmental factors (soil type, vegetation, etc.) with circumstances evolving in space and time (cyclone, depression, heavy rain, etc.). To address these issues, experts need methods to discover and model the dynamic of such phenomena. Spatio-temporal data mining methods aim at finding solutions to better understand and describe these complex phenomena. Existing works [20] typically use patterns (e.g. sequences, trees, graphs) to model the dynamic of the phenomenon. However, the spatio-temporal properties of our phenomena (e.g. dengue and erosion) cannot be totally captured by these patterns.

To deal with this problem, we define a new type of pattern, called complex spatio-temporal tree, to better capture the spatio-temporal properties of natural phenomena. We also propose an efficient method to capture the dynamic of such phenomenon from raw data (i.e. data where the spatio-temporal dynamic is implicit). This method constructs a tree forest revealing the evolution of the different variables in space and time. Then, we show how a "classical" tree mining algorithm can be used to extract these complex spatio-temporal patterns. Results are presented on real Dengue and Erosion data sets, and on a synthetic data set.

## 2 Related Works

Several studies have been conducted on spatial patterns or temporal patterns considering only one dimension at time [20]. Recently, more works analyze jointly spatial and temporal aspects. These works deal with two major problems: trajectory mining [20,48,21,12] and event sequence mining [17,18,14,5,11].

In trajectory mining, the input of algorithms is a database of known trajectories. In [4,21,7], authors characterize the trajectories of moving objects by sequences of  $(l, t)$ , where  $l$  is an object location at time  $t$ . Their problem is to extract frequent trajectories. [8] uses temporally annotated sequences to define the problem of trajectory mining. Most of these approaches are based on data in which the phenomenon (i.e. the moving object) and its dynamic are well identified. In many spatio-temporal applications, we don't have such informations. We only have raw data where the spatio-temporal dynamic of the phenomenon is implicit.

In event sequence mining, Mohan et al. [11] are looking for cascading spatio-temporal events whose instances are located together and occurred in successive slot time. The database is a set of boolean spatio-temporal even types and their instances. To extract such patterns, the authors use the co-locations concept [5]. The patterns studied in these works allow to follow the evolution of events in space and time, but don't take into account events environment (for example soil type for erosion problem, or humidity for Dengue spread). Qian et al. [14] study the spread of co-occurrences phenomena over the zonal space. The notion of spread patterns is similar to trajectory, since they track the trajectories of each set of features individually.

The existing works cited above cannot be applied to our problems. On one hand, the available database is not a set of spatio-temporal sequences where the

phenomenon and its dynamic are clearly identified. On the other hand, we are looking for patterns representing the evolution of a phenomenon in relation with its environment (and not only the evolution of some individual events).

### 3 Basic Concepts and Definitions

**Spatio-temporal database.** Let consider a set of ordered time  $T = \{t_1 < t_2 < \dots < t_{|T|}\}$ , a set of geo-referenced geographical zone  $Z = \{z_1, z_2, \dots, z_{|Z|}\}$ , and a set of boolean attributes  $I = \{i_1, i_2, \dots, i_{|I|}\}$  characterizing each zone in a given time. We define  $\Omega_t$  as triplet  $(Z, I, \mathfrak{R}_B)$  where  $\mathfrak{R}_B \subseteq Z \times I$  is the relationship such that  $\mathfrak{R}_B(z, i) = 1$  if the attribute  $i \in I$  cover  $z \in Z$  at time  $t$ .  $\Omega_t$  is called **temporal layer** and  $\Omega = \bigcup_{t \in T} \Omega_t$  is called a **spatio-temporal database**. For example, in figure 1,  $T = \{t_1, t_2, t_3\}$ ,  $Z = \{z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9, z_{10}\}$  et  $I = \{i_1, i_2, i_3, i_4, i_5, i_6\}$ . (a), (b) and (c) represent respectively  $\Omega_{t_1}$ ,  $\Omega_{t_2}$ ,  $\Omega_{t_3}$ , and constitute a spatio-temporal database  $\Omega$  at time  $t_1, t_2$  et  $t_3$ .

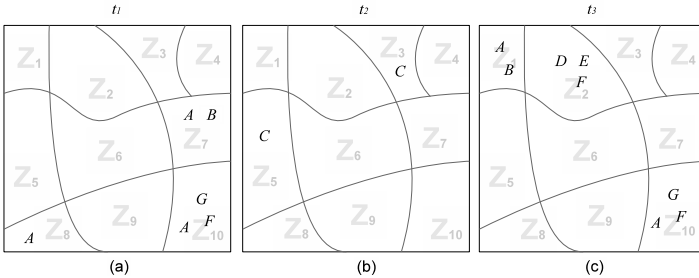


Fig. 1. Example of spatio-temporal database

**Complex spatio-temporal trees.** Let  $X \subseteq I$  be a set of boolean attributes called *itemset* [1]. Let  $z \in Z$ ,  $(X, z)$  is a **ge-referenced itemset**, if  $X$  is located in  $z$ . Let  $\mathfrak{R}_n \subseteq Z \times Z$  be a **neighbourhood relationship** such that  $\mathfrak{R}_n(z_a, z_b) = 1$  si  $z_a$  and  $z_b$  are neighbors in  $Z$ . A **complex spatio-temporal tree**  $T = (V, E)$  is a tree with  $V$  as a set of itemsets.  $E$  is a set of edges  $(X_t, X_{t'})$  located in neighbor zones and successive times  $(X_t, X_{t'} \in V)$ . For example, in  $\Omega_{t_1}$  (figure 1(a)), itemset  $\{A, B\}$  is located in zone  $z_7$ . In  $\Omega_{t_2}$  (figure 1(b)),  $\{C\}$  is located in zones  $z_3$  and  $z_5$ .  $z_7$  is a neighbor ( $\mathfrak{R}_n$  adjacency relationship) of  $z_3$ ,  $(\{A, B\}, \{C\}) \in E$  is an edge of complex spatio-temporal tree  $T$  (see figure 2(d)). Likewise,  $z_8$  is a neighbor of  $z_5$ , then  $(\{A\}, \{C\}) \in E$  (see figure 2(d)). At time  $t_3$  (figure 1(c)), the itemset  $\{D, E, F\}$  is located in zone  $z_2$ .  $z_3$  is a neighbor of  $z_2$ , the edge  $(\{C\}, \{D, E, F\})$  is created in  $E$  (figure 2(e)).

**Complex subtree mining.** Given a spatio-temporal database and a user-defined threshold *minoccur*. The problem of complex subtree mining is to extract unordered, embedded or induced (depending on application) complex subtrees where the total number of occurrences (i.e. weighted support) is greater or equal than *minoccur* (see [22]).

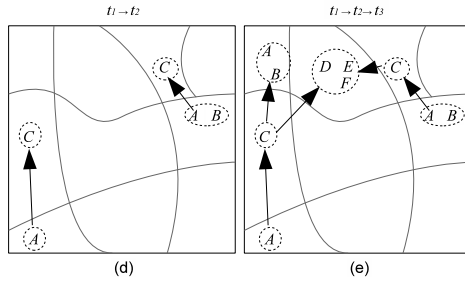


Fig. 2. Example of spatio-temporal trees generated from database of figure 1

## 4 Discovering Complex Spatio-temporal Trees from Raw Data

### 4.1 Incremental Construction of a Spatio-temporal Tree Forest

In many applications (e.g. natural hazards, environmental changes, spread of infectious diseases), the phenomenon to study and its dynamic are not clearly identified (to the opposite of applications such as trajectory mining). In other words, the database is not composed of spatio-temporal data sequences or trees representing the evolution in space and time of particular objects. Thus, to discover spatio-temporal patterns from raw data, the first step is to capture the dynamic of the phenomenon. The principle of our approach is to incrementally

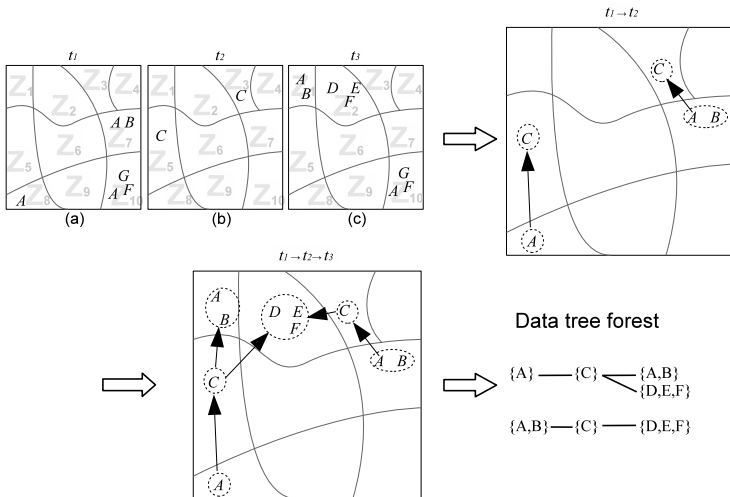


Fig. 3. Construction of a spatio-temporal data tree forest

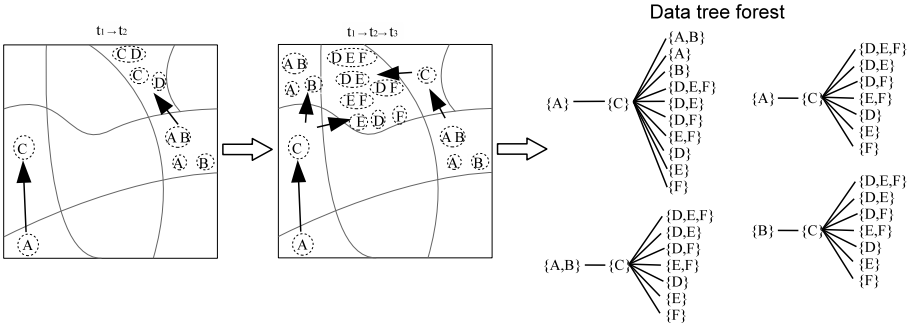
construct a data tree forest, where the vertices of the trees represent the evolution of neighboring areas in space and time (see example in figure 3).

This construction follows the steps below:

**Step 1 : zones characterization at time  $t$  and  $t + 1$ .** Each zone is associated to a set of items. This itemset represents different characteristics of the zone at a given time (e.g. temperature, nature of the ground, number of school, depending of the application). For example,  $\{A, B\}$  characterizes the zone  $Z_7$  at time  $t_1$  (figure 1). This association itemset-zone is called a geo-referenced itemset. The step 1 of our approach finds all these geo-referenced itemsets and associates to each one a vertex of the spatio-temporal data tree.

**Step 2 : evolution of neighboring zones between time  $t$  and  $t + 1$ .** This step generates the edges of the data tree forest. We recall that an edge of our spatio-temporal trees represents the evolution of two neighbor zones (i.e. the evolution of their itemsets) between time  $t$  and  $t + 1$ . In other words, an edge  $(p'_{from}, p'_{to})$  is such that itemsets  $p'_{from}$  and  $p'_{to}$  are geo-referenced itemsets of  $\Omega_t$  and  $\Omega_{t+1}$  in neighbor zones w.r.t. relationship  $\mathfrak{R}_n$ .

**Step 3 : tree extension** This last step extends existing trees and updates the spatio-temporal tree forest  $STF_{\mathfrak{R}_n}$ . An edge  $(p'_{from}, p'_{to})$  "extends" (i.e. is added to) a tree of  $STF_{\mathfrak{R}_n}$ , if there exists a leaf with itemset  $p_{leaf} = p'_{from}$  occurring at the same time ( $\Omega_t$ ) and in the same zone ( $z_{from}$ ). If there is no possible extension, then a new tree with edge  $(p'_{from}, p'_{to})$  is created in  $STF_{\mathfrak{R}_n}$ .



**Fig. 4.** Construction of a spatio-temporal data tree forest compatible with "classical" tree mining algorithms

In this paper, our objective is to extract complex spatio-temporal frequent subtrees using "classical" tree mining algorithms. However, these algorithms don't consider trees with itemsets in the vertices. They cannot extract the subtrees related to the sub-itemsets in the vertices. For example, they study the subtree  $T = (V_t, E_t)$ , with  $V_t = \{\{C\}, \{D, E, F\}\}$  and  $E_t = \{(\{C\}, \{D, E, F\})\}$ , but they don't consider subtrees such as  $T' = (V_{t'}, E_{t'})$ , with  $V_{t'} = \{\{C\}, \{D, E\}\}$  and  $E_{t'} = \{(\{C\}, \{D, E\})\}$ , or  $T'' = (V_{t''}, E_{t''})$ , with  $V_{t''} = \{\{C\}, \{D\}\}$  and  $E_{t''} = \{(\{C\}, \{D\})\}$ .



To deal with this problem, we need to integrate those sub-itemsets in the data trees. A zone will be characterized by all its sub-itemsets (except  $\emptyset$ ). The step 1 will generate this set of geo-referenced itemsets for each zone. Then, to construct the edges of the tree forest, the step 2 will execute a cartesian product between itemsets in neighbor zones between time  $t$  and  $t + 1$ . Figure 4 illustrates these modifications on the same example used in figure 3.

The number of vertices can be huge (until several millions) making the construction and mining of the data forest difficult. We have developed several mechanisms to improve the scalability of our approach. These mechanisms will be discussed in section 5.

## 4.2 Mining Embedded/Induced Spatio-temporal Subtrees

Recently subtree mining has received a lot of attention in the data mining community [2,19,6,13,22,15,16,3,9]. Our idea is to take advantage of all this work to extract new type of patterns. More particularly, we will focus on unordered subtree mining algorithms (since our vertices are unordered).

The approach described in the previous section generates a tree forest such as all the complex spatio-temporal subtrees can be found using either an induced or embedded (unordered) subtree mining algorithm. The choice between "induced" and "embedded" depends on the application. If induced subtrees are chosen, the resulting patterns will represent the evolution of neighboring zones time after time. If embedded subtrees are chosen, it will also extract patterns representing the evolution between time  $t$  and  $t + i$ , where  $i > 1$ .

Another point need to be considered: the measure and the predicate used to extract frequent patterns. The number of occurrences (also called the weighted support) is the frequency measure, since the spatio-temporal dynamic of the studied phenomenon can appear several times in the same data tree of the forest. Thus, we define the predicate as "a subtree is frequent if its number of occurrences is greater or equal than a given threshold". The main problem of this predicate is that it is not always monotone, which is necessary to use most tree mining algorithms. Let consider the example on figure 5. Suppose that we construct the spatio-temporal data forest and mine this forest with a tree mining algorithm. The mining algorithm will extract different subtrees among them the subtrees  $T$  and  $T'$  with support 3 and 1, respectively. If the support threshold is 2,  $T$  will be frequent but not  $T'$ , although  $T'$  is a subtree of  $T$ . Thus, the anti-monotonicity property is not satisfied.

To avoid this problem, we propose a fusion strategy. The principle is to merge sibling vertices having the same itemset. On figure 6, the two vertices with  $\{Y\}$  are merged (and the edges are modified in consequence), as well as the two vertices with  $\{Y\}$ . Thus, the meaning of an edge in the spatio-temporal trees is slightly different. With this solution, an edge with itemset  $I1$  and itemset  $I2$  means that  $I2$  appears in *at least one neighboring zone* of  $I1$  in the next time.

Note that the fusion solution modifies the definition of the extracted patterns. The number of occurrences and extracted patterns w.r.t. this new definition are correct, but it will underestimate (and maybe miss) some subtrees w.r.t.

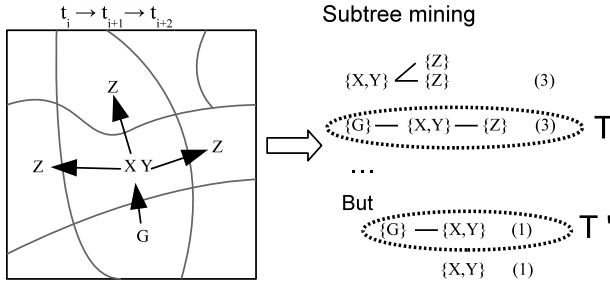


Fig. 5. Example of "incompatible" data tree

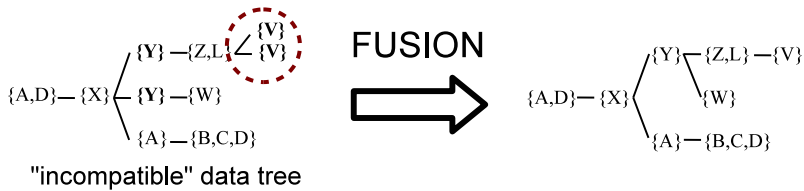


Fig. 6. Transforming an "incompatible" data tree in "compatible" data trees

the initial definition of complex spatio-temporal trees. For example, the subtree having edges  $(\{A, D\}, \{X\})$  and  $(\{X\}, \{Y\})$  will be counted once (figure 6). It represents the occurrence of  $X$  in at least one neighbor of  $\{A, D\}$ , followed by the occurrence of  $\{Y\}$  in at least one neighbor of  $\{X\}$ . However, if we consider the real data, the dynamic induced by this subtree appears twice.

## 5 Improving Performances

In this section, we propose a new data structure to improve the construction of the spatio-temporal data forest (step 3). This data structure, called a STP-tree (*Spatio-Temporal Pattern tree*), is based on a prefix tree associated with an array  $ZR$  (*Zonal References*). Each vertex of the STP-tree stores an itemset and a counter representing a number of occurrences. Each cell of the array  $ZR$  represents a zone and stores a list of vertices which have been added or updated in the data forest during the last iteration. In other words, all extendable vertices from zone  $z_{from}$  are referenced in the  $ZR[z_{from}]$  cell. Therefore, the prefix tree is used to compress the data forest and the array is used to improve its construction. Figure 7 shows an example of STP-tree. Only itemsets  $P_5$ ,  $P_6$  and  $P_7$  are linked to  $ZR$  since they are the last vertices added or updated in the data tree at time  $t_3$ . This combination of a prefix tree and a "spatial" array drastically improves the performances of step 3 at the spatio-temporal data forest construction. Recall that this step extends data trees, which consists in updating the dynamic found

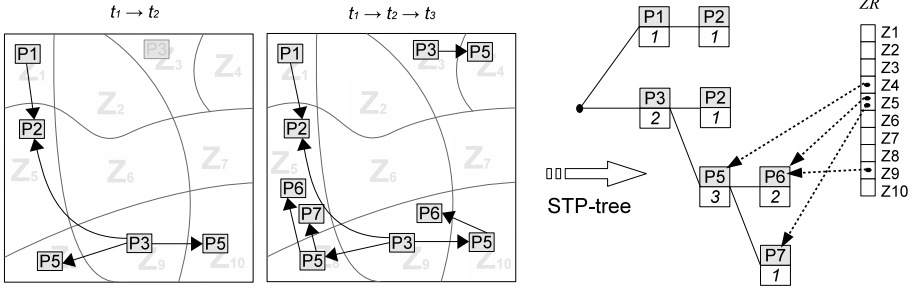


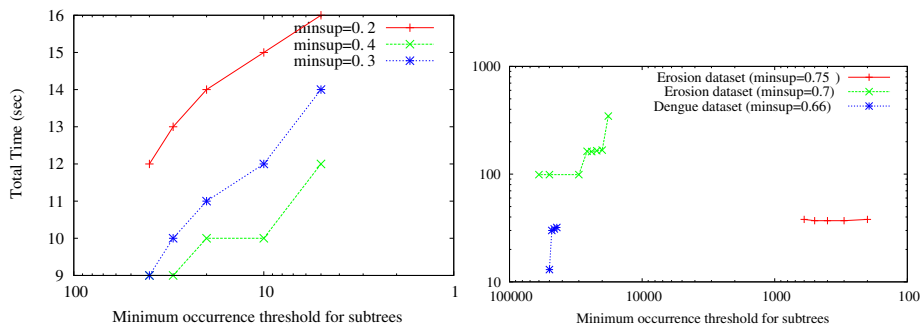
Fig. 7. Example of STP-tree

between time  $t$  and  $t + 1$ . Suppose that itemset  $P_6$  at time  $t_2$  is located at a neighbor zone of  $P_5$  at time  $t_3$  (zones  $Z_5$  and  $Z_8$  in figure 7). A question of step 3 is "does the edge  $(P_5, P_6)$  can extend trees constructed until time  $t_2$ ?". If we look the example, it is clearly yes, since we have between time  $t_1$  and  $t_3$ , the dynamic " $P_3 - > P_5 - > P_6$ ". So, a naive approach would have been to explore all the forest to find vertices having itemset  $P_5$  at time  $t_2$  in the same zone of itemset  $P_5$  of  $(P_5, P_6)$ . With our STP-tree, we don't need to explore all the vertices of the forest. At time  $t_3$ , we only have to look if the itemset  $P_5$  appears in the cell  $ZR[Z_8]$  and to link this vertex with a new vertex  $P_6$ .

## 6 Experimental Results

The proposals discussed in this paper have been integrated in a C++ prototype. We use the *SLEUTH* algorithm to extract frequent embedded unordered subtrees [22]. Our experiments were done on three data sets: a real soil erosion dataset, a real dengue dataset and a synthetic dataset. The soil erosion dataset represents the evolution of soil erosion in an area of New Caledonia. This dataset is composed of 5 dates (years) and 23 zones. It studies soil erosion w.r.t. 22 environmental attributes (443 items). The dengue dataset represents the spread of a dengue epidemic in Noumea (New Caledonia). This dataset is composed of 12 dates (months) and 32 zones (the districts of Noumea). It studies dengue w.r.t. 10 attributes (20 items). The synthetic dataset has been generated to experiment our approach on a dataset with more dates and more zones. It is composed of 700 dates, 2000 zones and 40 items.

Figure 8 shows the total execution time, i.e. time to construct the spatio-temporal data forest added to time to extract frequent subtrees. The spatial minimum support threshold (i.e. the one used for frequent closed itemsets mining) is fixed. These figures show the influence of the minimum number of occurrences threshold used to extract frequent subtrees. The table 1 details the characteristics of the spatio-temporal data forest constructed and frequent subtrees mined, for some of the previous thresholds. As show by this figure and this table, our approach is robust w.r.t. the complexity of the studied data (number



**Fig. 8.** Execution time with frequent tree mining for the synthetic dataset (left plot) and the real datasets (right plot)

**Table 1.** Spatio-temporal forest and frequent subtrees characteristics

	Synthetic dataset minoccur=5	Erosion dataset minoccur=2000	Dengue dataset minoccur=440000
number of data trees	1243	7 998 337	2 568 116
number of vertices in the forest	3 327	16 713 677	5 540 034
number of frequent subtrees	15 059	128	18
max. number of vertices in a frequent subtree	14	11	3

of zones, number of dates and size of spatio-temporal forests). It also highlights the good scalability of the SLEUTH tree mining algorithm.

## 7 Conclusion and Perspectives

This work concerns spatio-temporal data mining. One of the challenging task, in this emerging domain, is to extract knowledge to understand dynamic of spatio-temporal phenomena (spread epidemic, soil erosion evolution, etc.). In this purpose, we proposed an efficient method to generate a spatio-temporal data forest to represent these spatio-temporal phenomena. This method takes advantage of a new data structure based on a prefix tree and spatial array. The concept of complex spatio-temporal subtrees has been defined to better reflect the evolution in space and time of such phenomena. We also show how "classical" tree mining algorithms can be used to extract these new patterns. We experiment our approach on three datasets: synthetic data, real dengue data and real erosion data. The preliminary results highlighted the interest of our approach. There are many perspectives for this work. An important perspective is to propose a more convenient strategy for mining these complex subtrees (i.e. tree with itemsets in nodes). Indeed, some subtrees cannot be mined because they don't

satisfy the monotone property. We also plan to extend experimental results with experts in order to add new constraints. An other perspective is to develop an interactive visualization interface of complex spatio-temporal patterns.

**Acknowledgments.** We wish to thank especially Professor Mohammed J. Zaki for providing the SLEUTH source code. We also wish to thank the project "Prevention and prediction of dengue epidemics in New Caledonia" IRD-DASSNC-UNC-IPNC-MeteoFrance for giving us the *Dengue* data set.

## References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) VLDB, pp. 487–499. Morgan Kaufmann, San Francisco (1994)
2. Asai, T., Arimura, H., Uno, T., Nakano, S.I.: Discovering frequent substructures in large unordered trees. In: Grieser, G., Tanaka, Y., Yamamoto, A. (eds.) DS 2003. LNCS (LNAI), vol. 2843, pp. 47–61. Springer, Heidelberg (2003)
3. Balcázar, J.L., Bifet, A., Lozano, A.: Mining frequent closed rooted trees. *Machine Learning* 78(1-2), 1–33 (2010)
4. Cao, H., Mamoulis, N., Cheung, D.W.: Mining frequent spatio-temporal sequential patterns. In: ICDM, pp. 82–89. IEEE Computer Society, Los Alamitos (2005)
5. Celik, M., Shekhar, S., Rogers, J.P., Shine, J.A.: Mixed-drove spatiotemporal co-occurrence pattern mining. *IEEE Trans. Knowl. Data Eng.* 20(10), 1322–1335 (2008)
6. Chi, Y., Yang, Y., Xia, Y., Muntz, R.R.: Cmtreeminer: Mining both closed and maximal frequent subtrees. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 63–73. Springer, Heidelberg (2004)
7. Du, X., Jin, R., Ding, L., Lee, V.E., Thornton Jr., J.H.: Migration motif: a spatial - temporal pattern mining approach for financial markets. In: KDD, pp. 1135–1144 (2009)
8. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: KDD, pp. 330–339 (2007)
9. Jiménez, A., Berzal, F., Talavera, J.C.C.: Frequent tree pattern mining: A survey. *Intell. Data Anal.* 14(6), 603–622 (2010)
10. Kim, W., Kohavi, R., Gehrke, J., DuMouchel, W. (eds.): Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 22–25. ACM Press, USA (2004)
11. Mohan, P., Shekhar, S., Shine, J.A., Rogers, J.P.: Cascading spatio-temporal pattern discovery: A summary of results. In: SDM, pp. 327–338 (2010)
12. Monreale, A., Pinelli, F., Trasarti, R., Giannotti, F.: Wherenext: a location predictor on trajectory pattern mining. In: KDD, pp. 637–646 (2009)
13. Nijssen, S., Kok, J.N.: A quickstart in frequent structure mining can make a difference. Kim et al [10], 647–652
14. Qian, F., He, Q., He, J.: Mining spread patterns of spatio-temporal co-occurrences over zones. In: Gervasi, O., Taniar, D., Murgante, B., Laganà, A., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2009. LNCS, vol. 5593, pp. 677–692. Springer, Heidelberg (2009)

15. Tan, H., Dillon, T.S., Hadzic, F., Chang, E., Feng, L.: Imb3-miner: Mining induced/embedded subtrees by constraining the level of embedding. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 450–461. Springer, Heidelberg (2006)
16. Termier, A., Rousset, M.C., Sebag, M., Ohara, K., Washio, T., Motoda, H.: Dryadeparent, an efficient and robust closed attribute tree mining algorithm. *IEEE Trans. Knowl. Data Eng.* 20(3), 300–320 (2008)
17. Hsu, W., Mong Li Lee, J.W.: Mining generalized flow patterns. In: *Temporal and Spatio-Temporal Data Mining*, pp. 189–208. IGI Publishing (2009)
18. Hsu, W., Mong Li Lee, J.W.: Mining spatio-temporal trees. In: *Temporal and Spatio-Temporal Data Mining*, pp. 209–226. IGI Publishing (2009)
19. Xiao, Y., Yao, J.F., Li, Z., Dunham, M.H.: Efficient data mining for maximal frequent subtrees. In: *ICDM*, pp. 379–386. IEEE Computer Society, Los Alamitos (2003)
20. Yao, X.: Research issues in spatio-temporal data mining. In: *White paper UCGIS* (2003)
21. Yuan, M.: Toward knowledge discovery about geographic dynamics in spatiotemporal databases. In: Han, J., Miller, H.J. (eds.) *Geographic Data Mining and Knowledge Discovery*, pp. 347–365. Taylor and Francis, Abington (2008)
22. Zaki, M.J.: Efficiently mining frequent embedded unordered trees. *Fundam. Inform.* 66(1-2), 33–52 (2005)

# Inferring Fine-Grained Data Provenance in Stream Data Processing: Reduced Storage Cost, High Accuracy

Mohammad Rezwatul Huq, Andreas Wombacher, and Peter M.G. Apers

University of Twente, 7500 AE Enschede, The Netherlands  
{m.r.huq, a.wombacher, p.m.g.apers}@utwente.nl

**Abstract.** Fine-grained data provenance ensures reproducibility of results in decision making, process control and e-science applications. However, maintaining this provenance is challenging in stream data processing because of its massive storage consumption, especially with large overlapping sliding windows. In this paper, we propose an approach to infer fine-grained data provenance by using a temporal data model and coarse-grained data provenance of the processing. The approach has been evaluated on a real dataset and the result shows that our proposed inferring method provides provenance information as accurate as explicit fine-grained provenance at reduced storage consumption.

## 1 Introduction

Stream data processing often deals with massive amount of sensor data in e-science, decision making and process control applications. In these kind of applications, it is important to identify the origin of processed data. This enables a user in case of a wrong prediction or a wrong decision to understand the reason of the misbehavior through investigating the transformation process which produces the unintended result.

Reproducibility as discussed in this paper means the ability to regenerate data items, i.e. for every process  $P$  executed on an input dataset  $I$  at time  $t$  resulting in output dataset  $O$ , the re-execution of process  $P$  at any later point in time  $t'$  (with  $t' > t$ ) on the same input dataset  $I$  will generate exactly the same output dataset  $O$ . Generally, reproducibility requires metadata describing the transformation process, usually known as provenance data.

In [1], data provenance is defined as derivation history of data starting from its original sources. Data provenance can be defined either at tuple-level or at relation-level known as fine-grained and coarse-grained data provenance respectively [2]. Fine-grained data provenance can achieve reproducible results because for every output data tuple, it documents the used set of input data tuples and the transformation process itself. Coarse-grained data provenance provides similar information on process or view level. In case of updates and delayed arrival of tuples, coarse-grained data provenance cannot guarantee reproducibility.

Applying the concept of fine-grained data provenance to stream data processing introduces new challenges. In stream data processing, a transformation

process is continuously executed on a subset of the data stream known as a window. Executing a transformation process on a window requires to document fine-grained provenance data for this processing step to enable reproducibility. If a window is large and subsequent windows overlap significantly, then the provenance data size might be bigger than the actual sensor data size. Since provenance data is 'just' metadata, this approach seems to be too expensive.

In [3], we initially propose the basic idea of achieving fine-grained data provenance using a temporal data model. In this paper, we extend and complete our work to infer fine-grained data provenance using a temporal data model and coarse-grained data provenance. Adding a temporal attribute (e.g. timestamp) to each data item allows us to retrieve the overall database state at any point in time. Then, using coarse-grained provenance of the transformation, we can reconstruct the window which was used for the original processing and thus ensuring reproducibility. Due to the plethora of possible processing operations, a classification of operations is provided indicating the classes applicable to the proposed approach. In general, the approach is directly applicable if the processing of any window produces always the same number of output tuples. Eventually, we evaluate our proposed technique based on storage and accuracy using a real dataset.

This paper is structured as follows. In Section 2, we provide a detailed description of our motivating application with an example workflow. In Section 3, we discuss existing work on both stream processing and data provenance briefly. In Section 4, we explain our approach and associated requirements followed by the discussion on few issues in Section 5. Next, we present the evaluation of our approach in Section 6. Finally, we conclude with hints of future research.

## 2 Motivating Scenario

RECORD<sup>1</sup> is one of the projects in the context of the Swiss Experiment, which is a platform to enable real-time environmental experiments. One objective of the RECORD project is to study how river restoration affects water quality, both in the river itself and in groundwater. Several sensors have been deployed to monitor river restoration effects. Some of them measure electric conductivity of water. Increasing conductivity indicates the higher level of salt in water. We are interested to control the operation of the drinking water well by facilitating the available online sensor data.

Based on this motivating scenario, we present a simplified workflow, that will also be used for evaluation. Fig. 1 shows the workflow based on the RECORD project. There are three sensors, known as: Sensor#1, Sensor#2 and Sensor#3. They are deployed in different locations in a known region of the river which is divided into a grid with  $3 \times 3$  cells. These sensors send data tuples, containing *sensor id*, *(x,y) coordinates*, *timestamp* and *electric conductivity*, to source processing element named  $PE_1$ ,  $PE_2$  and  $PE_3$  which outputs data tuples in a *view*  $V_1$ ,  $V_2$  and  $V_3$  respectively. These views are the input for a *Union* processing

<sup>1</sup> <http://www.swiss-experiment.ch/index.php/Record:Home>



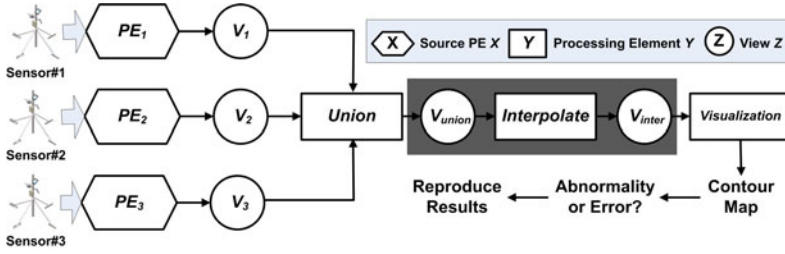


Fig. 1. Workflow based on RECORD scenario

element which produces a view  $V_{union}$  as output. This view acts as an input to the processing element *Interpolate*. The task of *Interpolate* is to calculate the interpolated values for all the cells of the grid using the values sent by the three sensors and store the interpolated values in the view  $V_{inter}$ . Next,  $V_{inter}$  is used by the *Visualization* processing element to produce a contour map of electric conductivity. If the map shows any abnormality, researchers may want to reproduce results to validate the previous outcome. The dark-shaded part of the workflow in Fig. 1 is considered to evaluate our proposed approach.

### 3 Related Work

Stream data processing engines reported in [4], [5], [6]. These techniques proposed optimization for storage space consumed by sensor data. However, neither of these systems maintain provenance data and cannot achieve reproducible results.

Existing work in data provenance addresses both fine and coarse-grained data provenance. In [7], authors have presented an algorithm for lineage tracing in a data warehouse environment. They have provided data provenance on tuple level. LIVE [8] is an offshoot of this approach which supports streaming data. It is a complete DBMS which preserves explicitly the lineage of derived data items in form of boolean algebra. However, both of these techniques incur extra storage overhead to maintain fine-grained data provenance.

In sensornet republishing [9], the system documents the transformation of online sensor data to allow users to understand how processed results are derived and support to detect and correct anomalies. They used an annotation-based approach to represent data provenance explicitly. In [10], authors proposed approaches to reduce the amount of storage required for provenance data. To minimize provenance storage, they remove common provenance records; only one copy is stored. Their approach has less storage consumption than explicit fine-grained provenance in case of sliding overlapping windows. However, these methods still maintain fine-grained data provenance explicitly.

## 4 Proposed Solution

### 4.1 Provenance Inference Algorithm

The first phase of our provenance inference algorithm is to document coarse-grained provenance of the transformation which is an one-time action, performed during the setup of a processing element. The algorithms for the next two phases are given here along with an example. To explain these algorithms, we consider a simple workflow where a *processing element* takes one *source view* as input and produces one *output view*. Moreover, we assume that, *sampling time of source view* is 2 time units and the window holds 3 tuples. The *processing element* will be executed after arrival of every 2 tuples.  $t_1$ ,  $t_2$  and so on are different points in time and  $t_1$  is the starting time.

**Document Coarse-grained Provenance:** The stored provenance information is quite similar to *process provenance* reported in [11]. Inspired from this, we keep the following information of a processing element specification based on [12] and the classification introduced in Section 4.2 as coarse-grained data provenance.

- Number of sources: indicates the total number of source views.
- Source names: a set of source view names.
- Window types: a set of window types; the value can be either *tuple* or *time*.
- Window predicates: a set of window predicates; one element for each source. The value actually represents the size of the window.
- Trigger type: specifies how the *processing element* will be triggered for execution. The value can be either *tuple* or *time*.
- Trigger predicate: specifies when a *processing element* will be triggered for execution. If *trigger type* is *tuple* and the value of *trigger predicate* is 10, it means that the processing element will be executed after the arrival of every 10th tuple.

---

#### Algorithm 1: Retrieve Data & Reconstruct Processing Window Algorithm

---

**Input:** A tuple  $T$  produced by processing element  $PE$ , for which fine-grained provenance needs to be found

**Output:** Set of input tuples  $I_j^{P_w}$  for each source  $j$  which form processing window  $P_w$  to produce  $T$

```

1 TransactionTime ← getTransactionTime(PE, T);
2 noOfSources ← getNoOfSources(PE);
3 for j ← 1 to noOfSources do
4     sourceView ← getSourceName(PE, j);
5     wType ← getWindowType(sourceView);
6     wPredicate ← getWindowPredicate(sourceView);
7      $I_j^{P_w}$  ← getLastNTuples(sourceView, TransactionTime, wType, wPredicate);
8 end
    
```

---

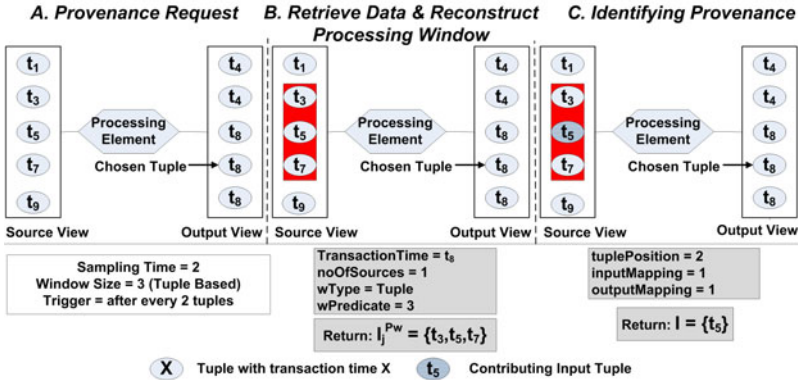


Fig. 2. Retrieval, Reconstruction and Inference phases of Provenance Algorithm

**Retrieve Data & Reconstruct Processing Window:** This phase will be only executed if the provenance information is requested for a particular output tuple  $T$  generated by a processing element  $PE$ . The tuple  $T$  is referred here as *chosen tuple* for which provenance information is requested (see Fig. 2.A).

We apply a temporal data model on streaming sensor data to retrieve appropriate data tuples based on a given timestamp. The temporal attributes are: i) **valid time** represents the point in time a tuple was created by a sensor and ii) **transaction time** is the point in time a tuple is inserted into a database. While *valid time* is anyway maintained in sensor data, *transaction time* attribute requires extra storage space.

The method of retrieving data and reconstructing processing window is given in Algorithm 1. The *transaction time* of the chosen tuple and number of participating sources are retrieved in line 1 and 2. Then, for each participating source view, we retrieve its *name*, *window type* and *window predicate* in line 4-6. Then, we retrieve the set of the input tuples which form the processing window based on the chosen tuple’s *transaction time* in line 7. If window type is *tuple*, we retrieve last  $n$  tuples added to the source view before the *TransactionTime* where  $n$  is the window predicate or window size. On the contrary, if window type is *time*, we retrieve tuples having *transaction time* ranging within  $[TransactionTime - wPredicate, TransactionTime)$ . The retrieved tuples reconstruct the processing window which is shown by the tuples surrounded by a dark shaded rectangle in Fig. 2.B.

**Identifying Provenance:** The last phase associates the chosen output tuple with the set of contributing input tuples based on the reconstructed window in the previous phase. This mapping is done by facilitating the output and input tuples order in their respective view. Fig. 2.C shows that the chosen tuple in the output view maps to the 2nd tuple in the reconstructed window (shaded rectangle in source view). To compute the tuple position and infer provenance, some requirements must be satisfied which are discussed next.

## 4.2 Requirements

Our provenance inference algorithm has some requirements to be satisfied. Most of the requirements are already introduced to process streaming data in existing literature. In [9], authors propose to use transaction time on incoming stream data as **explicit timestamps**. Ensuring **temporal ordering** of data tuples is one of the main requirements in stream data processing. This property ensures that input tuples producing output tuples in the same order of their appearance and this order is also preserved in the output view. **Classification of operations** is an additional requirement for the proposed approach.

In our streaming data processing platform, various types of SQL operations (e.g. *select*, *project*, *aggregate functions*, *cartesian product*, *union*) and generic functors (e.g. *interpolate*, *extrapolate*) are considered as *operations* which can be implemented inside a *processing element*. Each of these operations takes a number of input tuples and maps them to a set of output tuples.

*Constant Mapping Operations* are PEs which have a fixed ratio of mapping from input to output tuples per window, i.e.  $1 : 1, n : 1, n : m$ . As for example: project, aggregates, interpolation, cartesian product, and union. *Variable Mapping Operations* are PEs which have not a fixed ratio of mapping from input to output tuples per window, e.g. select and join. Currently, our inference algorithm can be applied directly to constant mapping operations. Each of these operations has property like *Input tuple mapping* which specifies the number of input tuples per source contributed to produce exactly one output tuple and *Output tuple mapping* which refers to the number of output tuples produced from exactly one input tuple per source. Moreover, there are operations where all sources (e.g. join) or a specific source (e.g. union) can contribute at once. These information should be also documented in coarse-grained data provenance.

## 4.3 Details on Identifying Provenance Phase

Algorithm 2 describes the approach we take to identify the correct provenance. First, we retrieve our stored coarse-grained provenance data in line 2-5. For operations where only one input tuple contributes to the output tuple (line 6), we have to identify the relevant contributing tuple. In case there are multiple sources used but only one source is contributing (line 7), a single tuple is contributing. Based on the temporal ordering, the knowledge of the nested processing of multiple sources, the contributing source and the output tuple mapping, the position of the tuple in the input view which contributed to the output tuple can be calculated (line 9). The tuple is then selected from the contributing input source in line 10.

If there is one source or there are multiple sources equally contributing to the output tuple, the position of the contributing tuple per source has to be determined (line 13). The underlying calculation is again based on the knowledge of the nested processing of multiple sources, the contributing source and the output tuple mapping, the position of the tuple in the input view  $j$ . In line 14 the tuple is selected based on the derived position from the set of input tuples.

---

**Algorithm 2:** Identifying Provenance Algorithm

---

**Input:** Set of input tuples  $I_j^{P_w}$  for each source  $j$  which form processing window  $P_w$  to produce  $T$

**Output:** Set of input tuples  $I$  which contribute to produce  $T$

```

1  $I = \emptyset$ ;
2 inputMapping  $\leftarrow$  getInputMapping( $PE$ );
3 outputMapping  $\leftarrow$  getOutputMapping( $PE$ );
4 contributingSource  $\leftarrow$  getContributingSource( $PE, T$ );
5 noOfSources  $\leftarrow$  getNoOfSources( $PE$ );
6 if inputMapping = 1 then           /* only one input tuple contributes */
7   if noOfSources > 1  $\wedge$  contributingSource = Specific then
8     parent  $\leftarrow$  getParent( $PE, T$ );
9     tuplePosition  $\leftarrow$  getPosition( $PE, T, parent, outputMapping$ );
10     $I \leftarrow$  selectTuple( $I_{parent}^{P_w}, tuplePosition$ );
11  else
12    for  $j \leftarrow 1$  to noOfSources do
13      tuplePosition  $\leftarrow$  getPosition( $PE, T, j, outputMapping$ );
14       $I \leftarrow$  selectTuple( $I_j^{P_w}, tuplePosition$ )  $\cup$   $I$ ;
15    end
16  end
17 else                               /* all input tuples contribute */
18   for  $j \leftarrow 1$  to noOfSources do
19      $I \leftarrow I_j^{P_w} \cup I$ ;
20   end
21
```

---

In cases where all input tuples contribute to the output tuple independent of the number of input sources, all tuples accessible of all sources (line 18) are selected. Thus, the set of contributing tuples is the union of all sets of input tuples per source (line 19).

## 5 Discussion

The proposed approach can infer provenance for *constant mapping* operations. However, *variable mapping* operations have not any fixed mapping ratio from input to output tuples. Therefore, the approach cannot be applied directly to these operations. One possible solution might be to transform these operations into *constant mapping operations* by introducing *NULL* tuples in the output. Suppose, for a *select* operation, the input tuple which does not satisfy the selection criteria will produce a *NULL* tuple in the output view, i.e. a tuple with a *transaction time* attribute and the remaining attributes are *NULL* values. We will give an estimation of storage overhead incurred by this approach in future.

Our inference algorithm provides 100% accurate provenance information under the assumption that the system is almost infinitely fast, i.e. no processing

delay. However, in a typical system due to other working load, it is highly probable that a new input tuple arrives before the processing is finished and our inference algorithm may reconstruct an erroneous processing window inferring inaccurate provenance. In future, we will address this limitation.

## 6 Evaluation

### 6.1 Evaluating Criteria and Datasets

The consumption of storage space for fine-grained data provenance is our main evaluation criteria. Existing approaches [8], [9], [10] record fine-grained data provenance explicitly in varying manners. Since these implementations are not available, our proposed approach is compared with an implementation of a fine-grained data provenance documentation running in parallel with the proposed approach on the Sensor Data Web<sup>2</sup> platform.

To implement the explicit fine-grained data provenance, we create one *provenance view* for each output view. This provenance view documents *output tuple ID*, *source tuple ID* and *source view* for each tuple in the output view. We also assign another attribute named as *tuple ID* which is auto incremental and primary key of the *provenance view*.

To check whether both approaches produce the same provenance information, explicit fine-grained provenance information is used as a ground truth and it is compared with the fine-grained provenance inferred by our proposed approach, i.e. the accuracy of the proposed approach.

For evaluation, a real dataset<sup>3</sup> measuring electric conductivity of the water, collected by the RECORD project is used. The experiments (see Section 2) are performed on a PostgreSQL 8.4 database and the Sensor Data Web platform. The input dataset contains 3000 tuples requiring 720kB storage space which is collected during last half of November 2010.

### 6.2 Storage Consumption

In this experiment, we measure the storage overhead to maintain fine-grained data provenance for the *Interpolation* processing element based on our motivating scenario (see Section 2) with *overlapping* and *non-overlapping* windows. In the non-overlapping case, each window contains three tuples and the operation is executed for every third arriving tuple. This results in about  $3000 \div 3 \times 9 = 9000$  output tuples since the interpolation operation is executed for every third input tuple and it produces 9 output tuples at a time, requires about 220kB space. In the overlapping case, the window contains 3 tuples and the operation is executed for every tuple. This results in about  $3000 \times 9 = 27000$  output tuples which require about 650kB. The sum of the storage costs for input and output tuples, named as sensor data, is depicted in Fig. 3 as dark gray boxes, while the provenance data storage costs is depicted as light gray boxes.

<sup>2</sup> <http://sourceforge.net/projects/sensordataweb/>

<sup>3</sup> <http://data.permasense.ch/topology.html#topology>

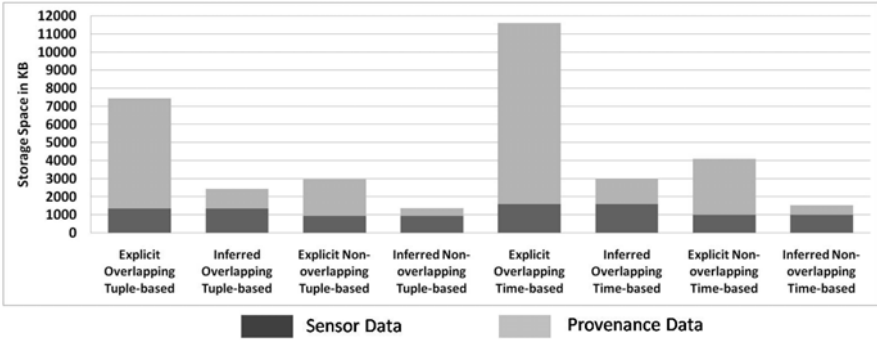


Fig. 3. Storage space consumed by Explicit and Inference method in different cases

From Fig. 3, we see that for *explicit* approach, the amount of required provenance information is more than twice the amount of actual sensor data in the best case (non-overlapping). On the contrary, the proposed *inference* approach requires less than half the storage space to store provenance data compared to the actual sensor data in non-overlapping cases and at least 25% less space in overlapping cases. As a whole, for *interpolation*, inferring provenance takes at least 4 times less storage space than the *explicit* approach. Therefore, our proposed approach clearly outperforms the explicit method. This is because the proposed approach adds only one timestamp attribute to each input and output tuple whereas the explicit approach adds the same provenance tuple several times because of overlapping sliding windows. Our proposed approach is not dataset dependent and also has window and trigger independent storage cost. The overhead ratio of provenance to sensor data depends on the payload of input tuples.

Additional tests confirm the results. We perform experiments for *project* and *average* operation with same dataset and different window size. In *project* operation, our method takes less than half storage space to maintain provenance data than the explicit method. For *average* operation, our proposed *inference* method takes at least 4 times less space than the *explicit* method. Please be noted that this ratio depends on the chosen window size and trigger specification. With the increasing window size and overlapping, our approach performs better.

### 6.3 Accuracy

To measure the accuracy, we consider provenance data tuples documented by explicit fine-grained data provenance as ground truth. Our experiment shows that the proposed *inference* method achieves 100% accurate provenance information. In our experiments, the processing time is much smaller than the minimum value of sampling time of data tuples, i.e. no new tuples arrive before finish processing, as discussed in Section 5). This is why, *inference* method is as accurate as *explicit* approach. These results are confirmed by all tests performed so far.

## 7 Conclusion and Future Work

Reproducibility is a requirement in e-science, decision making applications to analyze past data for tracing back problems in the data capture or data processing phase. In this paper, we propose a method of inferring fine-grained data provenance which reduces storage cost significantly compared to explicit technique and also provides accurate provenance information to ensure reproducibility. Our proposed approach is dataset independent and the larger the subsequent windows overlap, the more the storage reduction is. In future, we will address limitations in case of longer and variable delays for processing and sampling data tuples to ensure reproducibility at low storage cost.

## References

1. Simmhan, Y.L., Plale, B., Gannon, D.: A survey of data provenance in e-science. *SIGMOD Rec.* 34(3), 31–36 (2005)
2. Buneman, P., Tan, W.C.: Provenance in databases. In: *SIGMOD: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pp. 1171–1173. ACM, New York (2007)
3. Huq, M.R., Wombacher, A., Apers, P.M.G.: Facilitating fine grained data provenance using temporal data model. In: *Proceedings of the 7th Workshop on Data Management for Sensor Networks (DMSN)*, pp. 8–13 (September 2010)
4. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 1–16. ACM, New York (2002)
5. Abadi, D., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: a new model and architecture for data stream management. *The VLDB Journal* 12(2), 120–139 (2003)
6. Abadi, D., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., et al.: The design of the borealis stream processing engine. In: *Second Biennial Conference on Innovative Data Systems Research (CIDR 2005)*, Asilomar, CA, pp. 277–289 (2005)
7. Cui, Y., Widom, J.: Lineage tracing for general data warehouse transformations. *VLDB Journal* 12(1), 41–58 (2003)
8. Sarma, A., Theobald, M., Widom, J.: LIVE: A Lineage-Supported Versioned DBMS. In: Gertz, M., Ludäscher, B. (eds.) *SSDBM 2010*. LNCS, vol. 6187, pp. 416–433. Springer, Heidelberg (2010)
9. Park, U., Heidemann, J.: Provenance in sensornet republishing. *Provenance and Annotation of Data and Processes*, 280–292 (2008)
10. Chapman, A., Jagadish, H., Ramanan, P.: Efficient provenance storage. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 993–1006. ACM, New York (2008)
11. Simmhan, Y.L., Plale, B., Gannon, D.: Karma2: Provenance management for data driven workflows. *International Journal of Web Services Research* 5, 1–23 (2008)
12. Wombacher, A.: Data workflow - a workflow model for continuous data processing. Technical Report TR-CTIT-10-12, Centre for Telematics and Information Technology University of Twente, Enschede (2010)



# Approximate Query on Historical Stream Data

Qiyang Duan<sup>1</sup>, Peng Wang<sup>1</sup>, MingXi Wu<sup>2</sup>, Wei Wang<sup>1</sup>, and Sheng Huang<sup>1,3</sup>

<sup>1</sup> Fudan University, No. 220, Handan Road, Shanghai, 200433, China

{qduan, pengwang5, weiwang1}@fudan.edu.cn

<sup>2</sup> Oracle Corporation, Redwood shores, CA 94065, USA

mingxi.wu@oracle.com

<sup>3</sup> 399 Keyuan Road Shanghai, 201203, China

huangssh@cn.ibm.com

**Abstract.** We present a new Stream OLAP framework to approximately answer queries on historical stream data, in which each cell is extended from a single value to a synopsis structure. The cell synopses can be constructed by the existing well researched methods, including Fourier, DCT, Wavelet, PLA, etc. To implement the Cube aggregation operation, we develop algorithms that aggregate multiple lower level synopses into a single higher level synopsis for those synopsis methods. Our experiments provide comparison among all used synopsis methods, and confirm that the synopsis cells can be accurately aggregated to a higher level.

**Keywords:** Approximate Query, Synopsis, Stream, OLAP, Fourier, DCT, PLA, Wavelet.

## 1 Introduction

Recently stream data management becomes a distinct topic apart from other data types because of its unique nature: continuously growing large volume. Because of this, stream data can only be saved for a short period, e.g., in a sliding window [1]. Historical data beyond the sliding window is simply discarded. However, many real life applications demand range or similarity queries against the historical data. For example, in a system workload database, users want to understand the average workload between 6:00AM and 7:00AM in a specific month in a past year. This information is precious for capacity planning, workload distribution allocation [1]. It is not hard to imagine a range query example in such an application:

```
Select sum(workload)/count(workload) from workload_tab
where time between 6:00AM and 7:00AM and date = 'January 2006';
```

Since it is impossible to store the entire stream we have seen, in order to answer queries on the historical data, one has to build a data synopsis to capture the historical information and use it to answer queries to the past data. There have been some researches about approximate query [2] over extremely large data in the last decade, to name a few, Wavelet [3–6], DCT [7], and PLA [8] based approximate queries. However, even though the size of a synopsis is greatly compressed comparing to the original data, due to the continuous growing nature of a data stream, if we simply lay down each synopsis

one by one, the storage size will eventually limit the number of synopsis structures we can save and query.

In this paper, we propose **A Historical Stream Data Archive and Query Framework**, which includes the following technical contributions:

1. We show how different synopses can be easily embedded into the new Synopsis Cube Model and how to aggregate them over the time dimension. The new Synopsis Cube can answer fine level OLAP queries over historical data, which overcomes a major limitation of the Stream Cube model [9]. We design the aggregation algorithms for those widely known synopses: Fourier, DCT, Wavelet, and PLA.
2. We present the query technique over PLA and Fourier synopsis and compared the latter with the DCT query. Though it was widely conceived that DCT always performs better than Fourier, our experiments showed that Fourier can achieve better performance than DCT in some data sets.

**This paper is organized as follows.** Section 2 reviews the necessary background related to the paper. Section 3 to Section 5 detail our techniques. The experiments are reported in Section 6. Section 7 concludes the paper.

## 2 Background

### 2.1 Fourier/DCT Based Approximate Query

We refer readers to [10, 11] for Fourier Transformation families, including FFT, DFT, DCT, etc. In this paper, we use  $f(x)(x \in T)$  to denote stream value at time  $x$ . Then the Discrete Fourier Transform of  $f(x)$  is defined as:  $F(u) = \sum_{n=0}^{N-1} f(x)e^{-i2\pi ux/N}d(x)$ . J. Lee et al. [12] and M. Hsieh et al. [7] proposed using a small number of transformed DCT coefficients to represent original data stream and answer range queries.

### 2.2 Haar Wavelet

We use an example of 8 data points stream  $S = \{5, 7, 1, 9, 3, 5, 2, 4\}$  to show case how the Haar Wavelet transformation [3, 4, 6] works. There are always two types of Wavelet coefficients: the Average Coefficient and the Detail Coefficient. In each transformation step, every two adjacent data points are transformed to be one average coefficient and one difference value. The difference value is then saved to the end of whole stream as a detail coefficient. For example, the first two values  $\{5, 7\}$  of  $S$  become the first average coefficient  $\{6\}$  and the 5th detail coefficient  $\{-1\}$  in  $W_1 = \{6, 5, 4, 3, -1, -4, -1, -1\}$  after the first iteration. Then the same procedure is repeated until there are only one average value at the head and all the other detail coefficients at the end for different resolution levels, and we have  $W = \{4.5, 1, 0.5, 0.5, 1, -1, -4, -1, -1\}$ . We will refer to Haar Wavelet simply as Wavelet in the rest of this paper.

### 2.3 Piecewise Linear Approximation (PLA)

The Piecewise Linear Approximation (PLA, [8]) partitions the original stream data into multiple smaller streams clips, and approximates each clip by a straight line. Let  $S$  be

one stream clip and  $S = \{s_1, s_2, \dots, s_N\}$ . PLA uses a line segment  $s'_t$  of two parameters  $a$  and  $b$  to approximate the stream clip, i.e.  $S_t \approx s'_t = at + b$ . The equation 1 provides analytical formulas to calculate the two parameters  $a, b$  from the stream clip  $S$ .

$$a = \frac{12 \sum_{t=1}^N (t - \frac{(N+1)}{2}) s_t}{N(N+1)(N-1)}, \quad b = \frac{6 \sum_{t=1}^N (t - \frac{(2N+1)}{3}) s_t}{N(N+1)(N-1)} \quad (1)$$

## 2.4 Stream Cube and Tilted Time Model

To build an OLAP model on the stream data, Han et al. [9] proposed a Tilted Time Frame Dimension, along which the cell values are gradually aggregated from a higher resolution to a lower resolution. Fig 1 shows a natural tilted time window model [9]. In Fig 1 the values from the most recent 4 quarter hours are saved in a Stream

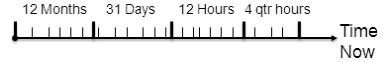


Fig. 1. Natural Tilted Time Window

Cube. Once we reach the beginning of the next hour, the 4 quarter hour cells are aggregated into only one cell in the hour level. Then all the incoming data for the next hour will be saved at the quarter hour level again. In this fashion, for one year data, we only need cells for the last 12 months, 31 days, 24 hours, and 4 quarter hours. As a result, there are only  $12 + 31 + 24 + 4 = 71$  cells for the entire year in the tilted time window model, which is a great saving in storage comparing with the naive MOLAP method demanding  $12 * 31 * 24 * 4 = 35,712$  cells.

## 3 The Synopsis Cube

We build our Synopsis Cube using Tilted Time Dimension [9]. According to Section 2.4, for the data of 5 years ago, only one sum value is saved for each year, even though in original stream, there could be millions of data points. We took for granted that this sum value should be the most important *feature* out of the whole data stream. Then, why not save a bit more information beyond a simple SUM value?

Now, in those distant cells, we extend the simple numerical measures to be some synopsis structures to save more than one features representing the original data falling in that cell. We call the new extended cell a *Synopsis Cell* and the Cube using this cell a *Synopsis Cube*. Any feature extraction method can be used to construct a synopsis cell, as long as we can design the two most important operators for it: *Aggregation*, and *Query*. In this paper, we construct our Synopsis Cells by those four methods: Fourier, DCT, Wavelet, and PLA.

To maximize the compatibility with the traditional OLAP cube, each synopsis cell (denoted by  $SC$ ) is defined as a tuple with five attributes:  $SC = \langle T, S, C, M, Syn \rangle$ . In a synopsis cell,  $T$  is the type of synopsis method used;  $S$  is the sum of all the data points in the original stream  $f(x)$ ;  $C$  is the count of all the data points;  $M$  is the number of coefficients saved into the synopsis cell, which is also the length of the next attribute  $Syn$ ;  $Syn$  is the saved most important coefficients from different synopsis methods.

$Syn$  for a Fourier, DCT or Wavelet synopsis cell is an array of the coefficient index and coefficient pairs, i.e.  $Syn = \{ \langle u_j, F(u_j) \rangle \mid j = 1, 2, \dots, M \}$ , while  $Syn$  for PLA synopsis cell is an array of parameter pairs, i.e.  $Syn = \{ \langle a_j, b_j \rangle \mid j = 1, 2, \dots, M \}$ .

## 4 Aggregating the Synopsis Cells

In the Stream Cube model, at the end of each time period (day, month, etc), lower level OLAP measures must be aggregated to a higher level one by certain arithmetic operations, like SUM. Likewise, in a Synopsis Cube, at the end of each time period, all the lower level synopsis cells should also be aggregated to a higher level synopsis cell.

We can support as many levels as end users want, but only two levels are involved in each aggregation operation. Let  $L1$  be the lower level, in which all stream clips are of a same length  $L_1$ , and  $L2$  be the higher level with length  $L_2$ . Let  $L = L_1 * L_2$  be the total length of the original stream data.

### 4.1 Aggregating Fourier Synopsis

We use  $F(u)$  to denote the lower level Fourier Coefficients and  $F'(u)$  for the aggregated Fourier coefficients on  $L2$ . Here the same  $u$  is used because the aggregated Fourier synopsis still represents the lowest level data as in the original Stream  $f(x)$ , no matter on which level this synopsis is aggregated. Let  $x$  be the index of the concatenated stream of all  $L1$  stream clips, then  $x$  can be expressed as  $x = zL_1 + y, z \in (1, 2, \dots, L_2), y \in (1, 2, \dots, L_1)$ . Similarly, let  $u$  be the frequency index of the concatenated  $L1$  frequency space. Then  $u$  can be expressed as  $u = zL_1 + w$ , in which  $w$  is the frequency index in corresponding  $z$ -th stream clip from  $L1$ . Let  $F_z(w)$  denote the Fourier coefficient at frequency  $w$  on the  $z$ -th stream clip from  $L1$ . Then the aggregated Fourier coefficient  $F'(u)$  can be approximately calculated by equation [2](#):

$$F_z(w) = \left( \sum_{y=1}^{L_1} f(zL_1 + y) e^{-2\pi i w \frac{y}{L_1}} \right)$$

$$F'(u) \approx \sum_{z=1}^{L_2} \left\{ e^{-2\pi i u \frac{z}{L_2}} \left( F_z(\text{floor}(\frac{u}{L_2})) \right) \right\} \quad (2)$$

The aggregation of DCT synopsis is similar to Fourier one, due to the fact that DCT is only the real part of Fourier Transformation[\[7\]](#). Therefore, we skip the aggregation formula for DCT synopsis.

### 4.2 Aggregating Wavelet

The aggregation of Wavelet Synopsis Cells is more straightforward than the Fourier/DCT one. There are two phases in the wavelet aggregation process: coefficient shuffle and wavelet transform. Since the Wavelet transform always happens on local blocks, the detail coefficients from  $L1$  can be used directly in  $L2$ . In the shuffle phase, we

move the Wavelet detail coefficients from  $L_1$  to the tailing positions of  $L_2$ , and the average coefficients from  $L_1$  to the head of  $L_2$ . Then, in Wavelet Transform phase, we simply apply the wavelet transformation algorithm from Section 2.2 again on all average coefficients, and concatenate the result with all shuffled  $L_1$  detail coefficients. Table 1 shows one example of this aggregation.

We can verify that the aggregated  $L_2$  coefficients are exactly same as directly transformed from original stream data. In other word, this is a lossless aggregation.

**Table 1.** Haar Wavelet Aggregation Demonstration

Step Name	Transformed Stream Data
Original Stream	$S_1 = \{5, 7, 1, 9\}, S_2 = \{3, 5, 2, 4\}, S_3 = \{8, 4, 5, 9\}, S_3 = \{1, 3, 7, 5\}$
$L_1$ Wavelet Synopses	$W_1 = \{5.5, -0.5, 1, 4\}, W_2 = \{3.5, -0.5, 1, 1\},$ $W_3 = \{6.5, 0.5, -2, 2\}, W_4 = \{4, 2, 1, -1\}$
After shuffle	$W = \{[5.5, 3.5, 6.5, 4], -0.5, -0.5, 0.5, 2, 1, 4, 1, 1, -2, 2, 1, -1\}$
Transform Again	$W = \{4.875, 0.3750, -1, -1.25, -0.5, -0.5, 0.5, 2, 1, 4, 1, 1, -2, 2, 1, -1\}$

### 4.3 Aggregating PLA

Equations 3, 4 give the formulas to calculate a high level line segment parameter pair  $\langle A, B \rangle$  from the low level parameters  $\{\langle a_p, b_p \rangle \mid p = 1, 2, \dots, L_1\}$ .

$$A = \frac{12}{L(L+1)(L-1)} \sum_{p=1}^{L_2} L_1 \left\{ (L_1 * (p-1) - \frac{L+1}{2}) \left( \frac{L_1+1}{2} a_p + b_p \right) + \frac{L_1^2}{3} a_p + \frac{L_1}{2} b_p \right\} \quad (3)$$

$$B = \frac{6}{L(1-L)} \sum_{p=1}^{L_2} L_1 \left\{ (L_1 * (p-1) - \frac{2N+1}{3}) \left( \frac{L_1+1}{2} a_p + b_p \right) + \frac{L_1^2}{3} a_p + \frac{L_1}{2} b_p \right\} \quad (4)$$

## 5 Querying the Stream Cube

A range query  $Q(m : n)$  can be answered over any of the four types of the synopsis cells directly, without reconstructing the original stream data. Technique from [12] can answer query  $Q_D(m_k : n_k)$  over a DCT synopsis cell, and the method from Vitter and Wang [3] can be deployed over a Wavelet synopsis cell. Equations 5 and 6 give the range query execution formulas over Fourier and PLA synopses respectively. Equations 5 and 6 are derived by integrating the Fourier sinusoid and PLA line functions.

$$Q_F(m_k : n_k) = \sum_{j=1}^M \left\{ \frac{F(u_j)}{i2\pi u_j} \left( e^{\frac{u_j}{N} i2\pi n_k} - e^{\frac{u_j}{N} i2\pi m_k} \right) \right\} \quad (5)$$

$$Q_P(m_k : n_k) = \frac{1}{2} a_k (n_k^2 - m_k^2) + b_k (n_k - m_k) \quad (6)$$

The similarity query can be executed over the saved coefficients in a similar manner as from the GEMINI framework [8, 13]. However, the Gemini framework can only work on a short period sliding window, while our Synopsis Cube can approximately save the Stream Data to a far distant history, and provide query capabilities.

## 6 Experiments

We validate our aggregation algorithms accuracy, and compare the synopsis cube performance using all the different synopsis methods. The experiments are conducted on various public real life datasets, including Mean monthly air temperature at Nottingham Castle, average wind speed at Ireland, DJI stock index (1980→2010), Power usage, Darwin sea level pressures, Buoy Sensor, Random Walk, etc [14]. The hardware configuration for all the experiments is Intel CPU U7600, 2GB Memory, running Windows XP system. All testing programs are coded and executed in Matlab 7.5. The testing program for Haar Wavelet query is implemented according to [3].

### 6.1 Range Query Accuracy over Aggregated Synopses Cells

**Setup:** We want to test the correctness of the aggregation algorithms from section 4. We tested three types of aggregated synopses: Fourier, Wavelet, and PLA. In this test we truncate and split each data stream into 32 smaller stream clips to simulate a stream data with 2 levels. Each lower level data stream contains 128 data points. We then compare the absolute query error on directly generated synopses from the original stream against the query error on the aggregated synopses from lower level ones.

**Discussion:** We tested all the data sets on hand, but due to text limitation, listed results on 3 streams only in Table 2. In this table, we see that the columns *Wavelet Direct* and *Wavelet Aggr* contain identical value, which means the query error are exactly same over the directly acquired synopsis and the aggregated synopsis. The value of column *PLA Aggr* is slightly higher than *PLA Direct*, but we believe that this difference is caused by the program boundary treatment. *Fourier Aggr* is ok on the Wind data (with strong periodical patterns), but too bad on the other two datasets.

**Conclusion:** This experiment confirmed correctness of our synopsis aggregation method. Also from the experiment, we can conclude that Wavelet and PLA shall work better than Fourier method in the aggregation operation.

**Table 2.** Range Query Error on aggregated synopsis

Data Set	Avg Real Value	Fourier Direct Error	Fourier Aggr Error	Wavelet Direct Error	Wavelet Aggr Error	PLA Direct Error	PLA Aggr Error
Stock	3.549	0.113	0.888	0.885	0.885	0.111	0.112
Buoy	-0.297	0.187	0.493	0.480	0.480	0.229	0.228
Wind	49.733	11.288	12.954	15.960	15.960	11.954	11.963

## 6.2 Minimum Balanced Cost of Different Synopses

**Setup:** On each different dataset, we want to test which synopsis method performs the best in our Synopsis Cube by comparing a balanced Cost  $c = \frac{E}{ME} + \frac{M}{N}$ . Here  $E$  is the average query error of a range query using  $M$  coefficients,  $ME$  is the maximum query error of all possible  $M$ , and  $N$  is the length of Stream Data. The minimum cost  $c$  of all possible  $M$  is recorded for each synopsis method and compared among the four different ones. This result can help select the best synopsis method on each dataset. Ding et al. already provided a comprehensive comparison on similarity query performance of different synopsis methods [15].

**Discussion:** We have tested the query errors on all 63 data sets [14]. Fig 2 shows the minimum cost from four different methods on all data sets. To easily compare minimum costs on all different datasets, we divide the four minimum costs of different synopses on each dataset by the maximum value among the four. In Fig 2, X-axis is the data set we tested, and Y-axis is the normalized minimum costs from the four methods. On each data set, exactly four points are plotted showing the costs, and the lowest one indicates the best synopsis method. In the 63 tested data streams, Fourier synopsis yielded the lowest cost for 7 times, DCT for 5 times, and PLA for 51 times. The Fourier and DCT methods outperform other methods on those data sets containing certain periodic patterns, like burst, ballbeam, etc.

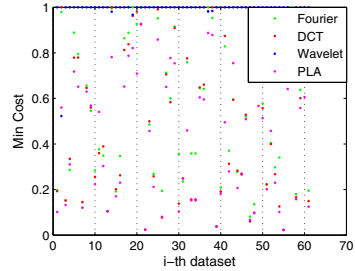


Fig. 2. Minimum Cost by four synopses

## 7 Conclusion

In this paper, we have shown an end-to-end solution of a Synopsis OLAP Cube using the Tilted Time frame and some well researched synopsis methods, including Fourier, DCT, Wavelet, and PLA. We devised necessary techniques to aggregate multiple lower level synopses into one higher level synopsis, and we also presented the range query techniques over each type of synopsis cells. Our experiments showed the performance comparison of different synopsis methods and also proved the correctness of our synopsis aggregation technique.

**Acknowledgement.** This work is supported in part by Chinese Tech Project 2010ZX01042-003-004, NSFC Key Program 61033010, Science and technology commission of shanghai municipality 10dz1511000, and IBM CRL UR project JSA201007005.

## References

1. Reeves, G., Liu, J., Nath, S., Zhao, F.: Managing massive time series streams with multi-scale compressed trickles. In: PVLDB, vol. 2(1), pp. 97–108 (2009), <http://www.vldb.org/pvldb/2/vldb09-434.pdf>

2. Aggarwal, C.C., Yu, P.S.: A Survey of Synopsis Construction in Data Streams, pp. 169–207. Springer, US (2007), <http://www.springerlink.com/content/wx435611v4678637/>
3. Vitter, J.S., Wang, M.: Approximate computation of multidimensional aggregates of sparse data using wavelets. In: SIGMOD Conference, pp. 193–204 (1999)
4. Chakrabarti, K., Garofalakis, M.N., Rastogi, R., Shim, K.: Approximate query processing using wavelets. VLDBJ. 10(2-3), 199–223 (2001), <http://link.springer.de/link/service/journals/00778/bibs/1010002/10100199.htm>
5. Yun-Bo Xiong, Y.-F.H., Liu, B.: Approximate query processing based on wavelet transform. In: Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, pp. 13–16 (2006)
6. Karras, P., Mamoulis, N.: The haar+ tree: A refined synopsis data structure. In: ICDE, pp. 436–445. IEEE, Los Alamitos (2007), <http://dx.doi.org/10.1109/ICDE.2007.367889>
7. Hsieh, M.-J., Chen, M.-S., Yu, P.S.: Approximate query processing in cube streams. IEEE Trans. Knowl. Data Eng. 19(11), 1557–1570 (2007), <http://doi.ieeecomputersociety.org/10.1109/TKDE.2007.190622>
8. Chen, Q., Chen, L., Lian, X., Liu, Y., Yu, J.X.: Indexable PLA for efficient similarity search. In: Proceedings of the 33rd International Conference on Very Large Data Bases, Austria, September 23–27, pp. 435–446. ACM, New York (2007)
9. Han, J., Chen, Y., Dong, G., Pei, J., Wah, B.W., Wang, J., Cai, Y.D.: Stream cube: An architecture for multi-dimensional analysis of data streams. Distributed and Parallel Databases 18(2), 173–197 (2005), <http://dx.doi.org/10.1007/s10619-005-3296-1>
10. Stein, E.M., Shakarchi, R.: Fourier Analysis I: An Introduction, pp. 134–140. Princeton University Press, Princeton (2003)
11. Smith.III, J.O.: Mathematics Of The Discrete Fourier Transform (DFT) With Audio Applications. W3K Publishing (2007)
12. Lee, J.-H., Kim, D.-H., Chung, C.-W.: Multi-dimensional selectivity estimation using compressed histogram information. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, pp. 205–214. ACM, New York (1999), <http://doi.acm.org/10.1145/304182.304200>
13. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: Proceedings of ACM SIGMOD, Minneapolis, MN, pp. 419–429 (1994)
14. Duan, Q.: Stream data collection (2011), <http://sites.google.com/site/qiyangduan/publications/stream-data-set>
15. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.J.: Querying and mining of time series data: experimental comparison of representations and distance measures. In: PVLDB, vol. 1(2), pp. 1542–1552 (2008), <http://www.vldb.org/pvldb/1/1454226.pdf>



# An Incremental Approach to Closest Pair Queries in Spatial Networks Using Best-First Search

Chunan Chen<sup>1</sup>, Weiwei Sun<sup>1,\*</sup>, Baihua Zheng<sup>2</sup>, Dingding Mao<sup>1</sup>, and Weimo Liu<sup>1</sup>

<sup>1</sup> School of Computer Science, Fudan University, Shanghai, China  
{chenchunan, wwsun, maodingding, liuweimo}@fudan.edu.cn

<sup>2</sup> Singapore Management University  
bhzheng@smu.edu.sg

**Abstract.** This paper addresses the problem of  $k$  Closest Pairs ( $k$ CP) query in spatial network databases. A Best-First search approach namely BFCP (Best-First Closest Pair) is proposed. Given two data sets of objects in a spatial network, BFCP first finds the 1st CP by computing the 1st NN (nearest neighbor) of each object in the set with smaller cardinality. Then BFCP retrieves the 2nd, 3rd, ...,  $k$ th CP in an incremental way by searching the next NN of the currently found CP's source point. Furthermore, a novel buffer replacement policy called MDU (Minimum Distance Unit) is proposed to reduce I/O cost of BFCP. Unlike LRU, which records only the last reference time, the MDU policy considers both temporal locality and spatial locality when selecting a buffer page as the victim. A comprehensive experimental study is conducted to demonstrate the advantage of BFCP and MDU.

**Keywords:** Closest Pair, Spatial networks, Location-based services, Buffer management.

## 1 Introduction

Spatial databases have attracted lots of attentions in the past decade with the advances in mobile computing and popularity of location-based services. In this paper, we focus on  $k$  Closest Pair ( $k$ CP) query in a Spatial Network Database (SNDB), which is commonly used in real applications related to traffic monitoring and route planning in city road map. Given two datasets  $S$  and  $T$ , object pairs  $(s_i, t_i) \in S \times T$  are sorted based on non-descending order of the network distance between  $s_i$  and  $t_i$  and a  $k$  Closest Pair query is to return  $k$  pairs of objects with minimal distances.  $k$ CP query is a common query in SNDB, and is widely used in many applications such as GIS. For example, consider a businessman who wants to take one day off his busy schedule to visit some place of attraction in Shanghai. In order to save time, he wants to stay in a hotel that is near to some place of attraction. A  $k$ CP query can be issued to find  $k$  pairs of scenic spot and hotel.

$k$ CP queries have been well studied in Euclidean spaces, where the Euclidean distance between two objects is employed to measure the distance between them.

---

\* Corresponding author.

Typically, objects are stored in two R-trees [1] and they are traversed by branch-and-bound search approaches [2]. However, the distance between two objects in SNDB is determined by their shortest path but not the Euclidean distance. To tackle this issue, two techniques, namely CPER (Closest Pair Euclidean Restriction) and CPNE (Closest Pair Network Expansion) are proposed [3]. CPER uses the Euclidean distance as the lower bound of the network distance between two points. It first employs the R-tree based  $kCP$  query algorithm to get the candidates of the  $kCP$  in spatial network and further finds the exact network distance of each candidate pair to confirm the final result in the refinement step. This approach suffers from poor performance when there is a big difference between the Euclidean distance and network distance, which is of frequent occurrence in reality. CPNE finds the  $kNN$  of each object in  $S \cup T$  based on network expansion. It then uses the maximum distance of the current  $kCP$  result as the upper bound for pruning; however, this upper bound might be very loose. As a result, CPNE spends time to search for many unnecessary NNs in the  $kCP$  search procedure and the cost of this approach is very expensive. Another existing solution to  $kCP$  query in SNDB is the Top- $k$  Distance Join algorithm introduced in [4]. It pre-computes the shortest path between each pair of nodes in a spatial network and stores all the paths in a quad-tree, which suffers from additional storage cost and is not suitable in cases that the network structure changes frequently.

In this paper, we study the problem of  $kCP$  query when the sizes of the two data sets are of great difference and/or the value of  $k$  is very large. To the best of our knowledge, CPNE requires the least CPU time and I/O cost among all the existing work. However, the search performance of  $kCP$  query can be still improved without pre-computation. The contributions of this paper can be summarized as follows:

- An efficient incremental approach called BFCP (Best-First Closest Pair) is proposed. Like CPNE, BFCP finds the  $kCP$  by computing the  $kNN$  of the objects in one of the given sets. Unlike CPNE, BFCP uses a best-first search to find the  $kCP$  in an incremental way so that it saves the computations of some unnecessary NNs. We conduct a theoretical analysis and extensive experimental study to demonstrate the advantage of BFCP over CPNE.
- A novel buffer replacement policy called Minimum Distance Unit (MDU) is proposed. It considers both the temporal locality and spatial locality of buffer pages when selecting a victim, which further boosts up the I/O cost of BFCP.

The rest of the paper is organized as follows. Section 2 overviews  $kNN$  queries and  $kCP$  queries in SNDB. Section 3 and Section 4 present the BFCP algorithm and the MDU buffer replace policy. Section 5 reports the experimental study results, and Section 6 concludes our work and discusses some future work.

## 2 Related Work

In this section, we briefly review existing works related to  $kCP$  query processing in SNDB, including disk-based storage schema for spatial network,  $kNN/kCP$  query in SNDB, and buffer management strategy.

## 2.1 Disk-Based Storage Schema of Spatial Network

A spatial network usually is modeled as a graph  $G$  consisting of vertex set  $V$  and edge set  $E$ . In most of the real-world applications, a spatial network normally contains at least thousands of vertices, and hence it is not practical to process all executions in the main memory. The main challenge of the disk storage of a network graph is how to cluster the adjacent nodes in the same page to reduce the I/O cost when accessing a graph. In the literature, different approaches have been proposed, including the connectivity-cluster access method (CCAM) [5] and Hilbert-curve based method [3]. Due to the simplicity and optimal locality of Hilbert curve, we employ the Hilbert-curve-based network storage schema to process  $kCP$  queries in this paper and further discuss how to reduce I/O cost by improving the performance of the buffer manager.

## 2.2 $kNN$ Query and $kCP$ Query in SNDB

Given a query point  $q$  and a set of objects  $S$ , a  $kNN$  query is to find the  $k$  closest objects in  $S$  to the query point  $q$ . Several algorithms have been proposed to support  $kNN$  query in spatial network. [3] introduces Incremental Euclidean Restriction (IER) and Incremental Network Expansion (INE). Some pre-computation based solutions to  $kNN$  query are proposed in [7-11]. Closest pair query is another important query in spatial databases. A well-known solution to  $kCP$  queries in Euclidean spaces is the heap algorithm proposed in [2]. In order to support  $kCP$  query in SNDB, CPER and CPNE are proposed in [3]. The Top- $k$  Distance Join algorithm in spatial networks proposed in [4] is a technique based on the SILC framework introduced in [9].

## 2.3 Buffer Replacement Policy

Efficient buffer management (e.g., LRU (Least Recently Used) [12] and LRU-K [13]) is an effective technique to reduce the I/O cost. Some page replacement policies for spatial databases have been proposed in recent years such as LRD-Manhattan [14] and a self-tuning page replacement strategy [15]. Like the self-tuning page replacement policy, MDU proposed in this paper selects the victim based on both the temporal locality and spatial locality. The innovation of MDU is that it explores the spatial locality in a network environment and works efficiently in our BFCP algorithm.

# 3 A Best-First $kCP$ Algorithm

## 3.1 Problem Definition and Search Algorithm

In this paper, we model the spatial network as a graph and use the Hilbert-curve based schema introduced in [3] to store the graph in the disk. The network distance of two nodes  $v$  and  $u$  is noted as  $D(v, u)$ , and a pair  $\langle s_i, t_i \rangle \in S \times T$  is denoted as  $p_j$ .  $kCP$  query is formally defined in Definition 1.

**Definition 1.**  $k$  Closest Pair ( $k$ CP) query. Given a network  $G(V, E)$  and two data sets of objects  $S$  and  $T$  ( $S, T \subseteq V$ ), a  $k$ CP query is to find the  $k$  pairs  $p_j \in S \times T$  such that  $\forall p_j \in P = \{p_1, p_2, \dots, p_k\}$ ,  $\forall p_i \in S \times T - P$ ,  $D(p_j) \leq D(p_i)$ .  $\square$

The  $k$ CP of  $S$  and  $T$  can be evaluated incrementally by computing the 1<sup>st</sup> NN from  $T$  for each point in  $S$  and searching the next NN of the current CP iteratively. We call this approach a ‘‘Best-First’’ approach because it only finds the next NN of the current CP retrieved during the search procedure. The pseudo code of BFCP is shown in Algorithm 1. It uses a priority queue  $Q$  to store the candidate pairs, which is a memory-based data structure and sorts the candidate pairs in a min-heap according to the pairs’ distance. First, BFCP retrieves the 1<sup>st</sup> NN for each object in  $S$  based on the INE approach proposed in [3]. All the pairs found in this step will be inserted into the priority queue (lines 2-5). Second, BFCP finds the 1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup>, ...,  $k^{\text{th}}$  CP in an incremental way (lines 6-12). In each iteration, the head pair  $\langle s, t \rangle$  is popped out as the next CP and the next NN of  $s$  is retrieved to form a new entry in  $Q$ .

---

**Algorithm 1:** BFCP( $S, T, k$ )

---

**Input:**  $G(V, E)$ , Set  $S$  and  $T$  ( $|S| < |T|$ ), an integer  $k$ .

**Output:**  $k$ CP of  $S$  and  $T$ .

1.  $count \leftarrow 0$ ,  $Q \leftarrow \text{NewPriorityQueue}()$
  2. **for** each object  $s_i$  in the set  $S$  **do**
  3.      $s_i.\text{1NN} = \text{INE\_1NN}(s_i)$
  4.      $\text{ENQUEUE}(Q, D(s_i, s_i.\text{1NN}), \langle s_i, s_i.\text{1NN} \rangle)$
  5. **enddo**
  6. **while** ( $count \leq k$ ) **do**
  7.      $\langle s, t \rangle \leftarrow \text{DEQUEUE}(Q)$
  8.     Report the pair  $\langle s, t \rangle$  as the next closest pair
  9.      $count \leftarrow count + 1$
  10.      $t_{\text{next}} \leftarrow \text{INE\_NextNN}(s)$
  11.      $\text{ENQUEUE}(Q, D(s, t_{\text{next}}), \langle s, t_{\text{next}} \rangle)$
  12. **enddo**
- 

## 4 The MDU Buffer Replacement Policy

In section 3, we propose BFCP algorithm to reduce the number of NN computations. In this section, we introduce a new buffer management strategy, namely *Minimum Distance Unit* (MDU), to further cut down the I/O cost of BFCP. In the following, we first explain the issue of employing the LRU buffer replacement policy in BFCP, and then we describe the MDU buffer replacement in detail.

As mentioned in section 2.1, we use Hilbert-curve-based network storage schema to store the road network in disk, which tries to cluster the adjacent nodes in the same page. When a graph operation needs to access the information of a node  $u$ , the page containing  $u$  is loaded into the main memory. This page will remain in the buffer till it is replaced. Without loss of generality, we assume LRU is the default buffer management policy. For a given object  $s_i \in S$ , BFCP will load the page  $p_0$  containing the 1<sup>st</sup> NN of  $s_i$  when retrieving  $s_i$ ’s nearest neighbor. However, the page  $p_0$  has a high

chance to be replaced by LRU while BFCP retrieves the 1<sup>st</sup> NN of other objects in  $S$ . However, if the distance between  $s_i$  and its 1<sup>st</sup> NN is very small, BFCP needs to find out the 2<sup>nd</sup> NN of  $s_i$ , which is probably located in the page  $p_0$ , due to the spatial locality of the storage schema. Consequently, LRU might not be a proper buffer management strategy for BFCP.

In order to design a policy to reduce the I/O cost of BFCP, we propose MDU policy. It associates a “distance” parameter with each page  $p$  and is defined as follows. Given a page  $p$  loaded into memory when searching for  $j$ -th NN (i.e.,  $s.j$ NN) of object  $s \in S$ , the distance of  $p$  is set to the network distance between  $s$  and  $s.j$ NN.

$$dis(p) = D(s, s.j$$
NN)

In addition, the last reference time of each buffer page is still useful. For example, consider a page  $p_0$  which contains only a source point  $s$  and its 1<sup>st</sup> NN. Page  $p_0$  might have a very small distance value, but it will not contribute to the search other than searching  $s.1$ NN. Because of its small  $dis(p)$  value, it will not be replaced. Consequently, we also record the last reference time of each buffer page and hence those pages that have small distance but have not been accessed for a long time could be identified and replaced as well.

Therefore, MDU records two attributes for each buffer page  $p$ : the last reference time  $t(p)$  and the distance  $dis(p)$ . If a page replacement is required at the time of  $t_{now}$ , it selects the page having largest  $TD$  value, with  $TD$  defined in the following.

$$TD(p) = \lambda \times \frac{t_{now} - t(p)}{t_{now}} + (1 - \lambda) \times \frac{dis(p)}{dis_{max}} \quad 0 \leq \lambda \leq 1$$

Notice that parameters  $t_{max}$  and  $dis_{max}$  are used to normalize the last reference time  $t(p)$  and the distance  $dis(p)$  as they have different scales. Here  $t_{max}$  is set to  $t_{now}$ , the largest last access time of all the buffer pages, and  $dis_{max}$  is set to the largest distance  $dis(p)$  of all the buffer pages. The parameter  $\lambda$  is a tradeoff between the spatial locality and temporal locality, when selecting a page to be replaced in the MDU policy. The effect of  $\lambda$  will be discussed in section 5.

## 5 Experimental Evaluation

In this section, we present the experimental study conducted to evaluate the performance of BFCP and other existing solutions. The real road network of California [16], consisting of 21,048 nodes and 21,693 edges, is used as the sample dataset. The nodes of graph correspond to the junctions of the road network. The graph was stored on disk according to the storage schema introduced in [3]. The page size is set to 4KB and the buffer size of both LRU and MDU is set to 10% of the size of the data set on disk. As  $k$ CP query involves two datasets, we randomly sample the nodes from the road networks to form datasets  $S$  and  $T$ , with their ratio (i.e.,  $|S|/|T|$ ) controlled by a parameter. All the experiments are run on a Genuine 1.8GHZ CPU desktop computer with 3GB memory. For each experiment, we execute 200 queries and report the average result.

In section 5.1, we compare the performance of BFCP with CPNE, which is known to incur the smallest CPU time and I/O cost when the datasets  $S$  and  $T$  have very different cardinalities among all the online-computing algorithms for the  $k$ CP query problem. In section 5.2, we explore the performance of MDU and compare it with LRU, the most widely used page replacement policy.

### 5.1 Performance Comparison of BFCP and CPNE

We evaluated the performance of BFCP and CPNE with respect to the number of NNs computations, CPU time and I/O cost, with different properties such as the size of the two object sets and the value of  $k$ . As mentioned previously, we focus on the cases where the cardinality difference between data sets  $S$  and  $T$  is large and/or  $k$  is large. Consequently, the size of  $S$  varies from  $0.0025|M|$  to  $0.01|M|$ , while that of  $T$  ranges from  $0.025|M|$  to  $0.1|M|$ , with  $|M|$  the number of the nodes in the road network. The value of  $k$  ranges from 100 to 250. Notice that in this set of experiments, LRU is employed as the default buffer management policy for both BFCP and CPNE. The impacts of the three parameters on the performance of the two algorithms are shown in Figures 1 to 3. As we can see in the figures, BFCP outperforms CPNE in both CPU time and I/O cost, for the number of NNs computations (denoted as #NN) incurred in BFCP is far less than that incurred in CPNE.

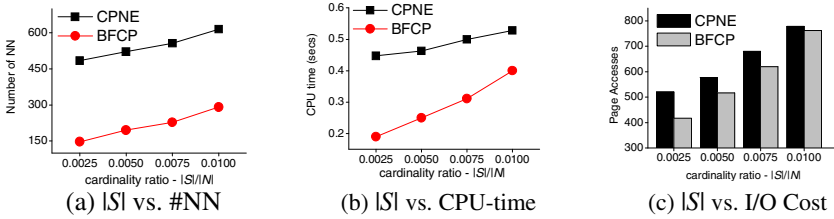


Fig. 1. The Impact of  $|S|$  ( $|T|=0.1|M|$ ,  $k=100$ )

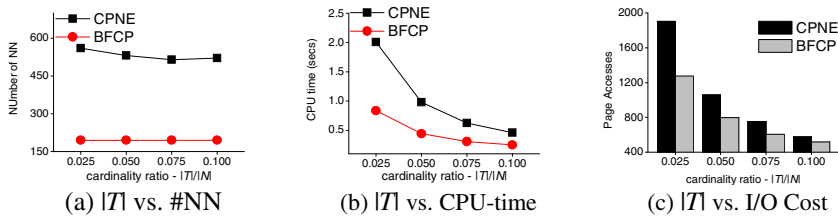


Fig. 2. The Impact of  $|T|$  ( $|S|=0.01|M|$ ,  $k=100$ )

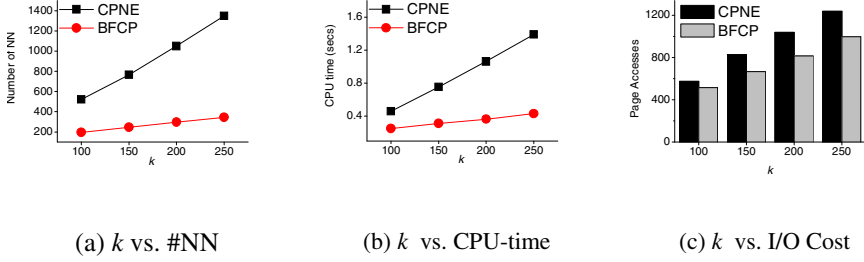


Fig. 3. The Impact of  $k$  ( $|S|=0.01|N|$ ,  $|T|=0.1|N|$ )

### 5.2 Performance Comparison of MDU and LRU

In the second set of experiments, we evaluate the performance of MDU buffer replacement policy, compared with LRU.

**Impact of  $\lambda$ .** We first evaluate the effect of parameter  $\lambda$  on the performance of MDU. It is observed that MDU with  $\lambda = 0.4$  has the best performance (see Figure 4(a)).

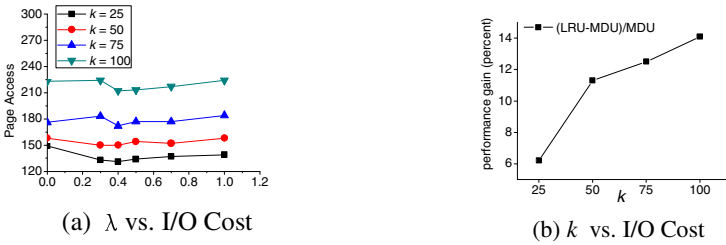


Fig. 4. The Impact of  $\lambda$  ( $|S|=0.01|N|$ ,  $|T|=0.1|N|$ )

MDU and LRU. Next, we compare MDU with  $\lambda = 0.4$  against LRU under different  $k$  values. We report the performance gain, as defined in the following:

$$\text{performance gain} = \frac{\text{BFCP-LRU} - \text{BFCP-MDU}}{\text{BFCP-MDU}} \times 100\%.$$

As shown in Figure 4(b), MDU outperforms LRU by reducing the false hits significantly. For example, when  $k = 100$ , MDU achieves a performance gain of about 15%.

## 6 Conclusion

In this paper, we present a Best-First solution (BFCP) to  $k$ CP queries in SNDB. In addition, a novel buffer replacement policy is proposed to reduce the false hits in our

algorithm. The performance of BFCP and MDU is tested on a real world network. The results show that BFCP performs well, and MDU buffer replacement policy can further improve its performance. We plan to explore the to monitoring of  $k$ CPs in a dynamic network environment in the future. One possible solution is to maintain a minimum spanning tree of each pair and update the result of  $k$ CPs through rebuilding the structure of the minimum spanning trees.

**Acknowledgment.** This research is supported in part by the National Natural Science Foundation of China (NSFC) under grant 61073001.

## References

1. Guttman, A.: R-trees: A Dynamic Index Structure for Spatial Searching. In: SIGMOD (1984)
2. Corral, A., Manolopoulos, Y., Theodoridis, Y., Vassilakopoulos, M.: Closest Pair Queries in Spatial Databases. In: SIGMOD (2000)
3. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query Processing in Spatial Network Databases. In: VLBD (2003)
4. Sankaranarayanan, J., Samet, H.: Distance Join Queries on Spatial Networks. In: GIS (2006)
5. Shekhar, S., Liu, D.: CCAM: A Connectivity-Clustered Access Method for Networks and Network Computations. TKDE 19(1), 102–119 (1997)
6. Mouratidis, K., Yiu, M.L., Papadias, D., Mamoulis, N.: Continuous Nearest Neighbor Monitoring in Road Networks. In: VLDB (2006)
7. Kolahdouzan, M., Shahabi, C.: Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases. In: VLDB (2004)
8. Shahabi, C., Kolahdouzan, M., Sharifzadeh, M.: A Road Network Embedding Technique for k Nearest Neighbor Search in Moving Object Databases. Geoinformatica 7(3), 255–273 (2003)
9. Samet, H., Sankaranarayanan, J., Alborzi, H.: Scalable Network Distance Browsing in Spatial Databases. In: SIGMOD (2008)
10. Sankaranarayanan, J., Samet, H.: Distance Oracles for Spatial Networks. In: ICDE (2009)
11. Sankaranarayanan, J., Samet, H.: Path Oracles for Spatial Networks. In: VLDB (2009)
12. Effelsberg, W.: Principles of Database buffer Management. ACM TODS 9(4), 560–595 (1984)
13. O’Neil, E.J., Neil, P.E.O., Weikum, G.: The LRU-K Page Replacement Algorithm for Database Disk Buffering. In: SIGMOD (1993)
14. Papadopoulos, A., Manolopoulos, Y.: Global Page Replacement in Spatial Databases. In: DEXA (1996)
15. Brinkhoff, T.: A Robust and Self-tuning Page Replacement Strategy for Spatial Database Systems. In: DEXA (2002)
16. Digital Chart of the World Server, <http://www.maproom.psu.edu/dcw/>



# Fast Top-K Query Answering

Claus Dabringer and Johann Eder \*

Alps Adria University Klagenfurt, Department of Informatics Systems  
{Claus.Dabringer, Johann.Eder}@aau.at

**Abstract.** Efficient retrieval of the most relevant (i.e. top-k) tuples is an important requirement in information management systems which access large amounts of data. In general answering a top-k query request means to retrieve the  $k$  objects which score best for an objective function. We propose some improvements to the *best position algorithm (BPA-2)* [2]. To the best of our knowledge BPA-2 is currently the fastest available top-k query answering approach based on the widely known and applied Threshold Algorithm (short TA) of Fagin et al. [5]. Our proposed improvements lead to significantly reduced time and memory consumption and better scalability compared to BPA-2: (1) we dynamically create value rather than object based index structures out of the query restrictions posed by the user, (2) we introduce look-ahead techniques to process those index structures. While BPA-2 processes all pre-calculated indexes in parallel we always examine the most promising indexing structure next. We prototypically implemented our *fast top-k query answering (FTA)* approach. Our experiments showed an improvement by one to two orders of magnitude over BPA-2.

**Keywords:** Top-K Query Answering, Approximate Querying, Result Ranking.

## 1 Introduction

In many application areas like search in real-estate databases, product catalogs or scientific databases (such as biobanks [4]) we are frequently confronted with overspecified queries, where users do not expect that all restrictions are satisfied but they rather state their preferences or *ideal* query results. In traditional query evaluation such queries return few or no results missing acceptable answer. Users have to repeat the queries slightly relaxing the restrictions several times to satisfy their information need. Top-k queries provide an approach to efficiently answer such queries: They return the best  $k$  tuples according to an objective function. The efficient retrieval of the top-k tuples matching the ideal object described through a query without scoring each object in a database has attracted a lot of research attention (see [7] for an overview). In this paper we focus on monotonous objective functions, in particular on functions which are monotonous in each

---

\* The work reported here was supported by the Austrian Ministry of Science and Research within the program GENAU - project GATIB II.

attribute. We often see such functions when the overall score is expressed as (weighted) sum or product of the degrees to which each restriction is fulfilled. In [3] we showed that the usage of appropriate indexing techniques can speed up query answer times by a factor of two to five. This work is a further development of [3]. The major contributions of this work are:

- Improved performance for top-k queries by one or two orders of magnitude.
- The query answering process is improved with the help of a novel look-ahead technique which always process the most promising index next.
- FTA basically uses a heuristic to determine the next index to examine.

Our algorithm can be classified according to the categorization for top-k approaches introduced by Ilya et al. in [7] as follows: *Query Model*: top-k selection and top-k join, *Data & query certainty*: certain data, exact methods, *Data access*: both sorted and random, *Implementation level*: application level, *Ranking function*: monotone.

The rest of the paper is organized as follows: In section [2] we give a short overview on related work and point out areas where the different approaches can be further improved. We present our approach in section [3]. In section [4] we take a detailed look into the performance of our approach and evaluate it against the best position algorithm BPA-2. In section [5] we draw some conclusions.

## 2 Related Work

The well known and widely applied Threshold Algorithm of Fagin et al. (in short TA) [5] uses a *pre-calculated index structure* for evaluation of similarity. It basically assumes  $m$  grades for each object in a database where  $m$  is the number of different attributes. Fagin assumes for every attribute a sorted list which stores each object and its similarity under a certain attribute value. TA processes the given lists row by row and maintains a buffer of objects most similar to the ideal object specified by the input query. Since the lists are ordered TA stops in case  $k$ -objects have been identified and no unseen object has the chance of achieving a higher score than the object with the lowest score stored in the buffer. The fastest known approach based on TA's idea is BPA-2 [2]. It relies on the same assumptions as TA does, but incorporates an earlier stopping condition. For each sorted list BPA-2 maintains the so called *best position*, i.e. the greatest position such that any lower position in the list has already been examined. In each step BPA-2 examines the next position after the best position in each list and does random access to all other lists to find the similarity values of the processed object there. With that technique BPA-2 achieves two improvements over TA: (1) it avoids duplicate access to one and the same object and (2) it can stop much earlier than TA does. In [2] the authors showed that BPA-2 can improve query answering performance of TA up to eight times.

In [3] we already showed that the usage of different indexing techniques can improve query answering times and memory requirements by a factor of two to five. We generate an index entry for each distinct attribute value and proposed a

parallel stepwise processing of the generated indexes. Nevertheless, this parallel stepwise processing of the index lists leads to examining many objects which do not make it in the answer set. Therefore, we propose a novel look-ahead technique and process always the most promising index structure next, i.e. our heuristic is to choose the index structure with the highest similarity value in the next position.

### 3 Top-K Query Answering

Our *fast top-k query answering (FTA)* approach is able to find the top-k objects that satisfy the restrictions of an input query best. The approach is generic and works not only on single relations but also on whole databases.

**Supported Queries.** We support a great range of queries which may contain an arbitrary number of attributes from different tables as well as restrictions affecting attributes from different tables. The restrictions can be very versatile e.g. we are supporting any binary operator available in standard SQL as well as range operators like *BETWEEN*. Restrictions composed of binary operators consist of a left-hand attribute, an operator and a right-hand attribute or value. An example restriction would be: *age = 30*. Additionally we introduce the operator *'~'*. All restrictions formulated with *'~'* as its operator are treated as soft restrictions. Thus they are used for searching similar objects in case not enough objects satisfied the strict restrictions.

**Necessary Functions.** One of the basic concepts of *FTA* is its flexibility since users may customize and thus affect the ranking by providing a user defined *similarity and an objective function* to the approach. *FTA* basically needs these two functions to find the most interesting objects in a database. The functions can be adapted by the user to reflect their needs appropriate, but *FTA* also works out of the box since it is equipped with two standard functions. For *FTA* the *similarity function* must be able to calculate the similarity between two values of an attribute in a relation. The *objective function* is used by *FTA* to calculate the rank or score of objects. It should reflect how well an object fulfils the given restrictions, i.e. how similar an object's attribute values are to the values in the restrictions. These two functions can be passed to *FTA* in an initialization step which precedes the actual searching for interesting objects. A concrete example for both functions is given in section 4 where we instantiate the *FTA* approach.

#### 3.1 How FTA Works

In contrast to traditional top-k approaches [6,5,9,2] which maintain preprocessed index lists only on attribute level, *FTA* uses the provided similarity function to efficiently retrieve a sorted similarity list (i.e. index) for any distinct attribute value. Using the given similarity function our approach is able to generate similarity lists during query processing. Since the index contains distinct attribute values only, the memory requirements of the *FTA* approach are proportional to

the selectivity of the restricted attributes. This leads to a significant speedup compared to approaches which calculate similarity measures for each object under each attribute [3]. Here we additionally introduce a novel look-ahead technique, which always processes the most promising index next.

Basically the *FTA* approach can be divided into two phases, (1) an *initialization phase* and (2) a *processing phase*. Within the *first phase* *FTA* is initialized with two functions, namely the *similarity function* and the *objective function*. The further *FTA* creates an *index lookup structure* based on the given restrictions. The *second phase* processes the index lists and fetches new objects until no more objects are available or enough objects have been published. The following piece of pseudo-code shows the process of obtaining the top-k objects for a given list of restrictions.

```

program FTA (IN string tableName, IN Set restr, IN Number K,
            IN Func Sim, IN Func Objective, OUT Set<object> objs)
  var idx: Set<LookupTable>, buf: Set<object>, pubCnt: Number
begin
  SetFunctions(Sim, Objective);
  CreateIndexTables(restr, OUT idx);

  while pubCnt < K and objects available
    fetchObjects(tableName, idx, IN_OUT buf);
    pubCnt := publishObjects(buf, idx, K, IN_OUT objs);
  end-while
end.

```

The approach basically needs five input parameters. (1) The *tableName* is the name of a certain relation containing the data to be searched. If more than one table is involved in a top-k request we first generate a view joining together all needed tables. (2) The restrictions which should be satisfied by the result tuples. (3) The amount specifying the number of objects returned by the algorithm is given via parameter *K*. (4) The reference to the similarity function which should be used. (5) A reference to the objective function used for scoring the tuples in the database. The output produced by the *FTA* approach is a sorted list of *K*-objects. The search loop stops if *K*-objects have been published or in the case there are no more objects in the database. In the following we take a deeper look into the major parts of the *FTA* approach. We assume a relation *R* which contains attributes *a1* (*number of seats*), *a2* (*type of car*).

**Generation of index lookup tables.** The final step in the initialization phase is the generation of a set of index lookup tables for each restriction in the input set. *FTA* creates all needed indexes during query processing very fast. Each of the indexes contains all distinct values of a certain attribute and its similarity to the restriction specified by the user. All indexes are ordered descending by the similarity, i.e. the best match is on top. The similarity is calculated using the similarity function specified in the first step of the initialization. In example *ex-1* we assume the user specified the following restrictions as input parameters

**Table 1.** Generated index structures for example ex-1 sorted descending

(a) Similarity to value 5 in R.a1	(b) Similarity to value 'Sedan' in R.a2																				
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr><th style="width: 20%;">a1</th><th>Similarity</th></tr> </thead> <tbody> <tr><td>5</td><td>1</td></tr> <tr><td>6</td><td>0,95</td></tr> <tr><td>4</td><td>0,8</td></tr> <tr><td>...</td><td>...</td></tr> </tbody> </table>	a1	Similarity	5	1	6	0,95	4	0,8	...	...	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr><th style="width: 20%;">a2</th><th>Similarity</th></tr> </thead> <tbody> <tr><td>Coupe</td><td>0,85</td></tr> <tr><td>Van</td><td>0,4</td></tr> <tr><td>SUV</td><td>0,3</td></tr> <tr><td>...</td><td>...</td></tr> </tbody> </table>	a2	Similarity	Coupe	0,85	Van	0,4	SUV	0,3	...	...
a1	Similarity																				
5	1																				
6	0,95																				
4	0,8																				
...	...																				
a2	Similarity																				
Coupe	0,85																				
Van	0,4																				
SUV	0,3																				
...	...																				

for our *FTA* approach: (1)  $a1 \sim 5$ , (2)  $a2 \sim \text{sedan}$ . Generating the index structures for example *ex-1* results in an output like shown tables [1a](#), [1b](#). Within *CreateIndexTables* all generated indexes are collected and returned to the *FTA*-method. The created index tables allow a fast locating of interesting objects in the processing phase which is described next.

**Look Ahead.** After the generation of an index structure for each restricted attribute the lists are processed top-down, i.e. row by row from highest similarity to the lowest. In contrast to BPA-2 and Fagin's TA the *FTA* approach does not process all lists in parallel. It maintains a separate current row-number  $r_i$  for each index  $i$ . With the help of the row-numbers *FTA* looks ahead to identify the most promising index: i.e the index  $i$  for which  $i[r_i + 1]$  is highest.

**Fetching Objects.** Within *fetchObjects* *FTA* makes use of the described *look-ahead technique* to find the most promising index. It iterates over all available indexes and searches the one with the highest similarity value in the next row. Afterwards the best index is used to build a query and to fetch the next objects. Finally all found objects are added to a buffer, iff the objects are not already contained in it. For all added objects *FTA* computes the *total score* the *objective function*. The function *fetchObjects* is called repeatedly by *FTA* and always chooses the best index structure to search for objects. Assuming table [1a](#) and table [1b](#) as the lookup structures *FTA* searches for objects with the following attributes:

- Call-1: value 5 on attribute R.a1
- Call-2: value 6 on attribute R.a1
- Call-3: value 'Coupe' on attribute R.a2
- ...

With the help of this look-ahead technique the *FTA* approach can ensure that it always processes the most promising index next. This look-ahead technique prevents from fetching objects of indexes with very low similarity values. Thus the top-k objects can often be found without stepping down indexes that contribute too little to the overall score w.r.t the objective function. Due to this intelligent fetching *FTA* heavily utilizes the interesting indexes while it spares the processing of minor interesting ones.

**Calculating the score of objects.** FTA builds a list of similarity values for a certain object with the help of the lookup tables. Since these index structures store all distinct attribute values and their similarity values FTA can directly obtain the similarity for each restricted attribute without a query against the database. With the help of the user given *objective function* FTA calculates the score for the list of similarity values. The calculation of the objects' score is applied two times within FTA. (1) It is used to calculate the score of all objects gathered within *fetchObjects*. The obtained score is then used within *publishObjects* to decide whether to publish an object or not. (2) In *publishObjects* it is used to calculate the maximum achievable score of all unseen objects. This is done by creating an ideal object out of current rows from the lookup tables and calculating its score. The maximum achievable score is used to publish objects that have a higher score than all unseen objects.

**Publishing Top-K objects.** An object can be published (i.e. returned to the user), if it is certainly in the answer set, i.e. the object is for sure within the top-k objects for a query with the given restrictions. At the beginning of *publishObjects* the score of a hypothetical best matching object is computed. Therefore, an object with the values which are located in the row to be processed next is created. After that the score of this object is calculated with the help of the *objective function*. Since all indexes are ordered by decreasing similarity and the objective function is monotonous we can conclude that each *unseen object* for sure achieves a score which is lower or equal to the score of the hypothetical object. This stems from the fact that each unseen object is located at least one row after the current in at least one of the lookup tables. See [5] for a theoretical justification for this publishing process.

## 4 Prototype and Experiments

*FTA* is a generic approach giving the possibility to define a similarity and an objective function to influence the scoring mechanism. In section 3 we primarily focused on how these two functions are used within *FTA* without talking about concrete functions. Below we present two potential candidates for these functions.

**Similarity Function(s).** When measuring the similarity between attribute values we distinguish between numerical values and categorical data. One possibility to calculate the similarity between two numerical values is the usage of the *Euclidean Distance* [8]. It models the similarity in a way that the farther two values are apart from each other, the smaller their similarity is. One possible similarity function for categorical data is the inverse document frequency [10,11]. Implementations of these two similarity functions have been used during our experiments.

**Objective Function.** Within our experiments we used the weighted sum of all similarities as our objective function. With the definition of weights it is

possible to support *user preference queries*, e.g. by specifying how important each restriction is.

**Implementation.** We implemented all algorithms described throughout this paper in a 3-tier architecture: (1) the DB-layer containing our FTA approach with the described similarity and objective function as well as BPA-2. This layer is completely embedded in the database and thus implemented in PL-SQL as stored procedures. (2) A database access layer which is used for sending queries to a database and receiving results. (3) A QBE-GUI which allows easy specification of top-k queries.

#### 4.1 Experimental Setup

To make a more accurate statement about the performance of FTA we carried out a detailed analysis and compared it with BPA-2. Within this paper we will primarily focus on *comparing the query answer time and the memory requirements* of both approaches.

**The Underlying Data Model.** The data model used during the experiments was a single relation containing nine attributes. Table 2 gives an overview on the chosen datatypes and the selectivity of each attribute. The selectivity is calculated as the inverse of the amount of distinct attribute values in the given relation.

**Table 2.** Attribute characteristics of our test relation

Name	Type	Selectivity
a1	numerical	0.5
a2	numerical	0.1
a3	categorical	0.04
a4	numerical	0.02
a5	categorical	0.01
a6	numerical	0.004
a7	categorical	0.002
a8	categorical	0.001
a9	numerical	0.0001

**Test Database.** The test database was filled with randomly generated data. The attributes of our test relation(s) are not correlated and their values are equally distributed. Since BPA-2 holds all objects in memory an index on database level does not affect query answering time for BPA-2. On the other hand FTA could additionally benefit from such an index. When indexing the search attributes on database level we found that FTA could improve its query answering time at an average of 1,5. Nevertheless to ensure a fair comparison between FTA and BPA-2 we decided to compare the query answering times without support of database indexing techniques.

**Test Scenarios.** For providing a meaningful analysis we adapted three parameters in our scenarios. (1) size of the relation: 10k, 20k, 50k, 100k, 200k and 500k entries, (2) amount of tuples to be returned: 10, 20, 50, 100, 250, 500, 1000 and (3) number of restricted attributes: 1 up to 9. The queries (Q1, Q2, ... Q9) posted in the test runs were constructed in the following way: Query Q1 restricts attribute a1, query Q2 additionally restricts attribute a2. Finally query Q9 restricts all available attributes and thus has the lowest selectivity of all posted queries. All these parameters were used in each possible combination for FTA and for BPA-2. This gives 378 queries to be executed for both algorithms and a profound basis to compare the two approaches.

## 4.2 Discussion of Results

Within this section we present six diagrams generated from the data produced by our test runs. For comparing the two approaches we also defined ratios for the query answer time (qat) and for the memory requirements:

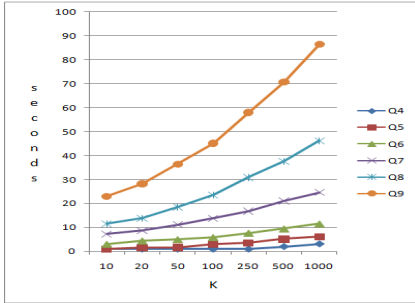
1.  $ratio_{qat} = qat_{BPA-2} / qat_{FTA}$ .
2.  $ratio_{mem} = bufferedObjects_{BPA-2} / bufferedObjects_{FTA}$

For the calculation of buffered objects we counted all objects fetched from the database and stored in internal buffers for handling by the algorithms. For each of the buffered objects both approaches calculated the objective function. We did not take into consideration that BPA-2 maintains one list for each restriction where *all objects* of the relation are stored. That would make the comparison even worse for BPA-2. Thus we only counted all objects examined while stepping down the sorted lists of BPA-2.

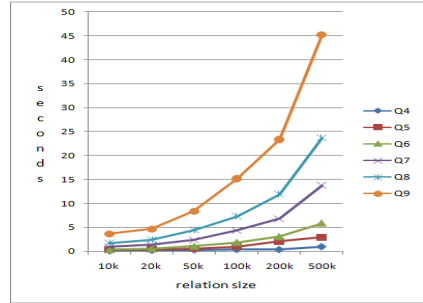
Figure 1.3.5 show the query answer time,  $ratio_{qat}$  and  $ratio_{mem}$  for a relation with 500k entries and varying K. In figure 1 we observe that FTA grows in a seemingly exponential way when the number of searched objects increases. However for large databases the query answer time is still reasonable and for FTA always by a factor of 10 to 35 below BPA-2. This information is illustrated in figure 3 where the ratio for the query answer time between FTA and BPA-2 is shown. We can also see that  $ratio_{qat}$  is almost constant even when the number of searched objects increases. The bursts where the  $ratio_{qat}$  is increasing rapidly stem from situations where FTA quickly finds enough objects in one of the best indexes. Since BPA-2 always accesses one index after the other many minor important objects are fetched and processed. This fetching and processing is costly and results in a much higher query answer time. The further we can see that for a reasonable amount of searched objects  $ratio_{mem}$  (figure 5) is mostly between two and six. Comparing these curves we can see that they are quite similar to the  $ratio_{qat}$  in figure 3.

In figures 2.4.6 we show the query answer time,  $ratio_{qat}$  and  $ratio_{mem}$  for searching the top-100 objects in relations with varying size. The query answer time (figure 2) and also the  $ratio_{qat}$  (figure 4) grow in a seemingly exponential way. We can see that FTA gets much better than BPA-2 when the size of the relation grows. The further the  $ratio_{mem}$  also grows steadily when the size of

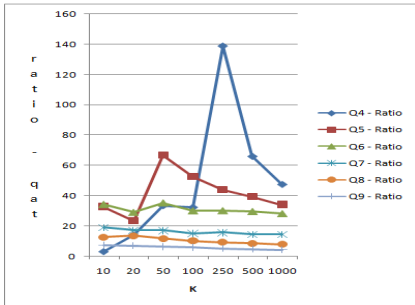




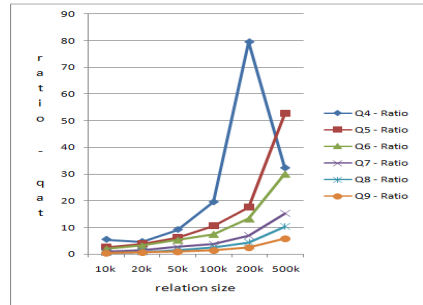
**Fig. 1.** Query answer times of *FTA* on a relation with 500k objects and varying search amount *K*



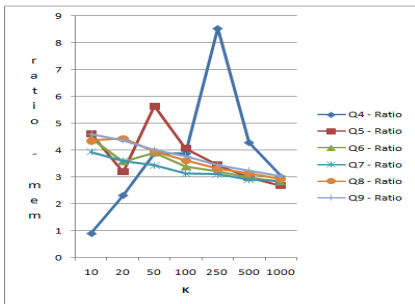
**Fig. 2.** Query answer times of *FTA* for searching the top-100 objects in relations with different size



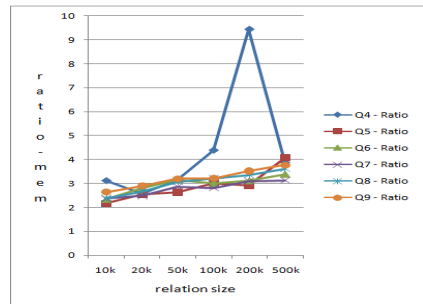
**Fig. 3.** Ratio of query answer times on a relation with 500k objects and varying search amount *K*



**Fig. 4.** Ratio of query answer times searching the top-100 objects in relations of different size



**Fig. 5.** Ratio for amount of examined objects on a relation with 500k objects and varying search amount *K*



**Fig. 6.** Ratio for amount of examined objects searching the top-100 objects in relations of different size

the relations gets larger. We can conclude that FTA compared to BPA-2 gets better the larger the amount of objects in the search relation is.

Additionally, we found that (1) when posting two queries with equal selectivity the query with fewer restricted attributes can be answered faster, and (2) when posting two queries with an equal amount of restricted attributes the query with the higher selectivity can be answered faster. In summary, FTA was faster for *most test cases* by one to two orders of magnitude.

## 5 Conclusion

The goal of all these efforts is to reduce the effort for users to retrieve relevant information from large databases where they typically fall into the recall/precision trap. We have shown that our algorithm for computing the top-k tuples outperforms BPA-2 of Akbarinia et al. [2]. To the best of our knowledge BPA-2 is the fastest top-k query answering approach based on the well known TA of Fagin et al. In our experiments FTA returned the k-objects 10 to 35 times faster than BPA-2. This performance speedup is achieved with the help of a different value based indexing structure and a novel look-ahead technique. The gain for query answering is the greater the smaller the fraction of returned tuples vs. the database size. In typical application areas for such queries, this ratio is typically large. Additionally, our algorithm proved to have much smaller memory requirements compared to BPA-2.

## References

1. Agrawal, S., Chaudhuri, S.: Automated ranking of database query results. In: CIDR, pp. 888–899 (2003)
2. Akbarinia, R., Pacitti, E., Valduriez, P.: Best position algorithms for top-k queries. In: Proc. of the 33rd Intl. Conf. on VLDBs, VLDB 2007 (2007)
3. Dabringer, C., Eder, J.: Efficient top-k retrieval for user preference queries. In: Proc. of the 26th ACM Symposium on Applied Computing (2011)
4. Eder, J., Dabringer, C., Schicho, M., Stark, K.: Information systems for federated biobanks. Trans. on Large Scale Data and Knowledge Centered Systems (2009)
5. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: Proc. of the 2001 ACM Symp. on Principles of Database Systems. ACM, New York (2001)
6. Guntzer, U., Balke, W.-T., Kiessling, W.: Optimizing multi-feature queries for image databases. In: Proc. of the 26th Int. Conf. on VLDBs, pp. 419–428. Morgan Kaufmann, San Francisco (2000)
7. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing techniques in relational database systems. ACM Comput. Surv. 40(4), 1–58 (2008)
8. Lesot, M., Rifqi, M., Benhadda, H.: Similarity measures for binary and numerical data. Int. J. Knowl. Eng. Soft Data Paradigm. 1, 63–84 (2009)
9. Marian, A., Bruno, N., Gravano, L.: Evaluating top-k queries over web-accessible databases. ACM Trans. Database Syst. 29(2), 319–362 (2004)
10. Robertson, S.: Understanding inverse document frequency: on theoretical arguments for idf. Journal of Documentation 60, 503–520 (2004)

# Towards an On-Line Analysis of Tweets Processing

Sandra Bringay<sup>1,2</sup>, Nicolas Béchet<sup>3</sup>, Flavien Bouillot<sup>1</sup>,  
Pascal Poncelet<sup>1</sup>, Mathieu Roche<sup>1</sup>, and Maguelonne Teisseire<sup>1,4</sup>

<sup>1</sup> LIRMM – CNRS, Univ. Montpellier 2, France  
{bringay,bouillot,poncelet,mroche}@lirmm.fr

<sup>2</sup> Dept MIAp, Univ. Montpellier 3, France

<sup>3</sup> INRIA Rocquencourt - Domaine de Voluceau, France  
nicolas.bechet@inria.fr

<sup>4</sup> CEMAGREF – UMR TETIS, France  
maguelonne.teisseire@cemagref.fr

**Abstract.** Tweets exchanged over the Internet represent an important source of information, even if their characteristics make them difficult to analyze (a maximum of 140 characters, etc.). In this paper, we define a data warehouse model to analyze large volumes of tweets by proposing measures relevant in the context of knowledge discovery. The use of data warehouses as a tool for the storage and analysis of textual documents is not new but current measures are not well-suited to the specificities of the manipulated data. We also propose a new way for extracting the context of a concept in a hierarchy. Experiments carried out on real data underline the relevance of our proposal.

## 1 Introduction

In recent years, the development of social and collaborative Web 2.0 has given users a more active role in collaborative networks. Blogs to share one's diary, RSS news to track the latest information on a specific topic, and tweets to publish one's actions, are now extremely widespread. Easy to create and manage, these tools are used by Internet users, businesses or other organizations to distribute information about themselves. This data creates unexpected applications in terms of decision-making. Indeed, decision makers can use these large volumes of information as new resources to automatically extract useful information.

Since its introduction in 2006, the Twitter website<sup>1</sup> has developed to such an extent that it is currently ranked as the 10<sup>th</sup> most visited site in the world<sup>2</sup>. Twitter is a platform of microblogging. This means that it is a system for sharing information where users can either follow other users who post short messages or be followed themselves. In January 2010, the number of exchanged tweets reached 1.2 billion and more than 40 million tweets are exchanged per day<sup>3</sup>.

<sup>1</sup> <http://twitter.com>

<sup>2</sup> <http://www.alexa.com/siteinfo/twitter.com>

<sup>3</sup> <http://blog.twitter.com/2010/02/measuring-tweets.html>

When a user follows a person, the user receives all messages from this person, and in turn, when that user tweets, all his followers will receive the messages. Tweets are associated with meta-information that cannot be included in messages (e.g., date, location, etc.) or included in the message in the form of tags having a special meaning. Tweets can be represented in a multidimensional way by taking into account all this meta-information as well as associated temporal relations. In this paper, we focus on the datawarehouse [1] as a tool for the storage and analysis of multidimensional and historized data. It thus becomes possible to manipulate a set of indicators (measures) according to different dimensions which may be provided with one or more hierarchies. Associated operators (e.g., Roll-up, Drill-down, etc.) allow an intuitive navigation on different levels of the hierarchy.

This paper deals with different operators to identify trends, the top-k most significant words over a period of time, the most representative of a city or country, for a certain month, in a year, etc. as well as the impact of hierarchies on these operators. We propose an adapted measure, called  $TF-IDF_{adaptive}$ , which identifies the most significant words according to level hierarchies of the cube (e.g., on the location dimension). The use of hierarchies to manage words in the tweets enables us to offer a contextualization in order to better understand the content. We illustrate our proposal by using the MeSH<sup>4</sup> (Medical Subject Headings) which is used for indexing PubMed articles<sup>5</sup>.

The rest of the paper is organized as follows. Section 2 describes a data model for cubes of tweets and details the proposed measure. In Section 3, we consider that a hierarchy on the words in tweets is available and propose a new approach to contextualize the words in this hierarchy. We present some results of conducted experiments in Section 4. Before concluding by presenting future work, we propose a state-of-the-art in Section 5.

## 2 What Is the Most Appropriate Measure for Tweets?

### 2.1 Preliminary Definitions

In this section we introduce the model adapted to a cube of tweets. According to [2], a fact table  $F$  is defined on the schema  $D = \{T_1, \dots, T_n, M\}$  where  $T_i$  ( $i = 1, \dots, n$ ) are the dimensions and  $M$  stands for a measure. Each dimension  $T_i$  is defined over a domain  $D$  partitioned into a set of categories  $C_j$ . We thus have  $D = \cup_j C_j$ .  $D$  is also provided with a partial order  $\sqsubseteq_D$  to compare the values of the domain  $D$ . Each category represents the values associated with a level of granularity. We note  $e \in D$  to specify that  $e$  is a value of the dimension  $D$  if there is a category  $C_j \subseteq D$  such that  $e \in \cup_j C_j$ . Note that two special categories are distinguished and are present on all dimensions:  $\perp_D \in C_D$  corresponding respectively to the level of finer and higher granularity. In our approach, the partial order defined on the domains of the dimensions stands for the inclusion of keywords associated to the values of the dimensions. Thus, let  $e_1, e_2 \in \cup_j C_j$  be two values, we have  $e_1 \sqsubseteq_D e_2$  if  $e_1$  is logically contained in  $e_2$ .

<sup>4</sup> <http://www.ncbi.nlm.nih.gov/mesh>

<sup>5</sup> <http://www.ncbi.nlm.nih.gov/PubMed/>

## 2.2 The Data Model

We instantiate the data model of the previous section to take into account the different dimensions of description and propose a specific dimension to the words associated to tweets.

Let us consider, for example, the analysis of tweets dedicated to the Duncan diet (e.g., "The Dukan diet is the best diet ever, *FACT!!! Its just meat for me for the next 5 day YEESSSS*"). We wish, for example, to follow the comments or opinions of different people on the effectiveness of a diet. In order to extract the tweets, we query Twitter using a set of seed words: *Duncan, diet, slim, protein, etc.* In this case, the original values of the word dimension are  $dom(word) = \{Duncan, diet, slim, protein, \dots\}$ .

Figure 1 illustrates the data model. We find the dimension ( $location \perp_{location} = City \leq State \leq Country \leq \top_{location}$ ), and the dimension  $time (\perp_{time} = day \leq month \leq semester \leq year \leq \top_{time})$ .

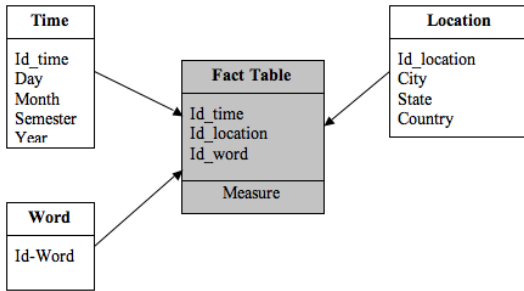


Fig. 1. The schema related to a diet application

The domain of the *word* dimension is that of the seed words with the words appearing frequently with them. In the fact table, several measures may be used. Traditionally it is the TF-IDF. This issue is addressed in the next section.

## 2.3 Towards an Appropriate Measure

Relying only on knowledge of the hierarchy in a cube does not always allow a good aggregation (i.e., corresponding to a real situation). For instance, the characteristics of the words in tweets are not necessarily the same in a State and in a City. The aggregation measure that we propose is based on approaches from Information Retrieval (IR).

In our process, the first step is to merge the number of occurrences of words specific to a level. More precisely, we list all the words located in tweets that match a given level (e.g., City, State, Country). If the user wishes to focus the search on a specific City, the words from the tweets coming from this city form a vector. We can apply this same principle to the State by using a Roll-up operator. The aim of our work is to highlight the discriminant words for each level.

Traditionally,  $TF-IDF$  measure gives greater weight to the discriminant words of a document [3]. Like [4], we propose a measure called  $TF-IDF_{adaptive}$  aiming to rank the words according to the level where the user is located and defined as follows:

$$TF_{i,j} - IDF_i^k = \frac{n_{i,j}}{\sum_k n_{k,j}} \times \log_2 \frac{|E^k|}{|\{e_j^k : t_i \in e_j^k\}|} \quad (1)$$

where  $|E^k|$  stands for the total number of elements of type  $k$  (in our example,  $k = \{City, State, Country\}$ ) which corresponds to the level of the hierarchy that the decision maker wants to aggregate.  $|\{e_j^k : t_i \in e_j^k\}|$  is relative to the number of elements of type  $k$  where the term  $t_i$  appears.

### 3 A Hierarchy of Words for Tweets

In this section, we adopt a hierarchy on the words to allow the contextualization of words in tweets.

#### 3.1 The Data and the Model

For the hierarchy, we use the MeSH (Medical Subject Headings)<sup>6</sup> National Library of Medicine's controlled vocabulary thesaurus. It consists of sets of terms naming descriptors in a twelve-level hierarchy that permits the search to be carried out at various levels of specificity. At the most general level of the hierarchical structure are very broad headings such as "Anatomy" or "Mental Disorders". More specific headings are found at more narrow levels, such as "Ankle" and "Conduct Disorder". In 2011, 26,142 descriptors are available in MeSH. There are also over 177,000 entry terms that assist in finding the most appropriate MeSH Heading, for example, "Vitamin C" is an entry term to "Ascorbic Acid".

The data model is updated to take into account this new dimension. Compared to the previous model (See Figure 1) the dimension "Word" has been replaced by MeSHWord. MeSHWord has a partial order,  $\sqsubseteq_{MeSHWord}$ , to compare the different values of the domain. One of the main problems with the use of this thesaurus is that different terms may occur at various levels in the hierarchy. This ambiguity raises the problem of using operators like Roll-up or Drill-down to navigate in the cube. In order to illustrate this problem let us consider the following example.

**Example 1.** *Let us consider the following tweet: "pneumonia & serious nerve problems. can't stand up. possible myasthenia gravis treatable with meds.". If we look in MeSH for the descriptor pneumonia, we find this term occurring in several places (See Figure 2). Depending on the position in the hierarchy, a Roll-up operation on pneumonia will not give the same result (i.e., "respiratory tract diseases" versus. "lung diseases").*

<sup>6</sup> <http://www.nlm.nih.gov/pubs/factsheets/mesh.html>

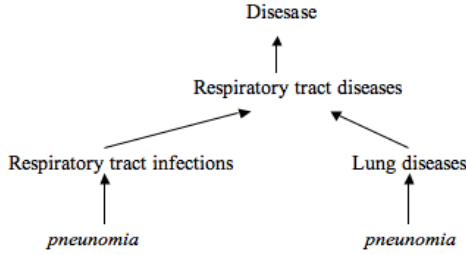


Fig. 2. An example of the MeSH thesaurus

### 3.2 How to Identify the Context of a Tweet?

We have shown in Example 1, the difficulty of locating the context of a term in the hierarchy. However, a closer look at the tweet shows that the words themselves can be helpful to determine the context. In order to disambiguate polysemous words in the hierarchy of MeSH, we adapt the  $AcroDef_{MI^3}$  method described in [5] where the authors show the efficiency of this method in a biomedical context. This measure is based on the Cubic Mutual Information [6] that enhances the impact of frequent co-occurrences of two words in a given context. For a context  $C$ ,  $AcroDef_{MI^3}$  is defined as follows:

$$AcroDef_{MI^3}^C(m1, m2) = \frac{(nb(m1 \text{ and } m2 \text{ and } C))^3}{nb(m1 \text{ and } C) \times nb(m2 \text{ and } C)} \quad (2)$$

In our case, we want to calculate the dependence between a word  $m$  to disambiguate and different words  $m_t$  of the tweets using the context of the hierarchy (i.e., parents  $p$  of the word  $m$ ).

**Example 2.** Let us consider the word 'pneumonia' to disambiguate in Example 1. Here we calculate the dependence between this word  $m$  and the other words following 'pneumonia' (nouns, verbs, and adjectives are selected with a Part-of-Speech process): 'serious' and 'nerve'. This dependence is calculated regarding the context of both possible fathers in the MeSH hierarchy. In order to predict where in the MeSH thesaurus we have to associate the word 'pneumonia', we perform the following operations:

- $nb(pneumonia, m_t, "lung \text{ diseases} ") = 227$  (number of returned pages with the queries 'pneumonia serious "lung diseases"' and 'pneumonia nerve "lung diseases"')
- $nb(pneumonia, m_t, "respiratory \text{ tract infections} ") = 496$

The dependence of the terms is given by:

- $AcroDef_{MI^3}^{lung \text{ diseases} } (pneumonia, m_t) = 0.02$
- $AcroDef_{MI^3}^{respiratory \text{ tract infections} } (pneumonia, m_t) = 0.11$

Thus, in the tweet from Example 1, for the word *pneumonia*, we will preferably do the aggregation at the level of the concept 'respiratory tract infections' of the MeSH.

Note that this step of disambiguation, which is essential for data from MeSH, is quite costly in terms of the number of queries. It therefore seems more appropriate to call these functions during the ETL process rather than carrying out such processing when browsing the cube.

## 4 Experiments

In order to evaluate our approach, several experiments were conducted. These were performed using PostgreSQL 8.4 with the Pentaho Mondrian 3.20 environment. To extract the tweets related to the vocabulary used in MeSH, we focus on the tweets related to "Disease" and queries Twitter by using all the terms of the corresponding hierarchy. We collected 1,801,310 tweets in English from January 2011 to February 2011. In these experiments, we analyze the first words returned by the TF-IDF<sub>adaptive</sub> (highest scores). For example, the following table presents the first 12 words of tweets in the United States, for the State of Illinois and the City of Chicago during the month of January 2011.

United Sates	Illinois	Chicago
wart	risk	risk
pneumonia	vaccination	wart
vaccination	wart	pneumonia
risk	pneumonia	wood
lymphoma	wood	colonoscopy
common cold	colonoscopy	x-ray
disease	x-ray	death
meningitis	encephalitis	school
infection	death	vaccination
vaccine	school	eye infection
life	eye infection	patient
hepatitis	man	russia

Now we consider an example of the application of our approach. Figures 3 and 4 visualize the worldwide coverage of the words *hepatitis* and *pneumonia* excluding the United States, the United Kingdom, and Canada. This coverage is obtained by fixing the location dimension and by examining the frequency of the Word over the period.



**Fig. 3.** Distribution of the use of the word hepatitis



**Fig. 4.** Distribution of the use of the word pneumonia

Finally we evaluated the prediction measure (i.e.,  $AcroDef_{MI^3}$ ) within the MeSH hierarchy (see section 3.2). We extracted more than 5,000 Facebook messages (the same kind of messages as tweets) from the *food* topic. These messages



contain at least one polysemous term (i.e. a term which can be associated to the hierarchy *food and beverages*) and one or two other hierarchies of MeSH: *Eukaryota*, *lipids*, *plant structures*, and so forth. A correct prediction means that  $AcroDef_{MJ3}$  associates this polysemous term with the *food and beverages* concept. In the following table, three types of elements are used in order to characterize the hierarchy (context of the  $AcroDef_{MJ3}$  measure): Father (F), grand-father (GF), and father + grand-father (FGF). This table shows that (1) the use of more generic information (grand-father) is more relevant, (2) the association of all the available information improves the quality of the prediction. In our future work we plan to add other hierarchical information (e.g. son, cousins).

Elements of the hierarchy used	F	GF	FGF
Prediction	60.8%	63.6%	68.0%

## 5 Related Work

The analysis of textual data from tweets has recently been addressed in many research studies and many proposals exist. For example, in [7], the authors propose analyzing the content of the tweets in real time to detect alarms during an earthquake. The authors of TwitterMonitor [8] present a system to automatically extract trends in the stream of tweets. A quite similar approach is proposed in [9]. However, to the best of our knowledge, most existing studies mainly focus on a specific analysis of tweets and do not provide general tools for the decision maker (i.e., for manipulating the information embedded in tweets according to their needs). Thus, few studies have been interested in the use of cubes to manage tweets. Recent work has focused on integrating textual data in data warehouses. In this context, aggregation methods suitable for textual data have been proposed. For example, in [10], the authors propose using Natural Language Processing techniques to aggregate the words with the same root or the same lemmas. They also use existing semantic tools such as Wordnet or Roget to group words together. Apart from using morpho-syntactic and semantic knowledge, other studies consider numerical approaches from the field of Information Retrieval (IR) to aggregate textual data. Thus, the authors of [11] propose aggregating documents according to keywords by using a semantic hierarchy of words found in the datawarehouses and some measures from IR. Such methods from IR are also used in the work of [2] which takes into account a "context" and "relevance" dimension to build a datawarehouse of textual data called R-cube. Other approaches add a new specific dimension. For example, in [12], the authors add a "topic" dimension and apply the PLSA approach [13] to extract the themes representing the documents in this new dimension. Finally, in [14] the authors propose aggregating parts of documents to provide the decision maker with words specific to the aggregation. In this context, the authors use a first function to select the most significant words using the classical  $TF-IDF$  measure.

## 6 Conclusion

In this paper we proposed a new approach to analyze tweets from their multidimensional characteristics. The originality of our proposal is to define and manipulate cubes of tweets. We have shown through two different models and applications: no predefined hierarchy on tweets (i.e., diet analysis) and existing hierarchy (i.e., using the MeSH thesaurus), that the analysis of tweets requires the definition of new measures and that a contextualization step is relevant.

Future work involves several issues. First we want to extend the proposed approach to take into account opinions or feelings expressed in the tweets. Recent studies analyze the mood of people (e.g., <http://twittermood.org/>). We want to enhance these approaches by analyzing the content of tweets and thus be able to automatically extract knowledge such as: who are the people who followed a diet and are dissatisfied? Secondly, we wish to consider tweets as available in the form of a stream and propose new techniques for efficiently storing the data.

## References

1. Codd, E., Codd, S., Salley, C.: Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate. In: White Paper (1993)
2. Pérez-Martínez, J.M., Llavori, R.B., Cabo, M.J.A., Pedersen, T.B.: Contextualizing data warehouses with documents. *Decision Support Systems* 45(1), 77–94 (2008)
3. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. *Commun. ACM* 18(11), 613–620 (1975)
4. Grabs, T., Schek, H.J.: ETH Zurich at INEX: Flexible Information Retrieval from XML with PowerDB-XML. In: Grabs, T., Schek, H.J. (eds.) *XML with PowerDB-XML. INEX Workshop*, pp. 141–148. ERCIM Publications (2002)
5. Roche, M., Prince, V.: Managing the acronym/expansion identification process for text-mining applications. *Int. J. of Software and Informatics* 2(2), 163–179 (2008)
6. Daille, B.: Approche mixte pour l'extraction automatique de terminologie: statistiques lexicales et filtres linguistiques. PhD thesis, Université Paris 7 (1994)
7. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In: *Proceedings of WWW*, pp. 851–860 (2010)
8. Mathioudakis, M., Koudas, N.: Twittermonitor: trend detection over the twitter stream. In: *Proceedings of SIGMOD, Demonstration*, pp. 1155–1158 (2010)
9. Benhardus, J.: Streaming trend detection in twitter. In: *National Science Foundation REU for Artificial Intelligence, NLP and IR* (2010)
10. Keith, S., Kaser, O., Lemire, D.: Analyzing large collections of electronic text using olap. Technical Report TR-05-001, UNBSJ CSAS (2005)
11. Lin, C.X., Ding, B., Han, J., Zhu, F., Zhao, B.: Text Cube: Computing IR Measures for Multidimensional Text Database Analysis. In: *Proc. of ICDM*, pp. 905–910 (2008)
12. Zhang, D., Zhai, C., Han, J.: Topic cube: Topic modeling for olap on multidimensional text databases. In: *Proc. of SIAM*, pp. 1123–1134 (2009)
13. Hofmann, T.: Probabilistic latent semantic analysis. In: *Proc. of Uncertainty in Artificial Intelligence, UAI 1999*, pp. 289–296 (1999)
14. Pujolle, G., Ravat, F., Teste, O., Tournier, R.: Fonctions d'agrégation pour l'analyse en ligne (OLAP) de données textuelles. Fonctions TOP\_KW et AVG\_KW opérant sur des termes 13(6), 61–84 (2008)

# The Fix-Point Method for Discrete Events Simulation Using SQL and UDF

Qiming Chen, Meichun Hsu, and Bin Zhang

HP Labs

Palo Alto, California, USA

Hewlett Packard Co.

{qiming.chen,meichun.hsu,bin.zhang2}@hp.com

**Abstract.** In this work we focus on leveraging SQL's expressive power and query engine's data processing capability for large scale discrete event simulation.

The challenge of using SQL query for discrete event simulation is that the simulation involves concurrent operations where the past events, the future events and the pending events (not fully instantiated yet) must be taken into account; but a SQL query is a set-wise operation that only cares of the current state. Specifically when the pending events are recursively generated as a result of processing primary events, there is no way to "insert" them back to the chronological sequence of input data, or to introduce a "clock" to the query processing. Due to these difficulties, there is no effort on using SQL query for discrete event simulation has been reported yet.

We propose a mechanism for discrete event simulation, where a chronological sequence of input records, which bear pending events, are processed by a SQL query recursively until reaching the fixpoint with all the pending events instantiated. This approach is characterized by leveraging the pipelined query processing to serialize concurrent events as well as to scope event chaining, and in this way to generate concurrent events incrementally and efficiently. Our experience reveals its flexibility and scalability.

## 1 Introduction

*Discrete event simulation* helps users to diagnose the operations, identify the bottlenecks, and understand the performance KPIs (Key Performance Indicators) of a system such as worker utilization, on-time delivery rate, customer waiting time, etc[1,5,6]. Complex event simulation is the counterpart of Complex Event Processing (CEP) [3]; they share the principles in concurrent event serialization, propagation, etc.

In discrete-event simulation, the operation of a system is represented as a chronological sequence of events. An event occurs at an instant time and marks a change of state in the system. As a result of processing a primary event, other pending events (not fully instantiated) may be subsequently created; the event propagation may be chained since a pending event may further result in other pending events. When multiple queues exist, concurrent events may present to each of them, even if the primary events are ordered by time. In the conventional event simulation, for

time- sequencing the instantiated and pending events, typically a system clock is used for pointing to the next event.

The major challenge of using SQL query for discrete event simulation lies in that discrete event simulation involves concurrent operations where both past events and future events must be taken into account; however, a SQL query is a set-wise operation that only cares of the current state. In query based event processing, the input tuples, that bear the primary events to be instantiated, are processed sequentially, but the subsequently derived pending events are hidden from the input sequence, and there is no way to “insert” them back to the input data; it is also difficult to associate a “clock” to the query processing. Due to these difficulties, to the best of our knowledge, so far no effort on using SQL query for discrete event simulation has been reported.

Motivated by integrating events simulation with data warehousing for scalability and data-access efficiency [2-4], we research into the issues of using SQL query with User Defined Functions (UDFs) to generate discrete event simulation data, where a chronological sequence of input records, which bear pending events, are processed by a SQL query recursively until reaching the fixpoint with all the pending events instantiated. This approach is characterized by leveraging the pipelined query processing to serialize concurrent events as well as to scope event chaining, and in this way to generate concurrent events incrementally and efficiently. Our experience reveals its flexibility and scalability.

## 2 Discrete Event Simulation Using SQL

For easy understanding let us start with an example where customers access services; we consider a *service\_access* event as a composite event with the time for a customer to arrive the queue of a service and the time to leave the service. A *service\_access* event with the time of entering the queue instantiated but the time of leaving the service not instantiated yet, is called a *pending event*. A pending event is *instantiated* by filling the time departing from the service; the instantiation of a pending event may result in another pending event (e.g. a customer arrives the queue of another service). Hereafter by events we refer to such *service\_access* events.

The query for generating events is applied to a sequence of Event Bearing Records (EBR) which bear the events, either instantiated or not. In the above example, an EBR is the *trajectory* of a customer of accessing several services, e.g. when the customer arrives queue A, leaves A, arrives queue B, ... etc. If N customers are concerned then there will be N EBRs (trajectories). The simulation is to fill in the time entries of the trajectory of each customer based on the preset or randomly generated plan (the planned order of accessing the services), as well as the concurrent events of other customers. Once all the trajectories are fully instantiated, various events and system statuses, such as the length of a given queue at a given time, the average waiting time, etc. can be derived from these trajectories. In this sense, event generation is made by EBR instantiation.

We consider three services A, B, C and accordingly three customer queues - one for a service. Each service is associated with an individual random number generator for assigning, on the fly, a random service-time in minutes within a given range.

There exist 15 possible routes for a customer to access one or more services, such as  $\langle A \rangle$ ,  $\langle A, C \rangle$ ,  $\langle C, A \rangle$ ,  $\langle B, C, A \rangle$ , ...etc, which are referred to as *routing plans* and identified by integers. We record the trajectory of a customer in an EBR with the following attributes.

[custID, eventTime, planID, nextService, arriveA, departA, arriveB, departB, arriveC, departC]

where

- a (composite) event for a customer to access a service is identified by custID, serviceID, timeArrive, timeDepart; if the value of timeDepart has not been determined yet, we call it a pending event;
- the time a customer arrives to the queue of the “nextService” is given as the value of “eventTime” and abbreviated by  $T_{npe}$ , which characterizes the next pending event. Note that  $T_{npe}$  is not a static, but an updatable timestamp.

Initially, a EBR is only filled with customer ID, routing plan ID, the first service ID and the corresponding eventTime,  $T_{npe}$ , i.e. the *customer-arrival* time to the first service queue, such as

[5, '9:20', 6, 'C', 0, 0, 0, 0, 0, 0]

The time entries of arriving and departing A, B and C, if applicable by the plan, are filled step by step during the simulation, representing the trajectory of a customer in accessing these services. When all the applicable entries are filled, we say that the EBR is fully instantiated, or completed. From the EBRs of all customers, the system state and event at any time can be derived, and therefore we turn our event generation task to the instantiation of these EBRs which log the events.

The EBRs containing pending events are buffered with a UDF and processed in the ascending order of  $T_{npe}$ . Before processing an input EBR (either updated or completed depending on its plan), with  $T_{npe}$ ,  $t$ , all the buffered EBRs with  $T_{npe}$ , earlier than  $t$  must be processed in the ascending order of  $T_{npe}$ ; if a further pending event  $e$  with  $T_{npe}$  within  $t$ , is derived as a result of processing a buffered EBR, the EBR will be (logically) moved back and inserted in the position consistent with the time order, and processed again later for instantiating event  $e$ . As the basic rule, a pending event can be instantiated only when all the related history (e.g. related to the same service) is known.

Based on this mechanism, the input EBR is actually processed last, *after* all the buffered EBRs with  $T_{npe}$ , earlier than  $t$  have been processed. This is because the history of events before  $t$  is fully known, after  $t$  is not fully known. We limit the cascade generation of pending events later than  $t$  to make the computation consistent (since the  $T_{npe}$  of the next query input is unknown) and incremental; this will be explained later.

A EBR is completed if all the applicable service specific arrival time and departure time are non-zero; in this case the  $T_{npe}$  is filled with 0 and the next service is filled with NULL. In the above example, EBR

[5, 0, 6, NULL, 0, 0, '9:30', '9:45', '9:20', '9:30']

is completed thus no further processing required; it indicates that customer 5 with plan 6 accesses service C from 9:20 to 9:30, then accesses service B from 9:30 to 9:45.

In summary, with our algorithm, a chronological sequence of EBRs,  $L$ , is processed one by one. During the processing of an input EBR, with  $T_{npe}$  as  $t$ , the EBRs previously buffered and subsequently updated with  $T_{npe}$  earlier than  $t$  are processed as well. The completed EBRs are put in a new chronological sequence,  $L'$ . By the end of processing  $L$ , all the incomplete EBRs remained in the cache will be appended to  $L'$  to be process in the next run, until all the EBRs become completed. Such incremental and recursive treatment is suitable for SQL query implementation. The events and the status of the simulated system, such as the average waiting time distribution over a time period, etc, can be easily derived from the EBRs using SQL.

An EBR is defined as a composite type (a tuple); thus the above ESF returns tuples to feed in the query. The initial customer trajectory table can be loaded by a query using ESF function scan, as

```
INSERT INTO trajectories SELECT * FROM init_event_gen (5, 420, 540);
```

In the following we discuss how to use SQL and UDF to generate simulation data. Note that although a query operation is set-oriented, the query evaluation is tuple-by-tuple pipelined; thus we do not rely on the query result, but *rely on the process of query evaluation, to guarantee the serialization in discrete event simulation.*

As the simulation query may generate subsequent *pending events* to be held in EBRs as a result of event processing. When a pending event is instantiated, a further pending event may be resulted, so on and so forth. As mentioned earlier, *a pending event cannot be processed without the full knowledge about the events before it for access the same queue.* How to infer such prior knowledge and act accordingly in a query with UDF, presents the major challenge.

### 3 Fix-Point Query Evaluation for Discrete Events Simulation

We developed a mechanism to generate discrete event simulation data by recursively evaluating a SQL query, i.e. recursively applying the same query to its last result, to the fixpoint with no new results generated. The input as well as the output of the query is a chronological sequence of EBRs. An execution of the query updates the input EBRs based on their plans and the system status, using UDFs. In each subsequent run, the completed EBRs are just pass-through (directly returned as it); the incomplete ones are processed steps further. Eventually, all the EBRs are fully instantiated from the initial EBRs,  $R_0$ , by applying a query  $Q$  recursively, i.e.

$$Q(\dots Q(Q(R_0))\dots)$$

to the fixpoint, i.e. no new results are derivable.

The key to this mechanism is relying on pipelined query processing to serialize event processing, and correctly streamlining event derivation and pending event processing.

**Event Generation along with EBR Evolvement.** In the example given above, a EBR records the trajectory of a customer with multiple *service\_access* (composite) events, each records when the customer arrives the service queue and departs from the service; if the value of *timeArrival* has, but *timeDepart* has not, been determined, it is

treated as a pending event. An EBR has at most one pending event at any time; its state bears the historical and next step trajectory of the customer.

A chronological sequence of EBR tuples are processed by the query with UDFs. The UDF is provided with a buffer to keep the EBRs containing pending events in the ascending order of  $T_{npe}$ . On each input EBR, the UDF acts in the following way.

- An input EBR with  $T_{npe}$ ,  $t$  is first put in the buffer, and together with other buffered EBRs with  $T_{npe}$  earlier than  $t$ , processed in the ascending time order.
- If a further pending event  $e$  having  $T_{npe}$  within  $t$ , is derived as a result of instantiating a buffered EBR, that EBR will be moved back and inserted in the position consistent with the time order, and processed again later for instantiate event  $e$ .
- Completed EBRs are returned from the *UDF* and removed from the buffer.

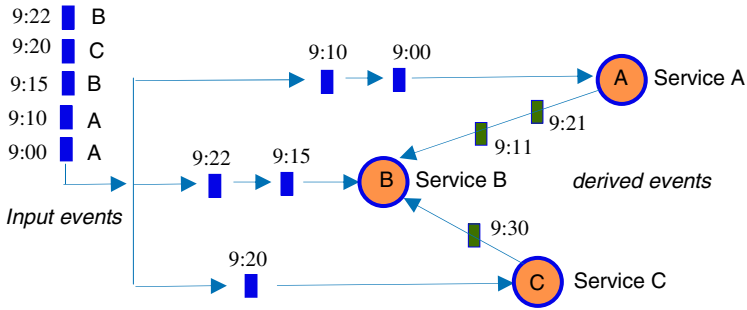
To make the order of processing pending events deterministic, we defined a priority order of services, say, A, B, C, and use the order of custIDs as the priority order of customers, such that in the events with the same timestamps, the one with higher service priority is considered the earlier event; if they have the same service priority, the one with higher customer priority is considered the earlier event. Note that no customer can access more than one service at a time.

The serialization concurrent event simulation is triggered and controlled by the query processing, which allows us to avoid the difficulty of handling clock in querying.

As the basic rule, a pending *service\_access* event in a EBR with timestamp ( $T_{npe}$ )  $t$  can be instantiated only when all the related history (e.g. related to the same service) is known, i.e., *after* all the related pending events earlier than  $t$  have been processed. For the same reason, a pending event with time stamp beyond  $t$  cannot be generated. We limit the scopes of pending event generation and processing based on this rule for computation consistency, and based on the pipelined query processing for incremental simulation.

**Cut Point of Event Processing and Derivation wrt Each Input Tuple.** Triggered by handling a query input tuple (EBR) with  $T_{npe}$  as  $t$ , the previous pending events with  $T_{npe}$  earlier than  $t$  will be instantiated based on the above rule. Although instantiating one pending event may result in another pending event. We have the pending events later than  $t$  left to the subsequent tuple processing, or event the next cycle query processing.

Refer to Fig. 1, the input EBRs are ordered by time. A UDF, *evolve()*, is introduced that is a table function with one EBR tuple (as a composite value) as its input and a set of EBRs as output where the result set may be empty for a particular invocation. This UDF may buffer an input EBR, instantiate its pending event, as well as process the previously buffered EBRs with pending events. It is also provided with the random number generators, one for each service, for assigning a random service time per *customer\_arrival* event. It may return the input EBR, the updated EBR or nothing in an invocation.



**Fig. 1.** The forming of concurrent events as a result of processing input events sequence

The query for generating customers’ trajectories may be written as

```
SELECT evolve(t.*, 10000) FROM (SELECT * FROM trajectories ORDER BY eventTime) t;
```

where 10000 is the maximal number of EBRs buffered.

The rules for naïve evaluation are simple; certain optional optimization will be discussed later.

- For each input EBR, say  $ebr_t$  if it is a completed EBR, or the buffer size already reaches the given limit, then it is returned; otherwise it is buffered. A primary event, say  $e_t$  with timestamp  $t$ , is read from  $ebr_t$  and processed together with the already buffered and cascade derived pending events with timestamps earlier than  $t$ . Any complete EBR after the above processing is removed from the buffer and sent to the output. After processing the last tuple, all the remaining EBRs are output from the table UDF, to be processed in the next run of the query.
- The fixpoint can be easily checked to see whether the query

```
SELECT * FROM trajectories WHERE eventTime > 0;
returns NULL.
```

Let us use  $X_t$  to represent a pending event to access service  $X$  with *customer\_arrival* time (i.e.  $T_{npe}$ )  $t$ . In the first run, the input events, the pending events as a result of processing the input events, the events processed as triggered by the input events, and the content of the buffer relating to service B are shown below.

Time	Input pending event	Events processed	Forward event	Buffered events for B
9:00	A <sub>9:00</sub>	A <sub>9:00</sub>	B <sub>9:11</sub>	B <sub>9:11</sub>
9:10	A <sub>9:10</sub>	A <sub>9:10</sub>	B <sub>9:21</sub>	B <sub>9:11</sub> , B <sub>9:21</sub>
9:15	B <sub>9:15</sub>	B <sub>9:11</sub> , B <sub>9:15</sub>		B <sub>9:21</sub>
9:20	C <sub>9:20</sub>	C <sub>9:20</sub>	B <sub>9:30</sub>	B <sub>9:21</sub> , B <sub>9:30</sub>
9:22	B <sub>9:22</sub>	B <sub>9:21</sub> , B <sub>9:22</sub>		B <sub>9:30</sub>



Then in the second run, all the completed EBRs are copied from query input to query output (or filtered), only the following event is processed.

Time	Input pending event	Events processed	Forward event	Buffered events for B
9:30	B <sub>9:30</sub>	B <sub>9:30</sub>		

After the second run, the recursion reaches the fixpoint, meaning that all the planned customer trajectories are generated.

Given the initial EBRs

[5, '9:00', 4, 'A', 0, 0, 0, 0, 0, 0]; [6, '9:10', 4, 'A', 0, 0, 0, 0, 0, 0]; [7, '9:15', 2, 'B', 0, 0, 0, 0, 0, 0]; [8, '9:20', 6, 'C', 0, 0, 0, 0, 0, 0]; [9, '9:22', 2, 'B', 0, 0, 0, 0, 0, 0]

After first iteration, they become

EBR	Updated Time
[5, 0, 4, NULL, '9:00', '9:11', '9:11', '9:16', 0, 0]	9:00, 9:15 (complete)
[6, 0, 4, NULL, '9:10', '9:21', '9:21', '9:26', 0, 0]	9:10, 9:22 (complete)
[7, 0, 2, NULL, 0, 0, '9:15', '9:20', 0, 0]	9:15 (complete)
[8, '9:30', 6, 'B', 0, 0, '9:30', 0, '9:20', '9:30']	9:20
[9, 0, 2, NULL, 0, 0, '9:22', '9:27', 0, 0]	9:22 (complete)

These EBRs become the input of the next run; after the second iteration

EBR	Updated Time
[5, 0, 4, NULL, '9:00', '9:11', '9:11', '9:16', 0, 0]	-
[6, 0, 4, NULL, '9:10', '9:21', '9:21', '9:26', 0, 0]	-
[7, 0, 2, NULL, 0, 0, '9:15', '9:20', 0, 0]	-
[8, 0, 6, NULL, 0, 0, '9:30', '9:35', '9:20', '9:30']	9:30
[9, 0, 2, NULL, 0, 0, '9:22', '9:27', 0, 0]	-

After this run, the simulation data generation is completed.

In short, the query runs a single query iteratively, and in each iteration, processes the whole temporal events (e.g. from 9am to 4pm), rather than by low-level code step by step following the clock. In each run, it is guaranteed that any incomplete EBR will be evolved at least one step.

The optional optimization is to extract the completed EBRs from the input of next iteration for reduced data load, which can be easily accomplished by filter the query results of the last run on condition “eventTime > 0” before the next run.

There exist various options to organize the SQL statements for generating the simulation data. Below is one option to write a query to be run iteratively: having the trajectory table extended with an additional integer attribute *cycle* to distinguish the EBRs generated in different executions of the query.

```

INSERT INTO trajectories
  SELECT cycle+1, evolve(t.*, 10000) FROM
  (SELECT * FROM trajectories ORDER BY eventTime) t
WHERE cycle = (SELECT MAX(cycle) FROM trajectories);

```

Using the above query, the complete trajectories may be generated in different cycles but can be easily retrieved under the time = 0 or nextService = NULL condition.

**Incremental Event Generation.** The proposed event generation is linearly scalable because it proceeds incrementally along a moving buffer of EBRs. A chronological sequence of input EBRs is processed one by one. Handled by a table UDF, the event generation is triggered by processing each input EBR,  $ebr_t$  with  $T_{npe}$  as  $t$ , together with those buffered EBRs with  $T_{npe}$  earlier than  $t$ .

## 4 Experiments and Conclusions

We have implemented the proposed approach on the extended PostgreSQL query engine, and demonstrated its merit by the discrete event generation for a proprietary enterprise application. We tested the scalability of using SQL query to generate discrete event simulation data, and learnt that it can scale linearly to very large size of input data. Most stand-alone simulation programs include a clock to keep track of the current simulation time, and require buffering all pending events in memory, which limits their scalability. The execution time of simulation varies wrt the number of service queues, as well as the throughput of the events, e.g. the number of customers and their arrival intervals, which is represented by the length of the UDF buffer. We set 20 hypothetical services and generated the input data in such a time intervals that keeps the UDF buffer size less than 10000, which means the more input data, the longer the time covered by the simulation. To focus, we omit further data derivation and analysis on the customer trajectories, and use the following basic query that is listed before:

```

SELECT evolve(t.*, 10000) FROM (SELECT * FROM trajectories ORDER BY eventTime) t

```

It runs 3 cycles to reach the fixpoint. In each subsequent run, the completed EBRs, if not filtered from input (not shown here), are just pass-through the UDF (directly returned as it); the incomplete ones are further processed. The experimental results are shown in Fig 2.

These experimental results are measured on HP xw8600 with 2 x Intel Xeon E54102 2.33 Ghz CPUs and 4 GB RAM, running Windows XP (x86\_32) and PostgreSQL 8.4.1; these results show that the simulation time is roughly linear to the input size, thus our approach can deal with very big data set, relying on the scalability of the database system. When the simulation task require the data already stored in the database, push data-intensive computation down to the data management layer allows fast data access and reduced data move. Further, our approach makes the event generation incremental thus limiting the memory consumption. All these contribute to high scalability and performance.

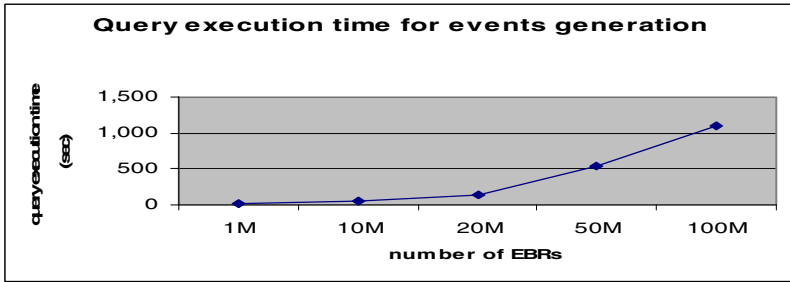


Fig. 2. Query time for discrete event generation with respect to input volume

With our approach, a chronological sequence of input records, which bear pending events, are processed by a SQL query recursively until reaching the fixpoint with all the pending events instantiated. This approach is characterized by leveraging the pipelined query processing to serialize concurrent events and to scope event chaining, and in this way to generate concurrent events incrementally and efficiently. Since the proposed discrete event simulation approach is integrated with data warehousing, it can be elaborated for knowledge discovery, information prediction and forecasting.

## References

1. Zeigler, B.P., Praehofer, H., Kim, T.G.: Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems, 2nd edn. Academic Press, London (2000)
2. Bryant, R.E.: Data-Intensive Supercomputing: The case for DISC. CMU-CS-07 128 (2007)
3. Chen, Q., Hsu, M., Zeller, H.: Experience in Continuous analytics as a Service (CaaS). In: Proc. EDBT (2011)
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 2007 (2008)
5. Robinson, S.: Simulation - The practice of model development and use. Wiley, Chichester (2004)
6. Tan, K.L., Thng, L.-J.: SNOOPY Calendar Queue. In: Proc. 32nd Winter Simulation Conference (2000)

# Approximate and Incremental Processing of Complex Queries against the Web of Data

Thanh Tran, Günter Ladwig, and Andreas Wagner

Institute AIFB, Karlsruhe Institute of Technology, Germany  
{ducthanh.tran, guenter.ladwig, a.wagner}@kit.edu

**Abstract.** The amount of data on the Web is increasing. Current exact and complete techniques for matching complex query pattern against graph-structured web data have limits. Considering web scale, exactness and completeness might have to be traded for responsiveness. We propose a new approach, allowing an affordable computation of an initial set of (possibly inexact) results, which can be incrementally refined as needed. It is based on approximate structure matching techniques, which leverage the notion of neighborhood overlap and structure index. For exact and complete result computation, evaluation results show that our incremental approach compares well with the state of the art. Moreover, approximative results can be computed in much lower response time, without compromising too much on precision.

## 1 Introduction

Recently, large amounts of semantic data has been made publicly available (e.g., data associated with Web pages as RDFa<sup>1</sup> or Linked Data<sup>2</sup>). The efficient management of semantic data at Web-scale bears novel challenges, which have attracted various research communities. Several RDF stores have been implemented, including *DB-based solutions* such as RDF-extensions for Oracle and DB2, Jena, Sesame, Virtuoso or *native solutions* for RDF like OWLIM, HStar, AllegroGraph, YARS [10], Hexastore [17] and RDF-3X [14]. Recently, also *IR technologies*, in particular the inverted index has been proposed for managing RDF data [18].

We observe that all these systems focus on computing complete and exact answers. Exact matching in a Web setting (with billions of RDF triples), however, results in unacceptable response times especially w.r.t. complex SPARQL<sup>3</sup> queries. The success of current Web search engines suggest that exact matching might be not needed. A more practical direction towards responsive and scalable solutions for Web-scale semantic data management is approximate matching equipped with sophisticated mechanisms for ranking. In this paper, we focus on the problem of *approximate matching* and how to refine matches *incrementally*.

<sup>1</sup> <http://w3.org/TR/xhtml1-rdfa-primer/>

<sup>2</sup> <http://www.w3.org/DesignIssues/LinkedData.html>

<sup>3</sup> <http://www.w3.org/TR/rdf-sparql-query/>

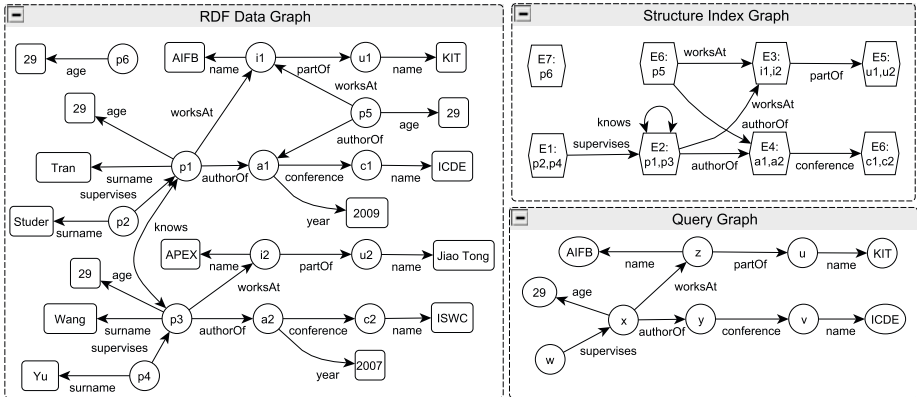


Fig. 1. a) A data graph, b) its structure index graph and c) a query graph

**Contribution.** We propose an approach for matching complex query patterns against the Web of Data. Our approach allows an “affordable” computation of an initial set of *approximate* results, which can be *incrementally* refined as needed. Our main contributions are:

- We propose a pipeline of operations for *graph pattern matching*, where results can be obtained in an *incremental* and *approximate* manner. We thereby allow a trade-off between precision and response time.
- Via four phases, results are reported early and can be incrementally refined as needed: First, entities matching the query are computed. Then, structural relationships between these entities are validated in the subsequent phases. To our knowledge, this is the first proposal towards a *pipelined processing* of complex queries for *incremental result computation*.
- For processing structural relationships, we introduce a novel *approximate structure matching technique* based on neighborhood overlap and show how it can be implemented efficiently using Bloom filters [3]. Another approximation is introduced for result refinement, which instead of using the large data graph, operates at summary level.
- Via a benchmark, we show that our incremental approach compares well w.r.t. time needed for computing exact and complete results. Further, it is promising w.r.t. approximate result computation.

**Outline.** In Section 2, we define the problem, describe the state-of-the-art and compare it to our approach. We discuss entity search in Section 3. In Section 4, we present our approximate structure matching technique, followed by the refinement phase in Section 5. We present an evaluation in Section 6. Finally, we conclude with Section 7.

## 2 Overview

In this section, we define the problem, discuss the state-of-the-art, highlight our main contributions and compare them with related work.

**Definition 1.** A data graph  $G$  is a tuple  $(V, L, E)$  where  $V$  is a set of nodes connected by labeled edges  $l(v_1, v_2) \in E \subseteq V \times V$  with  $v_1, v_2 \in V$  and  $l \in L$ . Further,  $V$  is the union  $V_E \uplus V_D$  with  $V_E$  representing entity nodes and  $V_D$  representing data nodes.  $E$  is the union  $E = E_R \uplus E_A$ , where  $E_R \subseteq V_E \times V_E$  represents relations between entity nodes and  $E_A \subseteq V_E \times V_D$  stands for entity attributes.

Note, our graph-structured data model is of interest in the Semantic Web and database community, as it captures RDF<sup>4</sup>, XML and relational data. Further, we consider *conjunctive queries*, a fragment of many query languages (e.g., SQL and SPARQL).

**Definition 2.** A conjunctive query  $q = (V_v \uplus V_c, P_r \uplus P_a)$  is an expression  $p_1 \wedge \dots \wedge p_n$ , where  $p_i \in P_r \uplus P_a$  are query atoms of the form  $p(n_1, n_2)$  with  $n_1 \in V_v, n_2 \in V_v \uplus V_c$  being variables  $V_v$  or constants  $V_c$  otherwise, and  $p_i$  are called predicates. We distinguish between relation query atoms  $p_r \in P_r$  and attribute query atoms  $p_a \in P_a$ , where  $p_r$  and  $p_a$  are drawn from labels of relation edges  $E_R$  and attribute edges  $E_A$  respectively. Relation query atoms paths  $(p_{r_1}, \dots, p_{r_k})$  contained in  $q$  have maximum length  $k^{max}$ .

Note, conjunctive queries can be conceived as graph patterns (corresponding to basic graph patterns in SPARQL). Fig. 1a, 1c depict a data and a query graph  $q(V_q = V_v \uplus V_c, P_q = P_r \uplus P_a)$ , with atoms as edges and variables (constants) as nodes. A match of a conjunctive query  $q$  on a graph  $G$  is a mapping  $\mu$  from

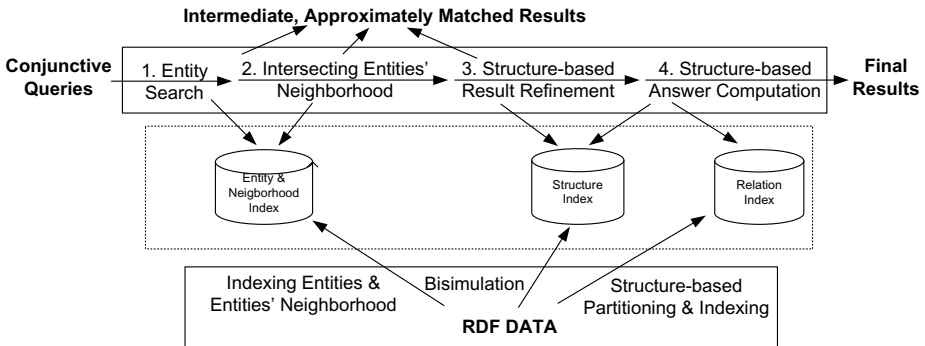


Fig. 2. Offline data preprocessing and online query answering

<sup>4</sup> <http://www.w3.org/TR/rdf-primer/>

variables and constants in  $q$ , to vertices in  $G$  such that the according substitution of variables in the graph representation of  $q$  would yield a subgraph of  $G$ . Query processing is a form of *graph pattern matching*, where the resulting subgraph of  $G$  exactly matches  $q$ . All such matching subgraphs are returned. As opposed to such an *exact* and *complete* query processing, an *approximate* procedure might output results, which only partially match the query graph (i.e., a result matches only some parts of  $q$ ). A query processing procedure is *incremental*, when results computed in the previous step are used for subsequent steps.

**Related Work.** Much work in RDF data management targets orthogonal problems, namely data partitioning [1] and indexing [10,17]. We now discuss related approaches that focus on the problem of query processing.

- *Query Processing.* Matching a query against a data graph is typically performed by retrieving triples and joining them along the query atoms. Join processing can be greatly accelerated, when the retrieved triples are already sorted. Sorting is the main advantage of vertical partitioning [1] and sextuple indexing [17] approaches, which feature data partitioning and indexing strategies that allow fast (nearly linear) merge joins. Further efficiency gains can be achieved by finding an optimal query plan [14].
- *Approximate Query Processing.* Above approaches deal with exact and complete query processing. In the Semantic Web community, notions for structural [11] and semantic approximation [7] have been proposed. So far, the focus is on finding and *ranking* answers that only approximately match a query. In database research, approximate techniques have been proposed for “taming the terabytes” [8,5,2]. Here, focus lies on *efficiency*. Instead of using the actual data, a query is processed over an appropriate synopsis (e.g., histograms, wavelets, or sample-based). Further, a suitable synopsis for XML data as been suggested [15], in order to compute approximate answers to twig-pattern queries. Unlike approaches for flat relational data [8], the synopsis used here takes both structural and value-based properties of the underlying data into account. Essentially, the synopsis is a structural summary of the data, which is augmented with statistical information (e.g., count or value distribution) at nodes and edges.
- *Incremental Query Processing.* Related to our incremental approach is work on top- $k$  query processing. Different algorithms for top- $k$  query processing have been proposed [12]. Here, the goal is to not compute all results, but to allow early termination by processing only the  $k$  best results.

**Overview of our Approach.** Fig. 2 illustrates the main concepts and techniques of our approach. The data graph is broken down into two parts. While attribute edges  $a \in E_A$  are stored in the entity index, relations  $r \in E_R$  are stored in the relation index. Also, a summary of the data (structure index [16]) is computed during data preprocessing. These indexes are employed in various operators in the pipeline, which we propose for query processing. We rely on sorted merge join and reuse related techniques [11,17]. However, as opposed to such exact and complete techniques, operations in our pipeline match the query

against the data in an approximate way to obtain possibly incorrect answers (which are refined during the process). Instead of operating on all intermediate answers, it is possible to apply a *cutoff* or let the user choose the candidates at every step.

Firstly, we decompose the query into entity queries and perform an *entity search* (ES), storing the results in sorted entity lists with a maximum length of *cutoff*. These results match attribute query atoms only. The next step is *approximate structure matching* (ASM): we verify if the current results also match the relation query atoms. By computing the overlap of the neighborhood of the entities obtained from the previous step, we verify if they are “somehow” connected, thereby matching the relation query atoms only in an approximate way. During structure-based result refinement (SRR), we further refine the matches by searching the structure index (a summary of the data) for paths, which “might” connect entities via relation query atoms. Only in the final step (*structure-based result computation* (SRC)), we actually use edges in the data graph to verify if these connections indeed exist, and output the resulting answers (exactly matching the query).

*Example 1.* During ES, we obtain 4 entity queries  $\{q_x, q_z, q_u, q_v\}$  from the initial query (Fig. 1c), and the corresponding results  $\{(p1, p3, p5, p6), i1, u1, c1\}$ , for a *cutoff*  $\leq 4$ . This and the results of the subsequent refinement steps are summarised in Table 1. During ASM, we find that all  $p1, p3, p5$  are somehow connected with the other entities, leading to 3 tuples. During SRR, we find out that  $p5$  is in the extension  $E6$ , and that this structure index node has no incoming *supervise* edge. Thus,  $p5$  cannot be part of an answer to  $q_x$ . During SRC, we observe that the previous approximate techniques lead to one incorrect result:  $p3$  could not be pruned through ASM, because  $p3$  *knows*  $p1$ , and is thus “indirectly” connected with the other entities  $i1, u1, c1$ .  $p3$  could also not be pruned through SRR, because when looking only at the summary (i.e., structure index),  $p3$  exhibits the same structure as  $p1$  (i.e., it is also in  $E2$ ) and thus, must be considered as a potential result. Clearly, using SRC we can not find out that  $p3$  is actually not connected with  $i1$  via *worksAt* (thus, could be ruled out).

**Design Rationales and Novelties.** Our design is based on the observation that state-of-the-art techniques perform well w.r.t queries containing highly selective atoms (e.g., attribute atoms with a constant). Query atoms containing variables (e.g., relation query atoms), on the other hand, are more expensive. Considering Web-scale, these query atoms become prohibitive. Processing *type(x, y)* or *friendOf(x, y)* for instance, requires millions of RDF triples to be retrieved. When dealing with complex graph patterns having many relation query atoms (that might involve a large number of triples), we propose a pipeline of operations, which starts with “cheap” query atoms to obtain an initial set of approximate answers, and incrementally continues with refining operations via more expensive query atoms.

Work on data partitioning and indexing [10, 17] are orthogonal, and complement our solution. Also, existing techniques for exact and complete query



processing based on sorted merge join are adopted [117]. Building upon these previous works, we present the first solution towards a *pipelined processing of complex queries* on Web data, enabling results to be computed approximately, incrementally, and reported early.

In particular, our approach is the first *approximate technique* for querying RDF data, which is capable of trading precision for time: approximately matched results can be reported early, and when needed, result precision can be improved through several subsequent refinement steps. Compared to existing techniques, the structure refinement step (SRR) resembles a technique for approximate twig pattern matching [15]. The difference is that our structure index is a synopsis for general graph-structured data, while the synopsis employed in [15], is for hierarchical XML data only. Different from any previous techniques, we introduce an additional level of approximation. This is realized by ASM, a novel approximate join operator that exploits the notion of neighborhood overlap for structure matching.

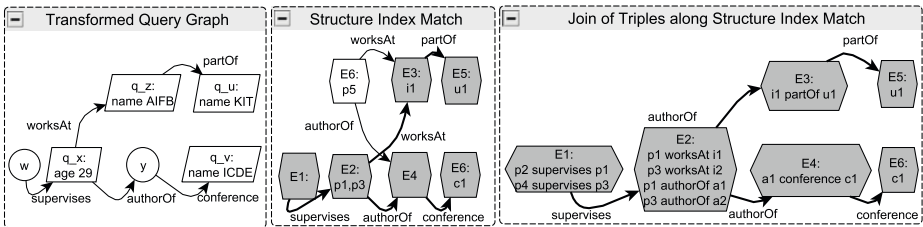
As opposed top- $k$  approaches, our *incremental approach* does not compute the best, but all approximate results, which are then iteratively refined in several steps. In particular, we do not focus on ranking aspects in this work and simply apply a predefined cutoff to prune large results.

**Table 1.** The different results for ES, ASM, SRR and SRC

ES				ASM			
$q_x$	$q_z$	$q_u$	$q_v$	$q_x$	$q_z$	$q_u$	$q_v$
p1	i1	u1	c1	p1	i1	u1	c1
p3	i1	u1	c1	p3	i1	u1	c1
p5	i1	u1	c1	p5	i1	u1	c1
p6	i1	u1	c1				

SRR				SRC			
$q_x$	$q_z$	$q_u$	$q_v$	$q_x$	$q_z$	$q_u$	$q_v$
p1	i1	u1	c1	p1	i1	u1	c1
p3	i1	u1	c1				



**Fig. 3.** a) The transformed query graph obtained in ES, b) the structure index match computed in SRR and c) SRC through joins along the structure index match

### 3 Entity Search

Let us first describe offline entity indexing and then online entity search.

**Entity Indexing.** Attributes  $a \in E_A$  that refer to a particular entity are grouped together and represented as a document (ENT-doc) in the inverted index. We use structured document indexing – a feature supported in standard IR engines such as Lucene – to store entities’ attributes values in different fields: we have (1) *extension id*: the entity’s extension id (used for SRR, c.f. Sec. 5);

(2) *denotations*: the entity’s URI and names; (3) *attributes*: concatenation of attribute/value; (4) *k-neighborhood*: neighbor entities reachable via paths with max. length  $k$ .

**Query Decomposition.** The goal is to parse the query graph  $q$  for computing intermediate results as follows:

- *Entity queries*  $Q_E$ . Each entity query  $q_{var} \in Q_E$  is composed of a set of attribute query atoms  $p_a(var, con) \in P_a$ . Every  $q_{var}$  requires entities  $var$  to have attribute values, matching the specified constants  $con \in V_c$ .
- *Maximum distance*.  $d_{q_x}^{max}$  is the max. distance between an entity query  $q_x \in Q_E$  and the other  $q_y \in Q_E$ , where distance  $d_{q_x}(q_y)$  between  $q_x$  and  $q_y$  is the length of the shortest path of relation atoms connecting the variable nodes  $x$  and  $y$ .
- *Transformed query*.  $q'(Q_E \subseteq V_{q'}, P_r)$  contains entity queries  $q_e \in Q_E$  as nodes and relation query atoms  $p_r \in P_r$  connecting these nodes. This is a compact representation of  $q$ , where attribute query atoms  $p_a$  are collapsed into the entity queries  $q_e$ , i.e., each entity query node in  $q'$  represents a set of attribute query atoms from  $q$ .

For result computation, we select an attribute query edge  $p_a(var, con)$  randomly and create an entity query  $q_{var}$  for the node  $var$ . Other attribute query edges referring to the same node  $var$  are added to  $q_{var}$ . Starting with this entity query node, we construct the transformed query graph  $q'$  by breadth-first-searching (BFS)  $q$ . We add visited relation query atoms as edges to the transformed query, and when attribute query atoms are encountered, we use them to create entity queries in same way as we did for  $p_a(var, con)$ . During the traversal, the length of visited relation chains are recorded. This allows us to compute the distance for every entity query pair. That is, for every entity query  $q_x$ , we compute its distance  $d_{q_x}(q_y)$  to other entity queries  $q_y$ . Finally, the maximum distance is computed for every entity query  $q_x$  from this information, i.e.,  $d_{q_x}^{max} = \arg \max\{d_{q_x}(q_y) : q_x, q_y \in Q_E\}$ .

**Processing Entity Queries.** Every entity query is evaluated by submitting its attribute query atoms as a query against the entity index, i.e.,  $q_e = \{p_{a_1}(e, con_1), \dots, p_{a_n}(e, con_n)\}$  is issued as a conjunction of terms “ $p_{a_1}/con_1, \dots, p_{a_n}/con_n$ ”. We use Lucene as the IR engine for indexing and for answering entity queries specified as keywords. Given  $q_e$ , this engine returns a sorted list of matching entities, where the maximum length of the list is less than a predefined *cutoff* value.

*Example 2.* The query  $q$  shown in Fig. 11c is decomposed into the entity queries  $q_x, q_z, q_u, q_v$ , resulting in the transformed query  $q'$  (Fig. 3a). For this, we start with  $age(x, 29)$  to create  $q_x = \{age(x, 29)\}$ . Then, we traverse the relation edges to obtain  $q' = \{q_x, worksAt(q_x, z), authorOf(q_x, y)\}$ . Further, encountering  $name(z, AIFB)$  results in  $z = q_z = \{name(z, AIFB)\}$ . The process continues for the remaining edges of  $q$ . For entity search, entity queries like  $q_x$  for instance, is submitted as “ $age/29$ ” to obtain the list of entities ( $p1, p3, p5, p6$ ).

## 4 Approximate Structure Matching

So far, the entity query parts of  $q$  have been matched, while the remaining  $p_r$  still have to be processed. Typically, this structure matching is performed by retrieving triples for the entities computed previously (i.e., edges  $e \in E_R$  matching  $p_r$ ), and joining them along  $p_r$ . Instead of an equi-join that produces exactly matched results, we propose to perform a *neighborhood join* based on the intersection of entities' neighborhoods. We now define this novel concept for approximate structure matching and discuss suitable encoding and indexing techniques.

**Definition 3.** *The  $k$ -neighborhood of an entity  $e \in V_E$  is the set  $E_{nb}^e \subset V_E$  comprising entities that can be reached from  $e$  via a path of relation edges  $e_r \in E_R$  of maximum length  $k$ . A neighborhood overlap  $e_1 \bowtie^{nb} e_2$  between two entities  $e_1, e_2$  is an evaluation of the intersection  $E_{nb}^{e_1} \cap E_{nb}^{e_2}$ , and returns true iff  $e_1 \bowtie^{nb} e_2 \neq \emptyset$  s.t.  $e_1$  is connected with  $e_2$  over some paths of relations  $e \in E_R$ , otherwise it returns false. A neighborhood join of two sets  $E_1 \bowtie^{nb} E_2$  is an equi-join between all pairs  $e_1 \in E_1, e_2 \in E_2$ , where  $e_1$  and  $e_2$  are equivalent iff  $e_1 \bowtie^{nb} e_2$  returns true.*

**Managing neighborhood via Bloom filters.** For every entity node  $e \in V_E$ , we compute its  $k$ -neighborhood via BFS. Then, all elements in this neighborhood (including  $e$ ) are stored in the entity index using the *neighborhood* field. We store the neighborhoods of entities as Bloom filters [3], a space-efficient, probabilistic data structure that allows for testing whether an element is a member of a set (i.e., the neighborhood). While false negatives are not possible, false positives are. The error probability is  $(1 - e^{-f \times n/m})^f$ , where  $m$  is the size of the Bloom filter in bits,  $n$  is the number of elements in the set and  $f$  is the number of hash functions used [3]. During the neighborhood computation, we count the number of neighbors  $n$  for each entity, and set the parameter  $m$  and  $f$  according to a probability of false positive that can be configured as needed.

**Approximate matching via Bloom filters.** Checking for connection between two entity queries  $q_{e_1}, q_{e_2} \in Q_E$  can be achieved by loading candidate triples matching query edges  $p_r$  and then performing equijoins between the candidates and the entities  $E_1, E_2$  obtained for  $q_{e_1}, q_{e_2}$ . However, because this may become expensive when a large number of triples match edges  $p_r$ , we propose to check for connections between these entities in an approximate fashion via a neighborhood join  $E_1 \bowtie_{E_{filter}}^{nb} E_2$ . This operates on the Bloom filters associated with the entities only, i.e., does not require retrieval and join of triples. In particular, the join is evaluated by processing  $e_1 \bowtie^{nb} e_2$  for all  $e_1 \in E_1$  and  $e_2 \in E_2$  in a nested loop manner, using the filters of elements in  $E_1$  or  $E_2$  denoted by  $E_{filter}$ .

For processing  $e_1 \bowtie_{e_2}^{nb} e_2$ , we evaluate if  $e_1 \in E_{nb}^{e_2}$  using the filter of  $e_2$ . Performing neighborhood overlap this way requires that the neighborhood index built for  $e_2$  covers  $e_1$ , i.e.,  $k \geq d_{e_2}(e_1)$ . This means that for supporting queries with relation paths of a maximum length  $k_q^{max}$ , we have to provide the appropriate neighborhood index with  $k = k_q^{max}$ . Note that for checking connections

between entities in the lists  $E_1$  and  $E_2$  along a chain of  $k$  query atoms  $p_r$ , only one set of Bloom filters has to be retrieved to perform exactly one neighborhood join, while with the standard approach,  $k + 1$  equi-joins have to be performed on the triples retrieved for all  $p_r$ .

The approximate matching procedure based on this neighborhood join concept is shown in Alg. 1. It starts with the center query node  $q_{center}$ , i.e., the one with lowest *eccentricity* such that maximum distance to any other vertex is minimized (where  $eccentricity(q_x) = d_{q_x}^{max}$ , the distance information computed previously). From  $q_{center}$ , we process the neighbor query nodes by traversing them in depth-first search (DFS) fashion. For every  $q_{neighbor}$  in the current DFS path, we neighborhood join the entities associated with this node with entities in the result table  $A$  (line 9). Note, at the beginning, we marked the center node as  $q_{filter}$ . This is to indicate that filters of  $E_{q_{center}}$  should be used for neighborhood join as long as possible, i.e., until finding out that  $E_{q_{neighbor}}$  is at a distance greater than  $k$ . In this case, we proceed with the filters of  $E_{q_{lastSeen}}$ , the elements lastly processed along the path we currently traversed (line 7). By starting from  $q_{center}$ , we aim to maximize the “reusability” of filters.

*Example 3.* The 2-neighborhoods for  $p_1, p_3, p_5$  and  $p_6$  are shown in Fig. 3a. For instance, for  $p_1$  the neighborhood is obtained by BFS to reach the 1-hop neighbors  $p_3, i_1$  and  $a_1$  and finally, the 2-hops neighbors  $p_5, u_1$  and  $c_1$ . In Fig. 2a, we illustrate the bloom filter encoding of the neighborhood of  $p_3$ , using three hash functions. We start with entities for  $q_x$  (Fig. 3a), as it has the lowest eccentricity of 2, i.e.,  $q_x = q_{center}$ . Via BFS of the query starting from  $q_x$ , we arrive at the 1-hop neighboring query nodes  $q_z$  and  $y$ . First, we use the filters of  $E_{q_x}$  ( $k = 2$ ) to check for overlap between entities  $E_{q_x}$  and  $E_{q_z}$ , i.e., lookup if  $i_1$  is in any of the filters retrieved for  $p_1, p_3, p_5$  and  $p_6$  (Fig. 2b) – to find out that  $e_1 \bowtie^{nb} p_n \neq \emptyset$ , except for  $p_n = p_6$ . Since  $y$  is not an entity query, no processing is required here. When encountering 2-hops neighboring nodes  $q_u$  and  $q_v$ , we find that the current filters

---

**Algorithm 1:** Approximate Matching based on Neighborhood Join
 

---

**Input:** Transformed query  $q'(Q_E \subseteq V_{q'}, p_r(x, y) \in P_r)$ . Every entity query  $q_e \in Q_E$  is associated with a set of entities  $E_{q_e}$ .

**Result:** Table  $A$ , where each row represents a set of connected entities.

```

1  $q_{center} \leftarrow ARGMIN\{eccentricity(q_i) : q_i \in Q_E\}$ 
2  $q_{filter} \leftarrow q_{center}$ 
3  $A \leftarrow E_{q_{center}}$ 
4 while  $\exists q_e \in Q_E : \neg visited(q_e)$  do
5      $q_{neighbor} \leftarrow q_e \in Q_E$  obtained via DFS along  $p_r$  from  $q_{center}$ 
6     if  $d_{q_{filter}}(q_{neighbor}) > k$  then
7          $q_{filter} \leftarrow q_{lastSeen}$ , where  $q_{lastSeen}$  is the one lastly seen along the path
            currently traversed via DFS
8     end
9      $A \leftarrow A \bowtie_{E_{q_{filter}}}^{nb} E_{q_{neighbor}}$ 
10 end
11 return  $A$ 
    
```

---

are still usable, because distance to these nodes  $d_{q_x}(q_u), d_{q_x}(q_v) = k = 2$ . If  $k = 1$  instead, we would need to retrieve the filter of  $i1$  to check for set membership of  $u1$ , i.e., set  $q_{filter} = q_z$  for processing  $q_u$ .

## 5 Structure-Based Result Refinement and Computation

Result of the previous step is a set of tuples. Every tuple is a set of entities that are somehow connected, i.e., connected over some unknown paths. During refinement, we want to find out whether they are really connected via paths captured by query atoms. For this, we propose the *structure-based result refinement*, which helps to refine the previous results by operating against a summary called the structure index. Using the summary, we check if tuples computed in the previous step match query relation paths. If so, the final step called *structure-based result computation* is performed on the refined tuples.

**Structure Index for Graph Structured Data.** Structure indexes have been widely used for semi-structured and XML data [4,13,6]. A well-known concept is the dataguide [9], which is a structural description for rooted data graphs. Dataguide nodes are created for groups of data nodes that share the same incoming edge-labeled paths starting from the root. Similar to this concept, a structure index has been proposed for general data graphs [16]. Nodes in a structure index stand for groups of data elements that have equal structural “neighborhood”, where equal structural neighborhood is defined by the well-known notion of *bisimulation*. Accordingly, two graph nodes  $v_1, v_2$  are *bisimilar* ( $v_1 \sim v_2$ ), if they cannot be distinguished by looking only at their outgoing or incoming “edge-labeled trees”. Pairwise bisimilar nodes form an extension. Applying the bisimulation  $\sim$  to the graph  $G(V, L, E)$  of our data graph that contains relation edges only, results in a set of such *extensions*  $\{[v]^\sim \mid v \in V\}$  with  $[v]^\sim := \{w \in V \mid v \sim w\}$ . These extensions form a complete partition of the entity nodes  $V$  of the data graph, i.e., form a family  $\mathcal{P}^\sim$  of pairwise disjoint sets whose union is  $V$ . Based on this notion of bisimulation, the *structure index graph*  $G^\sim$  of  $G(V, L, E)$  can be defined in terms of extensions and relations between them. In particular, extensions from the partition  $\mathcal{P}^\sim$  form the vertices of  $G^\sim$ . An edge with label  $l$  links  $E_1, E_2 \in \mathcal{P}^\sim$  of  $G^\sim$  iff  $G$  contains an  $l$ -edge linking an element in the extension  $E_1$  to some element in extension  $E_2$ .

*Example 4.* The data graph shown in Fig. 1a can be partitioned into 8 extensions, shown as nodes of the index graph in Fig. 1b. For instance,  $p1$  and  $p3$  are grouped into the extension  $E2$  because they are bisimilar, i.e., both have incoming *supervise* and *knows* links and both have the same outgoing trees (paths) of edges *knows*, (*worksAt, partOf*) and (*authorOf, conference*).

It has been shown that the structure index is appropriate for investigating structures that can be found in the data [16]. In particular, it exhibits a property that is particularly useful for our approach:

**Algorithm 2:** Structure-based Result Refinement using Structure Index

**Input:** Transformed query  $q'(V_{q'}, p_r(q_s, q_t) \in P_r)$ . Entity query nodes  $Q_E \subseteq V_{q'}$ . Table  $A_{m \times n}(q_{e_1}, \dots, q_{e_n})$ , where each row represents a set of somehow connected entities. Structure index graph  $G^\sim(V^\sim, E^\sim)$ .

**Data:**  $EXT_{q_e}(q_e, ext(q_e))$  is a two column table containing the results  $e \in E_{q_e}$  of  $q_e$  and their extensions  $ext(e)$ .  $E^\sim(source(r), target(r))$  is a two column table containing source and target nodes of the edge  $r$ .

**Result:** Refined table of entities  $A$ . Intermediate result table  $M(c_1, \dots, c_n)$  containing entities and entity extensions, where  $c_n$  denotes a query  $q_e$  or an extension  $ext(q_e)$ .

```

1 for  $p_r(q_1, q_2) \in P_r$  do
2    $E^\sim(e_{q_1}, e_{q_2}) \leftarrow \{r^\sim(x, y) \in E^\sim \mid p_r = r^\sim\}$ 
3   for  $q_n \in \{q_1, q_2\}$  do
4     if  $q_n \in Q_E$  then  $E^\sim(e_{q_1}, e_{q_2}) \leftarrow E^\sim(e_{q_1}, e_{q_2}) \bowtie_{q_n} EXT_{q_n}$ 
5   end
6   if  $M = \emptyset$  then  $M = E^\sim(e_{q_1}, e_{q_2})$ 
7   else  $M \leftarrow E^\sim(e_{q_1}, e_{q_2}) \bowtie_{q_n} M$ 
8    $A \leftarrow \pi_{q \in Q_E}(M)$ 
9 end
10 return  $A$  and  $M$ 

```

*Property 1.* If there is a match of a query graph on a data graph  $G$ , the query also matches on the index graph  $G^\sim$ . Moreover, nodes of the index graph matches will contain all data graph matches, i.e., the bindings to query variables.

**Structure-based Result Refinement.** Property 1 ensures that nodes of the index graph matches will contain all data graph matches, i.e., the bindings to query variables. Therefore, entities computed in the previous step can only be answers to the query, when they are contained by some matches of the query on the structure index graph. Drawing upon this observation, Alg. 2 (1) matches the transformed query graph  $q'$  against the structure index and (2) checks if the resulting index graph matches contain the previously computed entities in table  $A$ . For index graph matching, edges  $E^\sim$  of the index graph are retrieved (line 2) and joined along the query atoms  $p_r(q_1, q_2) \in P_r$ . When entity query nodes are encountered, i.e.,  $q_n$  is an element of  $Q_E$ , we check if entities previously computed for  $q_n$  (stored in  $A$ ) are contained in the matching extensions retrieved for  $q_n$ . For this, we use the extensions associated with these entities (as stored in ENT-doc) to construct an extension table  $EXT_{q_n}$  and join this table with  $E^\sim$ . Thereby, extensions that do not contain entities in  $A$  are discarded during the computation. After processing all edges,  $M$  contains only index matches, which connect entities in  $A$ . Finally, by projecting on the attributes  $q_e$ , we obtain the refined entities  $A$  from  $M$  (line 8).

*Example 5.* This example demonstrates refining result table  $A = \{(p1, i1, u1, c1), (p3, i1, u1, c1), (p5, i1, u1, c1)\}$ . The result of the refinement step is one index match (Fig. 3b). To obtain the index match, we can, e.g., start with the query

atom  $supervise(w, q_x)$ . For this, one matching edge  $supervise^\sim = \{(E1, E2)\}$  is retrieved from  $G^\sim$ .  $supervise^\sim$  is joined with the extension table for  $q_x$ , i.e.,  $\{(E1, E2)\} \bowtie_{q_x} \{(E2, p1), (E2, p3)\}$ . This results in  $supervise^\sim = \{(E1, E2, p1), (E1, E2, p3)\}$ , i.e., extension  $E2$  obtained for  $q_x$  contains entities  $p1, p3$  (previously computed for  $q_x$ ). Thus, no match is discarded in this case. We continue with  $authorOf(q_x, y)$  to obtain  $authorOf^\sim = \{(E6, E4), (E2, E4)\}$ . By joining on  $q_x$ , i.e.,  $\{(E6, E4), (E2, E4)\} \bowtie_{q_x} \{(E2, p1), (E2, p3)\}$ , we obtain  $\{(E2, p1, E4), (E2, p3, E4)\}$ , i.e., we discard the extension  $E6$ , as it does not contain  $p1, p3$ . Since  $y$  is not an entity query, we do not need to check if the extension  $E4$  contains entities in  $A$ . Now,  $M = authorOf^\sim \bowtie supervise^\sim$ , i.e.,  $M = \{(E1, E2, p1), (E1, E2, p3)\} \bowtie_{q_x} \{(E2, p1, E4), (E2, p3, E4)\} = \{(E2, E2, p1, E4), (E1, E2, p3, E4)\}$ . This process continues for the remaining query atoms to obtain  $M = \{(E1, E2, p1, E4, E3, i1, E5, u1, E6, c1), (E1, E2, p3, E4, E3, i1, E5, u1, E6, c1)\}$ . Projecting  $M$  on the attributes  $q \in Q_E$  results in  $A = \{(p1, i1, u1, c1), (p3, i1, u1, c1)\}$ .

**Complete Structure-based Result Computation.** Finally, results which exactly match the query are computed by the last refinement. Only for this step, we actually perform joins on the data. To improve efficiency, we do not retrieve and join data along the query atoms in a standard way [1]. Instead, we incrementally refine the results, i.e., reuse the index matches and the entities associated with them as stored in the intermediate result set  $M$ . Given the index graph match  $G_q^\sim$ , the algorithm for result computation iterates through the edges  $l_q^\sim([e_1]^\sim, [e_2]^\sim) \in L^\sim$  of  $G_q^\sim$ , retrieves matching triples, and joins them. However, if results exist, i.e., there are entities contained in  $[e_1]^\sim$  or  $[e_2]^\sim$  such that  $[e_1]^\sim.E \vee [e_2]^\sim.E \neq \emptyset$ , they are taken into account. In particular, only triples  $l_q^m(e_1, e_2)$ , where  $e_1 \in [e_1]^\sim.E$  and  $e_2 \in [e_2]^\sim.E$  are retrieved from the data graph. In Fig. 3c, we see the triples that are retrieved and joined to obtain the final result of the query. Only by inspecting the actual triples along this structure index match, we note that  $p3$  is not connected with the other entities.

## 6 Evaluation

We conducted a complexity analysis for our approach. Given a query graph with bounded size, we can prove that the complexity of query processing is polynomial, which is more promising than the worst-case exponential complexity of exact and complete graph-pattern matching. Due to space reasons, the details were omitted here but can be found in our technical report [5]. In this section, we present empirical performance results and also analyze the efficiency-precision trade-off to shed light on the incremental and approximate features of our approach.

**Systems.** We have implemented the incremental process (INC) based on vertical partitioning and sextuple indexing [11, 17]. To compare our solution with the exact and complete approach [1], we implement sorted-merged equi-join using the

<sup>5</sup> <http://people.aifb.kit.edu/awa/ApproxIncrementalQueryProcessing.eps>

**Table 2.** Statistics for the data graphs and indexes

	Data(#Edges)	Data(MB)	EntityIdx(MB)	RelIdx(MB)	StrucIdx(KB)	Schema(KB)
DBLP	12,920,826	2,084	2210	2,311	132	28
LUBM5	722,987	122	142	112	100	24
LUBM10	1,272,609	215	253	198	80	24
LUBM50	6,654,596	1,132	1391	1,037	82	24

same data partitions and indexes (VP). Since query optimization as proposed for the RDF-3X [14] is orthogonal, all experiments here were performed without optimization, i.e., based on fixed query plans (same for both approaches). There is no appropriate baseline for the approximate and incremental features of our solution. ASM is based on Bloom filter, which has not been applied to this problem of structure matching before. Also, there is no alternative for SRR. We have already pointed out (related work) that, while SRR is based on a summary, which is conceptually similar to the synopsis previously proposed for approximate query processing, it is not clear how to extend these concepts to graph-structured data and in particular, to use them in a pipeline. Our implementation is freely available.<sup>6</sup>

**Datasets.** We used *DBLP*, which captures bibliographic information. Further, we used the LUBM data generator to create 3 datasets for 5, 10 and 50 universities (Table 2). Note that the structure indexes were consistently bigger than the schemas, but were of magnitudes smaller than the data graphs.

**Queries.** For studying the proposed algorithms in a principled way, test queries were generated via random data sampling. We generated queries ranging from simple path-shaped to graph-shaped queries. For this, we use as parameters the maximum number of constants  $con_{max}$ , the maximum number of paths  $p_{max}$ , the maximum path length  $l_{max}$  and the maximum number of cycles  $cy_{max}$  in the query graph. We sampled constants from data values  $V_D$  of the data graph. Paths and cycles were sampled from data graph edges  $E$ . The parameters used in the experiments are  $con_{max} = 20$ ,  $p_{max} = 6$ ,  $l_{max} = 3$ ,  $cy_{max} = 2$ .

**Setting.** We used a machine with two Intel Xeon Dual Core 2.33 GHz processors and 48GB of main memory running Linux (2GB were allocated to JVM). All data and indexes were stored on a Samsung SpinPoint S250 200GB, SATA II. All components have been implemented in Java 5. The bit-vector length and the number of hash functions used for Bloom filter encoding were computed to reach the configured probability of false positive of 0.1%. Neighborhood indexes were created for  $k = 3$ . All times represent the average of 10 runs of 80 queries generated for DBLP, and 80 queries for LUBM. For different steps of INC, we computed the precision using the formula:  $\text{precision} = (|\text{correct results}| \cap |\text{results retrieved}|) / |\text{results retrieved}|$ . A result of an entity query in ES is correct, if it is contained in the respective column of the final result table. The precision for ES is computed as the average precision obtained for all entity query nodes of

<sup>6</sup> <http://code.google.com/p/rdfstores/>



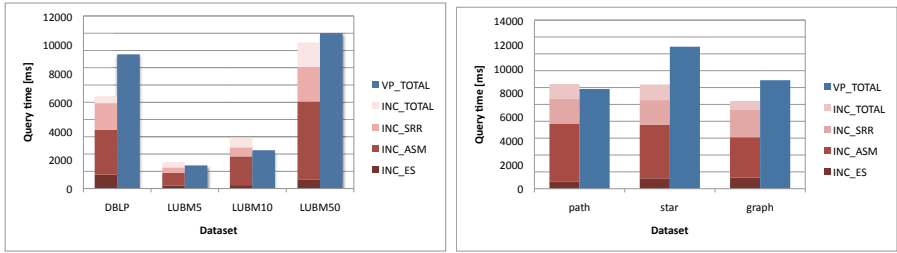


Fig. 4. Query processing times for a) different datasets and b) different query shapes

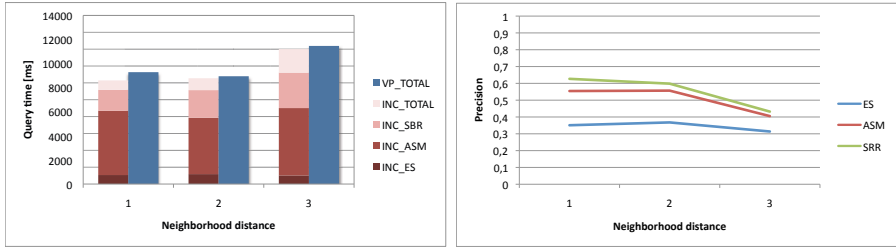
$q$ . A tuple computed during ASM and SRR is correct, if it is contained as a row in the final result table.

**Average Processing Time.** For INC, we decomposed total processing time into times for ES, ASM, SRR and SRC. Averaging the processing time over 80 queries, we obtained the results shown in Fig. 4a. The time needed for ES is only a small fraction of the total time. Times for SRR and SRC make up a greater portion, and ASM constitutes the largest share. Thus, these results suggest that users can obtain an initial set of results in a small fraction of time via ES. In particular, instead of waiting for all exact results, users might spend only 6, 71 or 84 percent of that times when they choose to finish after ES, ASM or SRR respectively. The comparison of total times shows that INC was slower than VP for LUBM5 and LUBM10, but faster for the larger datasets LUBM50 and DBLP. While these results might change with query optimization, this promising performance indicates that our incremental approach was able to effectively reuse intermediate results.

**The Effect of Data Size.** We have measured total time for LUBM of different data sizes (shown in Table 2). As illustrated in Fig. 4a, query processing time increased linearly with the size of the data, for both VP and INC. Further, INC became relatively more efficient as the data size increased. It can be observed that the share of total time from ASM decreased with the data size, i.e., the gain from ASM unfolded as the dataset grew larger. This is particularly important in the Data Web context; ASM can help to quickly obtain initial results from a large amount of data.

**The Effect of Query Complexity.** Considering query complexity, we classified the 80 queries into three classes according to query shape. As shown in Fig. 4b, INC did not perform well on path queries. For this type of queries, ASM was particularly expensive. This is because in many cases, the reusability of Bloom filters was low (i.e., when path length was higher than  $k$ ). Filter loading and nested loop joins became the bottleneck, resulting in slightly higher processing times compared to VP.

**The Effect of Relation Path Length  $k$ .** In another experiment we classified queries into three classes according to the length of the longest relation path (i.e.,



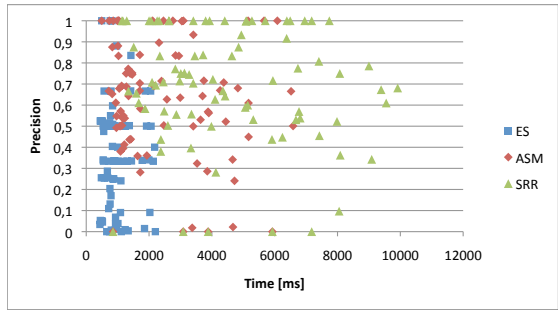
**Fig. 5.** Effect of neighborhood distance on a) processing times and b) precision

the neighborhood distance between entities, respectively). As shown in Fig. 5a, queries with longer relation paths required more time, for both VP and INC. For INC, the share contributed by ASM remained relatively constant, suggesting that this step can be performed efficiently even for long relation paths. Thus, ASM can also help to deal with complex queries with long relation paths.

**Precision.** The average precision for the different steps at various  $k$  is shown in Fig. 5b. The precision for ES was relatively constant (0.3 - 0.4). This was expected, because  $k$  should have no effect on the quality of entity search. For ASM and SRR, precision decreased with larger  $k$ . The highest precision obtained for ASM was 0.56 and this increased to 0.62 after SRR.

### Time-Precision Trade-off.

We illustrate average time and precision for different steps in Fig. 6. Clearly, through the incremental refinement steps, both precision and processing times increased. There are some outliers – however, overall, a trend may be noticed: ES produces fast results at low precision, i.e., below 50 % for most cases. Precision can be largely improved through ASM, i.e., in 30 % of the cases, ASM drove precision from 50 % up to 80 %. For most of these cases (60 %), the amount of additional processing was less than 10 % of total time.



**Fig. 6.** Precision vs. time

## 7 Conclusion and Future Work

We proposed a novel process for approximate and incremental processing of complex graph pattern queries. Experiments suggest that our approach is relatively fast w.r.t exact and complete results, indicating that the proposed mechanism for

incremental processing is able to reuse intermediate results. Moreover, promising results may be observed for the approximate feature of our solution. Initial results could be computed in a small fraction of total time and can be refined via approximate matching at low cost, i.e., a small amount of additional time. We believe that our approach represents the appropriate paradigm, and embodies essential concepts for dealing with query processing on the Web of data, where responsiveness is crucial. At any time, users should be able to decide if and for which results exactness and completeness is desirable. As future work, we will elaborate on ranking schemes, based on which we plan to integrate top- $k$  techniques into the pipeline.

**Acknowledgements.** Research reported in this paper was supported by the German Federal Ministry of Education and Research (BMBF) in the Collab-Cloud (grant 01IS0937A-E) and iGreen (grant 01IA08005K) projects.

## References

1. Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.J.: Scalable semantic web data management using vertical partitioning. In: VLDB, pp. 411–422 (2007)
2. Babcock, B., Chaudhuri, S., Das, G.: Dynamic Sample Selection for Approximate Query Processing. In: SIGMOD Conference, pp. 539–550 (2003)
3. Bloom, B.H.: Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM* 13(7), 422–426 (1970)
4. Buneman, P., Davidson, S., Fernandez, M., Suciu, D.: Adding structure to unstructured data. In: ICDT, pp. 336–350. Springer, Heidelberg (1997)
5. Chakrabarti, K., Garofalakis, M.N., Rastogi, R., Shim, K.: Approximate Query Processing Using Wavelets. In: VLDB, pp. 111–122 (2000)
6. Chen, Q., Lim, A., Ong, K.W.: D(k)-index: an adaptive structural summary for graph-structured data. In: SIGMOD, pp. 134–144. ACM, New York (2003)
7. Corby, O., Dieng-Kuntz, R., Faron-Zucker, C., Gandon, F.L.: Searching the Semantic Web: Approximate Query Processing Based on Ontologies. *IEEE Intelligent Systems* 21(1), 20–27 (2006)
8. Garofalakis, M.N., Gibbons, P.B.: Approximate Query Processing: Taming the TeraBytes. In: VLDB (2001)
9. Goldman, R., Widom, J.: Dataguides: Enabling query formulation and optimization in semistructured databases. In: VLDB, pp. 436–445 (1997)
10. Harth, A., Decker, S.: Optimized Index Structures for Querying RDF from the Web. In: LA-WEB (2005)
11. Hurtado, C.A., Poulouvasilis, A., Wood, P.T.: Ranking approximate answers to semantic web queries. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) *ESWC 2009*. LNCS, vol. 5554, pp. 263–277. Springer, Heidelberg (2009)
12. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top- $k$  query processing techniques in relational database systems. *ACM Comput. Surv.* 11, 1–11 (2008)
13. Kaushik, R., Bohannon, P., Naughton, J.F., Korth, H.F.: Covering indexes for branching path queries. In: SIGMOD, pp. 133–144 (2002)
14. Neumann, T., Weikum, G.: RDF-3X: a RISC-style engine for RDF. *PVLDB* 1(1), 647–659 (2008)

15. Polyzotis, N., Garofalakis, M., Ioannidis, Y.: Approximate xml query answers. In: SIGMOD 2004, pp. 263–274. ACM, New York (2004)
16. Tran, D.T., Ladwig, G.: Structure index for RDF data. In: Workshop on Semantic Data Management at VLDB (September 2010)
17. Weiss, C., Karras, P., Bernstein, A.: Hexastore: sextuple indexing for semantic web data management. In: PVLDB, vol. 1(1), pp. 1008–1019 (2008)
18. Zhang, L., Liu, Q., Zhang, J., Wang, H., Pan, Y., Yu, Y.: Semplore: An IR approach to scalable hybrid query of semantic web data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 652–665. Springer, Heidelberg (2007)

# Conjunctive Query Optimization in OWL2-DL

Petr Křemen and Zdeněk Kouba

Department of Cybernetics, Czech Technical University in Prague, Czech Republic  
{petr.kremen,kouba}@fel.cvut.cz

**Abstract.** Conjunctive query answering is becoming a very important task on the Semantic Web as the adoption of SPARQL query language increases. There is considerable work done in the area of optimizing conjunctive query answering for RDF and OWL2-DL ontologies, in the latter case namely for queries without undistinguished variables. However, there has not been much emphasis on how to handle queries with both distinguished and undistinguished variables efficiently. In this paper, we present a novel algorithm for answering conjunctive queries over OWL2-DL ontologies with undistinguished variables efficiently. These optimizations are implemented in the Pellet query engine and our experimental evaluation shows that the new optimization improves the query performance significantly.

**Keywords:** OWL2-DL, conjunctive query, optimization.

## 1 Introduction

Since the very beginning, lots of research efforts in description logics have been spent to study properties [2], [10] and optimizations [8] of tableau algorithm consistency checks and their basic inference tasks, like *instance checking* (e.g. check whether an individual *Jan* is an instance of a concept *Student*), *instance retrieval* (e.g. give me all married students - instances of the concept  $Student \sqcap \exists hasMarried \cdot \top$ ), or *role fillers retrieval* (e.g. give me all individuals that take the course *Math*). However, in practical applications basic inferences often do not suffice and a richer query language is required. The most widely accepted generalization of the basic inference tasks are *conjunctive queries* [13], [3].

So far, significant attention has been paid to *boolean queries*, i.e. queries that test just the possibility to map a query pattern to all models of an ontology, without retrieving any variable binding. In [13] the problem of answering boolean queries is reduced to the description logics satisfiability for the *ALC* language [2]. To get rid of variables, the authors use *rolling-up technique* to replace a general boolean query with a set of instance retrievals or instance checks. Although for *ALC* the proposed technique works fine, the authors noticed that its generalization beyond *ALC* is problematic and it is possible only (i) for logics employing the *tree-model property* [2], or (ii) for queries that do not contain undistinguished variables in a cycle (we will often handle a query as a *query graph*, see Definition 2 and [13]).

These limitations of boolean query answering using tableau algorithms and rolling-up technique were overcome in [15] by introducing a specialized tableau algorithm to directly check entailment of boolean queries over *SHIQ* ontologies. This modified tableau algorithm extends the standard one for consistency checking in *SHIQ* [12] with a novel blocking condition<sup>1</sup>. However, this technique to be applicable requires all roles in a conjunctive query to be *simple*, i.e. not transitive and without transitive subroles. As discussed in [15], the same approach can be generalized to unions of boolean conjunctive queries in a straightforward manner.

The presence of transitive roles in query atoms is handled in [5] and [6] for description logics *SHIQ* and *SHOQ*. In both works, the respective logic (*SHIQ*/*SHOQ*) is extended with *role conjunction* construct to capture “short-cuts” introduced by a transitive role. This work has shown that the problem of answering conjunctive queries for both of these subsets is decidable. However, there is still no decision procedure for conjunctive query answering in the language as expressive as *SHOIN* (resp. *SROIQ*), a logic backing the OWL-DL [16] (resp. OWL2-DL [1]) language without data types.

Based on this analysis, we complement the current techniques for conjunctive query answering with an optimization technique that significantly decreases execution times when both distinguished and undistinguished variables (see Definition 1) are present. Comparing to [5], [6], and [15], our optimizations are applicable to conjunctive queries without cycles through undistinguished variables for a wide range of description logics, including *SHOIN* and *SROIQ*. For *SHIN*, the presented techniques can be complemented by an ABox summarization [4] to avoid full computation of the completion forests for efficient evaluation of queries without undistinguished variables. The algorithms presented later in this paper were designed on the top of a standard tableau algorithm for consistency checking of OWL2-DL ontologies in the Pellet reasoner and in OWL2Query<sup>2</sup>, our novel expressive query engine for OWL2-DL.

## 2 Preliminaries

We shortly recall notions relevant to query answering techniques described below. First, an overview of *SROIQ* relevant for understanding this paper is introduced. Next, current optimization techniques for basic reasoning services are presented.

### 2.1 Conjunctive Queries for *SROIQ*

For the sake of brevity, we use the description logic syntax instead of OWL2-DL and we do not consider concrete domains and data types. Readers unfamiliar with OWL2-DL, or *SROIQ* should refer to [1] and/or [9].

<sup>1</sup> That preserves the blocking in a tableau algorithm to occur too early and thus ensures existence of a syntactic mapping from a boolean query to each of resulting completion forests if such a mapping exists.

<sup>2</sup> <http://krizik.felk.cvut.cz/km/owl2query>

Consider finite sets  $CN$ ,  $RN$ ,  $IN$  of *concept names*, *role names* and *individuals*, respectively. We denote elements of  $CN$  as  $A_{(k)}$ , elements of  $RN$  as  $S_{(k)}$  elements of  $IN$  and  $\mathbf{i}_{(k)}$ .  $C_{(k)}$  (resp.  $R_{(k)}$ ) denotes a concept (resp. role), built up using *SROIQ* concept constructors, e.g.  $\exists R \cdot C$ , or  $C_1 \sqcap C_2$  (resp. role constructors, e.g.  $R^-$ ). A *SROIQ ontology*  $\mathcal{K}$  is a set of *SROIQ* axioms (defined in [9]), e.g. concept assertion  $C(\mathbf{i})$  or concept subsumption  $C_1 \sqsubseteq C_2$ .

The semantics is defined using a first-order *interpretation*  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ , where  $\Delta^{\mathcal{I}}$  is the interpretation domain and interpretation function  $\cdot^{\mathcal{I}}$  maps elements from  $CN$  ( $RN$ ) to the subsets of  $\Delta^{\mathcal{I}}$  ( $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ ) and elements from  $IN$  to the elements of  $\Delta^{\mathcal{I}}$ . The interpretation function is extended to general concepts/roles according to the semantics of concept/role constructors (defined in [9]), e.g.  $(\exists R \cdot C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ , or  $(C_1 \sqcap C_2)^{\mathcal{I}} = (C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}})$ , or  $(R^-)^{\mathcal{I}} = \{(x, y) \mid (y, x) \in R^{\mathcal{I}}\}$ .

An axiom  $\alpha$  of the form  $C(\mathbf{i})$  (resp.  $C_1 \sqsubseteq C_2$ ) is satisfied by interpretation  $\mathcal{I}$  ( $\mathcal{I}$  is a model of  $\alpha$ ), denoted as  $\mathcal{I} \models \alpha$ , if  $\mathbf{i}^{\mathcal{I}} \in C^{\mathcal{I}}$  (resp.  $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ ) and similarly for other axiom types. A set  $X$  of axioms is satisfied by  $\mathcal{I}$  ( $\mathcal{I}$  is a model of  $X$ ), denoted as  $\mathcal{I} \models X$ , whenever  $\mathcal{I} \models \alpha$  for each  $\alpha \in X$ . An axiom  $\alpha$  is a logical consequence of a set  $X_2$  of axioms ( $X_2 \models \alpha$ ) if  $\mathcal{I} \models \alpha$ , whenever  $\mathcal{I} \models X_2$ . A set  $X_3$  of axioms is consistent if  $\mathcal{I} \models X_3$  for some  $\mathcal{I}$ .

**Definition 1 (Query Syntax).** A conjunctive ABox query  $Q$  (or simply query  $Q$ ) is a list

$$Q = [q_1, \dots, q_M] \quad (1)$$

where each query atom  $q_i$  is of the form  $C(\bullet)$ , or  $R(\bullet, \bullet)$  and each argument  $\bullet$  is either an individual  $\mathbf{i}_k \in IN$ , or a distinguished variable  $?v_k \in \{?v_1, \dots, ?v_N\} = H(Q) \subseteq V(Q)$ , or an undistinguished variable  $!v_k \in V(Q)$ , where the set of all variables  $V(Q)$  is disjoint from all  $CN, RN, IN$ . Furthermore,  $H(Q)$  (resp.  $U(Q)$ , resp.  $I(Q)$ ) is the set of distinguished variables (resp. undistinguished variables, resp. individuals) in  $Q$ . A semi-ground atom is a query atom  $q$  containing no distinguished variable and a boolean query is a query where  $H(Q) = \{\}$ .

A function  $B = \{?v_1 \mapsto \mathbf{i}_1, \dots, ?v_N \mapsto \mathbf{i}_N\}$  is a binding for  $Q$ . Each strict subset of  $B$  is a partial binding for  $Q$ . An application  $q_i|B$  of a binding  $B$  to query atom  $q_i$  results in a new query atom  $q'_i$ , where each variable  $?v_k$  is replaced by an individual  $\mathbf{i}_k = B(?v_k)$ . An application of a binding  $B$  to a query  $Q = [q_1, \dots, q_M]$  is a query  $Q|B = [q_1|B, \dots, q_M|B]$ .

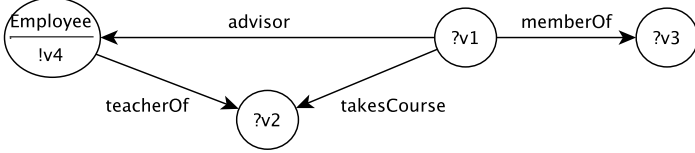
**Definition 2 (Query Graph).** A graph of a conjunctive query  $Q$  is a directed labeled graph  $G_Q = (\mathcal{V}, \mathcal{E}, \mathcal{L})$ , where the set of vertices  $\mathcal{V}$  is the set of distinguished variables, undistinguished variables and individuals that occur in some  $q \in Q$ . The set of directed edges  $\mathcal{E}$  is a set of pairs  $(x, y)$ , where  $R(x, y) \in Q$  and the labeling  $\mathcal{L}$  is defined as  $\mathcal{L}(x) = \{C \mid C(x) \in Q\}$  and  $\mathcal{L}(x, y) = \{R \mid R(x, y) \in Q\}$ . A query  $Q$  has a cycle over  $\mathcal{U} \subseteq \mathcal{V}$  whenever the undirected subgraph  $\text{Sub}(G_Q, \mathcal{U})$  of  $G_Q$  induced by  $\mathcal{U}$  contains a cycle. A query  $Q$  has a cycle, if  $Q$  has a cycle over  $\mathcal{V}$ .

*Example 1.* Let's take the following query (with the query graph shown in Figure 1) into the LUBM dataset [7]. "Find all students ( $?v_1$ ) that take courses

( $?v_2$ ) taught by their advisors ( $!v_4$ ). Retrieve also the courses and the affiliation ( $?v_3$ ) of the students” :

$$Q_1 = [\text{advisor}(?v_1, !v_4), \text{teacherOf}(!v_4, ?v_2), \text{Employee}(!v_4), \\ \text{takesCourse}(?v_1, ?v_2), \text{memberOf}(?v_1, ?v_3)],$$

and thus  $H(Q_1) = \{?v_1, ?v_2, ?v_3\}$  and  $U(Q_1) = \{!v_4\}$ .



**Fig. 1.** Query graph  $G_{Q_1}$  for  $Q_1$ . Although the undirected counterpart of  $G_{Q_1}$  contains a cycle  $?v_1, ?v_2, !v_4$ , the only undistinguished variable in the cycle is  $!v_4$ .

**Definition 3 (Query Semantics).** A boolean query  $Q$  is satisfied by a model  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  of  $\mathcal{K}$  (denoted as  $\mathcal{I} \models Q$ ) if there is an extension  $\nu$  of  $\cdot^{\mathcal{I}}$  that maps each distinguished and undistinguished variable in the query to an element of  $\Delta^{\mathcal{I}}$ , such that for each query atom  $C(x) \in Q$ , resp.  $R(x, y) \in Q$  the following holds:  $\nu(x) \in C^{\mathcal{I}}$ , resp.  $(\nu(x), \nu(y)) \in R^{\mathcal{I}}$ .  $Q$  is a logical consequence of  $\mathcal{K}$ , denoted as  $\mathcal{K} \models Q$ , if  $\mathcal{I} \models Q$  for each model  $\mathcal{I}$  of  $\mathcal{K}$ . A (partial) binding  $B$  for a (possibly non-boolean) query  $Q'$  is valid iff the boolean query comprising all semi-ground atoms from  $Q'|B$  is a logical consequence of  $\mathcal{K}$ .

Intuitively, a boolean query is a logical consequence of an ontology  $\mathcal{K}$ , if the graph (pattern) represented by the query “can be mapped” to each model of  $\mathcal{K}$ .

In the rest of the paper, we assume only conjunctive queries, the graph of which is connected. Each query, a graph of which contains more connected components, can be split into several queries that can be evaluated independently for efficiency reasons and their results combined in the end, see [17]. Note that, as stated before, all techniques described in this paper apply only to queries without cycles over undistinguished variables.

## 2.2 Optimizations of Basic Reasoning Services

For the purpose of this paper a *SROIQ* tableau-based reasoner [11] provides the following optimized services for an ontology  $\mathcal{K}$ , a concept  $C$  and an individual  $i$ :

- consistency checking  $\text{CC}(\mathcal{K})$  returns *true*, iff  $\mathcal{K}$  is consistent, *false* otherwise.
- instance checking  $\text{IC}(\mathcal{K}, C, i)$  returns *true*, iff  $\mathcal{K} \models C(i)$ , *false* otherwise.
- instance retrieval  $\text{IR}(\mathcal{K}, C)$  returns all  $i \in IN$ , for which  $\mathcal{K} \models C(i)$ .



Although both IR and IC can be reduced to one or more CC operations (see [10]), we consider these operations as a well-defined interface to a tableau reasoner, as (i) they are implemented in the state of the art tableau-based reasoners, including Pellet, Fact++<sup>3</sup>, or RacerPro, and (ii) they are very efficiently optimized, as sketched in the rest of this section.

Since both IC and IR services of a *SROIQ* tableau reasoner require the queried ontology to be consistent, let's shortly recall the structures created during an initial consistency check using a tableau algorithm run for *SROIQ*. During a consistency check of a *SROIQ* ontology  $\mathcal{K}$  using a tableau algorithm, a set of *completion graphs* (see [8] and [15]) is evolved by applying *completion rules*. Each of the completion graphs is a finite representation of a (possibly infinite) set of potential models of  $\mathcal{K}$ . If there is no applicable rule and there exists at least one completion graph that does not contain a clash (such a graph is called a *completion*), the algorithm terminates with the result that  $\mathcal{K}$  is consistent. *Obvious non-instances using completion* optimization of IC and IR refuses each  $C(i)$  or  $R(i, j)$  inference that is in clash with *some* completion.

While a completion is a finite representation of one or more models, a *precompletion* represents *all* models of  $\mathcal{K}$ , being the largest subgraph of a completion not depending on any nondeterministic completion rule, like  $\sqcup$ -rule, or  $\leq$ -rule, see [11]. *Obvious instances using precompletion* optimization of IC and IR makes use of precompletions to cache class and property assertion inferences  $C(i)$ , or  $R(i, j)$  that are valid in all models of  $\mathcal{K}$ .

In addition to caching completion and precompletion information, IR can be further optimized using methods presented in [8], namely *binary instance retrieval* and its variants. A naive way of finding all instances of a concept  $C$  is to perform an instance check  $\mathcal{K} \models C(i)$  for each individual  $i$  mentioned in  $\mathcal{K}$  (linear instance retrieval). *Binary instance retrieval* optimization tries to reduce the number of instance checks by checking many individuals being instances of  $C$  during a single tableau algorithm run using divide and conquer strategy.

### 3 Query Evaluation Methods

This section presents methods, that are used in state-of-the-art *SROIQ* tableau reasoners for evaluation of conjunctive queries without cycles over undistinguished variables. Let's have an ontology  $\mathcal{K}$ , a *SROIQ* tableau reasoner as specified in Section 2.2, and a conjunctive query  $Q$  without cycles of undistinguished variables. Before evaluating  $Q$  against  $\mathcal{K}$  it is necessary to check consistency of  $\mathcal{K}$ . If the initial consistency check fails, the query answering procedure stops with an empty result (for non-boolean  $Q$ ) or *false* (for boolean  $Q$ ). If the initial consistency check succeeds, the tableau reasoner constructs the *completion* and *precompletion* structures along the way, as described in Section 2.2. Let's assume that  $\mathcal{K}$  is consistent.

<sup>3</sup> <http://code.google.com/p/factplusplus>

### 3.1 Boolean Queries

A simple way to enforce the semantics of a boolean query  $Q$ , shown in Algorithm [1](#), is presented in [13](#). This algorithm makes use of so called *rolling-up technique*,

---

#### Algorithm 1. Evaluation of Boolean Queries

---

**Input:** consistent  $\mathcal{K}$ , a boolean query  $Q$ .

**Output:** *true* if  $\mathcal{K} \models Q$ , *false* otherwise

```

1: function EVALBOOL( $\mathcal{K}, Q$ )
2:   if some  $q \in Q$  references an individual  $i$  then
3:     if IC( $\mathcal{K}, \text{ROLL}(Q, i), i$ ) then return true
4:   else
5:     if CC( $\mathcal{K} \cup \{\text{ROLL}(Q, !v) \sqsubseteq \perp\}$ ) = false for  $!v \in U(Q)$  then return true
6:   return false

```

---

represented by function  $\text{ROLL}(Q, x)$ , that transforms a boolean query  $Q$ , for which  $\text{Sub}(G_Q, V(Q))$  is a tree<sup>4</sup>, into a single  $C(\bullet)$  query atom. The rolling-up technique describes a query  $Q$  by a complex concept description “from the position of a given term  $x$ ”, by iteratively replacing each query atom  $R(x, y)$  with a query atom  $(\exists R \cdot C_Y)(x)$ , where  $C_Y$  is a concept that represents the rest of the query rolled-up into  $y$  in a similar way. As presented in [13](#), Algorithm [1](#) is a decision procedure for boolean queries  $Q$  for which  $\text{Sub}(G_Q, U(Q))$  is a tree. On line 3 boolean queries with at least one individual are evaluated by a single IC call, while line 5 handles queries with only undistinguished variables as follows: whenever  $\text{ROLL}(Q, !v) \sqsubseteq \perp$  is the cause for inconsistency, it must be the case that  $(\text{ROLL}(Q, !v))^{\mathcal{I}}$  is non-empty in each model  $\mathcal{I}$  of  $\mathcal{K}$ , and thus  $\mathcal{I} \models Q$ .

*Example 2.* Let’s continue with Example [1](#) and consider  $Q_{1g} = Q_1 | B_{12}$ , where  $B_{12} = \{?v_1 \mapsto \text{Jim}, ?v_2 \mapsto \text{Math}, ?v_3 \mapsto \text{DeptMath}\}$  :

$$Q_{1g} = [\text{advisor}(\text{Jim}, !v_4), \text{teacherOf}(!v_4, \text{Math}), \text{Employee}(!v_4), \\ \text{takesCourse}(\text{Jim}, \text{Math}), \text{memberOf}(\text{Jim}, \text{DeptMath})].$$

$Q_{1g}$  can be evaluated by rolling-up e.g. into Jim, obtaining  $\text{ROLL}(Q_{1g}, \text{Jim}) = C_{\text{Jim}}$ , where

$$C_{\text{Jim}} = \exists \text{advisor} \cdot (\text{Employee} \sqcap \exists \text{teacherOf} \cdot \{\text{Math}\}) \sqcap \exists \text{takesCourse} \cdot \{\text{Math}\} \\ \sqcap \exists \text{memberOf} \cdot \{\text{DeptMath}\}$$

and checking  $\text{IC}(\mathcal{K}, C_{\text{Jim}}, \text{Jim})$ , as described in Algorithm [1](#).

---

<sup>4</sup> Individuals are not considered in the subgraph, as they do not violate the applicability of the rolling-up technique: each individual  $i$  can be forked into two vertices to break the cycle without any semantic impact, as  $i^{\mathcal{I}} = \nu(i)$  for any extension  $\nu$  of the interpretation function  $\mathcal{I}$ .

### 3.2 Queries without Undistinguished Variables

The authors of [17] present an algorithm for evaluating queries without undistinguished variables. To make their algorithm compliant with the rest of this paper, we modify their description by introducing two functions, resulting in Algorithm

---

#### Algorithm 2. Eval. of Queries without Undist. Variables

---

**Input:** consistent  $\mathcal{K}$ , a query  $Q = [q_1, \dots, q_N]$ , a binding  $B$  and a set  $\beta$  of valid bindings  $B'$  for  $Q$  found so far.

**Output:** a set  $\beta$  of all valid bindings  $B'$  for  $Q$

```

1: function EVAL( $\mathcal{K}, Q, B, \beta$ )
2:   if  $Q = []$  then return  $\beta \cup \{B\}$ 
3:    $[q, q_{p_1}, \dots, q_{p_{N-1}}] \leftarrow \text{NEXT}(\mathcal{K}, Q, B)$ 
4:   for  $B' \in \text{EVALATOM}(\mathcal{K}, q|B, B)$  do  $\beta \leftarrow \text{EVAL}(\mathcal{K}, [q_{p_1}, \dots, q_{p_{N-1}}], B', \beta)$ 

```

---

2: (i)  $\text{NEXT}(\mathcal{K}, Q, B)$  returns a query that is a reordering of atoms of  $Q$ . As different orderings of the same set of atoms are interpreted equally (see Definition 3) a naive implementation of  $\text{NEXT}(\mathcal{K}, Q, B)$  might return  $Q$ . Reordering query atoms that preserves connectedness of the evaluated query, and minimizes the number of tableau reasoner runs is discussed in [17].

---

#### Algorithm 3. Evaluation of a query atom

---

**Input:** consistent  $\mathcal{K}$ , a query atom  $q$ , binding  $B$

**Output:** a set of all (partial) bindings  $B' \supseteq B$  such that  $\mathcal{K} \models q|B'$

```

1: function EVALATOM( $\mathcal{K}, q, B$ )
2:   if  $q = C(i)$  then return  $\text{IC}(\mathcal{K}, C, i)$ 
3:   if  $q = R(i, j)$  then return  $\text{IC}(\mathcal{K}, (\exists R \cdot \{j\}), i)$ 
4:    $\beta = \{\}$ 
5:   if  $q$  is  $C(?x)$  then
6:     for  $i \in \text{IR}(\mathcal{K}, C)$  do  $\beta \leftarrow \beta \cup \{B \cup \{?x \mapsto i\}\}$ 
7:   else if  $q$  is  $R(?x, i)$  (resp.  $R(i, ?x)$ ) then
8:     for  $j \in \text{IR}(\mathcal{K}, (\exists R \cdot \{i\}))$ , resp.  $(\exists R^- \cdot \{i\})$  do  $\beta \leftarrow \beta \cup \{B \cup \{?x \mapsto j\}\}$ 
9:   else if  $q$  is  $R(?x, ?y)$  then
10:    for  $i \in \text{IR}(\mathcal{K}, (\exists R \cdot \top))$  do
11:      for  $j \in \text{IR}(\mathcal{K}, (\exists R^- \cdot \{i\}))$  do  $\beta \leftarrow \beta \cup \{B \cup \{?x \mapsto i, ?y \mapsto j\}\}$ 
12:   return  $\beta$ 

```

---

(ii) Given the current binding  $B$ , the function  $\text{EVALATOM}(\mathcal{K}, q, B)$ , shown in Algorithm 3, finds all (partial) bindings  $B' \supseteq B$  such that  $\mathcal{K} \models q|B'$ . For example,  $\text{EVALATOM}(\mathcal{K}, \text{teacherOf}(?x, \text{Math}), \{\})$  returns a set  $\{B_k\}$  of bindings  $B_k = \{?x \mapsto i_k\}$ , where  $i_k \in \text{IR}(\mathcal{K}, \exists \text{teacherOf} \cdot \{\text{Math}\})$ . Algorithm 2 is

executed by a call  $\text{EVAL}(\mathcal{K}, Q, \{\}, \{\})$  that recursively searches the state space of possible (partial) bindings and backtracks once all query atoms are evaluated. The soundness of Algorithm 2 and Algorithm 3 is presented in 17.

### 3.3 Handling Undistinguished Variables

In practical applications queries with both distinguished and undistinguished variables are the most common. This section describes techniques for evaluating conjunctive queries without cycles over undistinguished variables that were implemented in the Pellet reasoner, yet not published so far. Techniques described in this section try to reduce the number of calls to IC (some of which might require consistency checks). As shown in the examples below, even if obvious non-instances using completion and obvious instances using precompletion optimizations mentioned in Section 2.2 prevent other than a single (initial) consistency check to occur, the number of IC calls might be still prohibitive for an efficient evaluation.

**Naive Evaluation Strategy.** The naive way to evaluate conjunctive queries is described in Algorithm 4. On line 3, each distinguished variable is replaced with

---

#### Algorithm 4. Naive Evaluation Strategy

---

**Input:** consistent  $\mathcal{K}$ , a non-boolean query  $Q$

**Output:** a set  $\beta$  of all valid bindings  $B'$  for  $Q$

```

1: function EVALNAIVE( $\mathcal{K}, Q$ )
2:    $\beta \leftarrow \{\}$ 
3:   for each  $B = \{?v_1 \mapsto i_1, \dots, ?v_N \mapsto i_N\}, ?v_k \in H(Q), i_k \in IN$  do
4:     if IC( $\mathcal{K}, \text{ROLL}(Q|B, i_1), i_1$ ) then  $\beta \leftarrow \beta \cup \{B\}$ 
5:   return  $\beta$ 

```

---

an individual mentioned in  $\mathcal{K}$  and the resulting boolean query is evaluated using the EVALBOOL function described in Algorithm 1. The Algorithm 4 is clearly sound, as it simply tries all possible bindings. However, as shown in Example 3, the exponential blow-up is what makes it unusable for queries with more than one distinguished variable.

*Example 3.* Let's evaluate  $Q_1$ , see Example 1, against the LUBM(1) dataset. As LUBM(1) contains more than 17000 individuals the variable substitution results in about  $17000^3 = 5 \times 10^{12}$  of different boolean queries to be checked on line 4 of Algorithm 4. Even if checking logical consequence of each of them might be cheap in this case (the boolean queries can be matched against precompletion and thus the only interaction with the tableau reasoner remains the initial consistency check that constructs the precompletion),  $Q_1$  evaluation still fails to terminate within reasonable time due to the huge number of required IC operations.

**Simple Evaluation Strategy.** As noticed in [17], for queries with distinguished variables, the *rolling-up technique* can be used as a preprocessing step to reduce the number of boolean queries to be tested using EVALBOOL function. This approach results in Algorithm 5. For each distinguished variable  $?v$  a concept  $\text{ROLL}(Q, ?v)$  is computed and instances of this concept are retrieved using the optimized IR service (line 3), as mentioned in Section 2.2. Then, each individual from the retrieved set  $IN_v = IR(\mathcal{K}, \text{ROLL}(Q, ?v))$  is used as a candidate for  $?v$  in the subsequent EVALBOOL calls. The soundness of this algorithm is ensured by the fact that the rolling-up technique typically describes only a subset of  $Q$  (as it breaks some of the cycles), but not a superset and thus cannot discard any binding that would be valid for  $Q$ .

---

**Algorithm 5.** Simple Evaluation Strategy

---

**Input:** consistent  $\mathcal{K}$ , a non-boolean query  $Q$

**Output:** a set  $\beta$  of all valid bindings  $B'$  for  $Q$

```

1: function EVALSIMPLE( $\mathcal{K}, Q$ )
2:    $\beta \leftarrow \{\}, \delta \leftarrow \{\}$ 
3:   for  $?v \in H(Q)$  do  $\delta(?v) \leftarrow IR(\mathcal{K}, \text{ROLL}(Q, ?v))$ 
4:   for each  $B = \{?v_1 \mapsto i_1, \dots, ?v_N \mapsto i_N\}, ?v_k \in H(Q), i_k \in \delta(?v_k)$  do
5:     if  $IC(\mathcal{K}, \text{ROLL}(Q|B, i_1), i_1)$  then  $\beta \leftarrow \beta \cup \{B\}$ 
6:   return  $\beta$ 

```

---

*Example 4.* Evaluating  $Q_1$  against LUBM(1) using Algorithm 5,  $Q_1$  is rolled up into each distinguished variable  $?v_1, ?v_2$  and  $?v_3$ . For  $?v_3$ , we get :

$$\text{ROLL}(Q_1, ?v_3) = (\exists \text{memberOf}^- \cdot (\exists \text{advisor} \cdot (\text{Employee} \\ \sqcap \exists \text{teacherOf} \cdot \exists \text{takesCourse}^- \cdot \top \sqcap \exists \text{takesCourse} \cdot \top)))$$

For each  $v_i$  an IR call is required. Due to the optimizations sketched in Section 2.2, the number of EVALBOOL (and thus IC) calls is significantly less than in Algorithm 4. In case of Pellet, none of these instance retrieval executions requires a consistency check and prune the number of candidates to about 3300 for  $?v_1$ , about 1500 for  $?v_2$  and 15 for  $?v_3$ . Thus, the number of boolean queries, logical consequence of which is to be checked by EVALBOOL, is reduced to about  $3300 \times 1500 \times 15$ , i.e.  $75 \times 10^6$ , still with just one (initial) consistency check.

On the other hand, each partial combination of invalid bindings is checked many times on line 5 of Algorithm 5. However, as shown in the next section, it is not necessary to try to extend partial binding  $(?v_1 \mapsto \text{Jim}, ?v_2 \mapsto \text{Math})$  with a binding for  $?v_3$ , whenever  $\mathcal{K} \models \text{takesCourse}(\text{Jim}, \text{Math})$  does not hold. This observation could significantly reduce the number of EVALBOOL calls on line 5.

## 4 Optimizing Rolling-Up Technique

None of the techniques presented in Section 3.3 takes into account the partial bindings for IR and IC operations which causes the rolling-up technique to lose

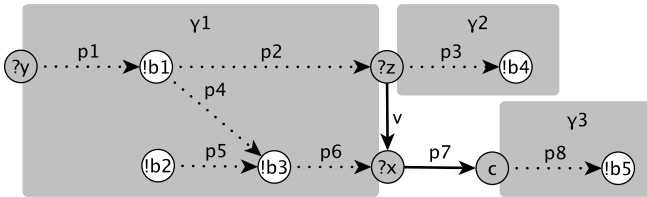
information about distinguished variable bindings of all but one (the variable to which the query is rolled-up) distinguished variables in the query. In this section we present a novel technique that makes use of the current binding to make the IR calls more selective and reduces the number of IC calls.

### 4.1 Cores

The main idea of the core evaluation strategy is that parts of a query  $Q$  that contain undistinguished variables can be localized to *cores* that are evaluated separately (as special query atoms  $CORE_\gamma$ , see below), while the rest of the query is evaluated as a query without undistinguished variables using Algorithm 2. As a result, invalid partial bindings are pruned at early stages of the query processing, even before other distinguished variables and some (or all) of the undistinguished variables have to be evaluated.

**Definition 4 (Core).** Consider a query  $Q = [q_1, \dots, q_Z, q_{Z+1}, \dots, q_N]$  with  $H(Q) \neq \{\}$  and  $U(Q) \neq \{\}$ , where no query atom from  $D(Q) = [q_1, \dots, q_Z]$  contains an undistinguished variable, each query atom from  $C(Q) = [q_{Z+1}, \dots, q_N]$  contains an undistinguished variable. A core is a query  $\gamma \subseteq C(Q)$ , such that graph  $G_\gamma$  of  $\gamma$  is a maximal connected component of the graph  $G_{C(Q)}$ . A signature  $sig(\gamma) = H(Q) \cup I(Q)$  of  $\gamma$  is a set of all distinguished variables and individuals referenced in  $\gamma$ .

Note that the ordering of query atoms in  $Q$  in Definition 4 is not a limitation, as all query atom orderings are interpreted equally, see Definition 3. Definition 4 shows how to construct cores: connected components are built from  $G_{C(Q)}$ , a complex example of which is shown in Figure 2.



**Fig. 2.** Extracting cores from a query  $Q_X = [p7(?x, c), v(?z, ?x), p1(?y, !b1), p2(!b1, ?z), p3(?z, !b4), p4(!b1, !b3), p5(!b2, !b3), p6(!b3, ?x), p8(c, !b5)]$ . Dotted arrows represent the edges of  $G_{C(Q_X)}$  that build up the cores  $\gamma_1, \gamma_2, \gamma_3$ , while simple arrows represent edges of  $G_{D(Q_X)}$ . The grey rounded rectangles remark the cores extracted from the  $Q_X$  (maximal connected components of  $G_{C(Q_X)}$ ).

## 4.2 Core Evaluation

Introduction of cores allows us to transform the query  $Q = [q_1, \dots, q_Z, q_{Z+1}, \dots, q_N]$  into a new query  $Q' = [q_1, \dots, q_Z, CORE_{\gamma_1}, \dots, CORE_{\gamma_K}]$  with  $U(Q') = \{\}$  as all atoms from  $Q$  with undistinguished variables are replaced with  $CORE$  query atoms, one for each core  $\gamma_1, \dots, \gamma_K$  extracted from  $Q$ . The transformed query  $Q'$  can be evaluated by Algorithm 6.

---

### Algorithm 6. Core Evaluation

---

The same as Algorithm 2 introduced in Section 3.2, where the call EVALATOM in line 4 is replaced with the call EVALATOMUV.

---

The function EVALATOMUV is an extended version of EVALATOM that handles also atoms of type  $CORE_{\gamma}$ . The function returns all (partial) bindings  $B' \supseteq B$  such that (i)  $\mathcal{K} \models q|B'$  if  $q = C(\bullet)$  or  $q = R(\bullet, \bullet)$ , like EVALATOM(), or (ii)  $B'$  is a valid binding for  $\gamma$  if  $q = CORE_{\gamma}$ .

---

### Algorithm 7. Evaluation of an Atom in a Transformed Query

---

**Input:** consistent  $\mathcal{K}$ , a regular query atom  $q$  or a  $CORE_{\gamma}$  atom, current binding  $B$

**Output:** a set of all (partial) bindings  $B' \supseteq B$  such that  $\mathcal{K} \models q|B'$ , resp.  $\mathcal{K} \models \gamma|B'$

```

1: function EVALATOMUV( $\mathcal{K}, q, B$ )
2:   if  $q$  is  $CORE_{\gamma}$  then
3:      $\beta = \{\}$ 
4:      $\gamma' \leftarrow \gamma|B$ 
5:     if  $H(\gamma') = \{\}$  and EVALBOOL( $\mathcal{K}, \gamma'$ ) then
6:        $\beta \leftarrow \beta \cup \{B\}$ 
7:     else
8:       for  $B' \in$  EVALSIMPLE( $\mathcal{K}, \gamma'$ ) do
9:          $\beta \leftarrow \beta \cup \{B'\}$ 
10:    return  $\beta$ 
11:  else
12:    return EVALATOM( $\mathcal{K}, q, B$ )

```

---

*Example 5 (Query Transformation).* Using the core evaluation technique for evaluating  $Q_1$  from Example 4 requires its transformation into a single core  $\gamma$  and the transformed version  $Q'_1$  of  $Q_1$  :

$$\gamma = [Employee(!v_3), advisor(?v_1, !v_4), teacherOf(!v_4, ?v_2)],$$

$$Q'_1 = [advisor(?v_1, ?v_2), memberOf(?v_1, ?v_3), CORE_{\gamma}].$$

At this point  $Q'_1$  is evaluated using the EVAL function in Algorithm 6. Thus, first candidate bindings for  $?v_1$ ,  $?v_2$  and  $?v_3$  are pruned iteratively using optimized IR operations (lines 8, 10 and 11 of Algorithm 3) instead of checking

each candidate binding  $B$  of variables  $?v_1, ?v_2$  and  $?v_3$  using IC (line 5 of Algorithm 5). Next, when evaluating  $CORE_\gamma$  a single IC (line 7 of Algorithm 7) is required for each boolean  $\gamma|B$ , the number of which is significantly less than in Example 4.

**Proposition 1 (Correctness).** *Algorithm 6 generates the same set of results for query  $Q' = [q_1, \dots, q_Z, CORE_{\gamma_1}, \dots, CORE_{\gamma_K}]$  as the Algorithm 4 (and 5) for query  $Q = [q_1, \dots, q_Z, q_{Z+1}, \dots, q_N]$ , where  $Q'$  is a transformation of  $Q$  as described in Section 4.2.*

*Proof.* The following reasoning relies on the soundness of Algorithm 1 for boolean queries, Algorithm 2 for queries without undistinguished variables and Algorithm 4, Algorithm 5 for queries with undistinguished variables. Let's take a binding  $B$  valid for  $Q$ , found by Algorithm 4. Since  $D(Q) = [q_1, \dots, q_Z]$  is a subquery (without undistinguished variables) of both  $Q$  and  $Q'$ ,  $B$  is valid for  $D(Q)$  and thus it will be found by Algorithm 2 for queries with distinguished variables. For each atom  $CORE_{\gamma_j}$ , the query  $\gamma_j$ , for  $1 \leq j \leq K$  is constructed *only* from some atoms of  $C(Q) = [q_{Z+1}, \dots, q_N]$ , and is evaluated with Algorithm 7, which passes the core to the function EVALBOOL on line 5, or to function EVALSIMPLE, on line 8. Thus, since  $B$  is valid for  $Q$ , it must be also valid for  $\gamma_j$ .

Let's take a binding  $B$  found by Algorithm 6. Since Algorithm 2 is sound,  $B$  is valid for  $D(Q)$ , as functions EVALATOM and EVALATOMUV do not differ on atoms without undistinguished variables. Since each atom of  $C(Q)$  is present in some core  $\gamma_j$ , for  $1 \leq j \leq K$ , it must have been evaluated by function EVALBOOL, or EVALSIMPLE in Algorithm 7, when evaluating a core atom  $CORE_{\gamma_j}$ . Therefore,  $B$  is valid for  $C(Q)$  and thus also for  $Q$ .

The core evaluation tries to evaluate as many query atoms as possible using the atom-by-atom state space search for queries with distinguished variables (Algorithm 2). This allows the algorithm to apply of the current variable binding to the query to make the IC and namely IR operations as selective as possible. The nature of Algorithm 2 allows for application of further optimizations that reflect the query shape, e.g. query reordering, described in 17.

## 5 Experiments

This section evaluates the described optimization techniques, implemented in the query engine of the open-source OWL2-DL reasoner Pellet, on two benchmark datasets. The LUBM dataset 7 has expressivity  $\mathcal{SHI}(D)$ . The dataset has a dominant ABox part with approx. 17000 individuals in LUBM(1), while the TBox consists of just several tens of classes and properties. The second dataset is UOB 14, a  $\mathcal{SHOIN}(D)$  extension of LUBM. Since neither of the benchmarks contains conjunctive queries with undistinguished variables a new query set was created.

Before executing each of the queries in the subsequent sections an initial consistency check is performed. Its execution times range from 800 ms to 1100 ms for LUBM and from 1800 ms to 2100 ms for UOB. All tests were run on Intel(R) Core(TM)2 CPU 6400 at 2.13GHz with 3GB RAM.



## 5.1 Performance of the Undistinguished Variables Optimizations

To present the efficiency of the core evaluation strategy, we will use the following queries. The chosen query set does not contain any query that is itself just a single core, since for those queries the core evaluation performance is the same for both the original execution and the core evaluation strategy. As can be observed from the nature of the core evaluation strategy in Section 4.2, Algorithm 6 is beneficial only for queries that contain some non-core atoms.

*Example 6 (An acyclic query with one undistinguished variable).* “Retrieve all teachers of some course that is taken by at least one student.”

$$Q_2 = [teacherOf(?v_1, ?v_2), takesCourse(!v_3, ?v_2)]$$

*Example 7 (Difference between dist. and undist. variables).* “Get all members of a research group together with courses they take and teachers of these courses”.

$$Q_3 = [worksFor(?v_1, ?v_2), ResearchGroup(?v_2), \\ takesCourse(?v_1, ?v_3), teacherOf(?v_4, ?v_3)]$$

$Q_4$  is the same as  $Q_3$  but with  $?v_2$  replaced by  $!v_2$ .

*Example 8 (A query with two undistinguished variables, one of which is in a cycle (a modified version of  $Q_1$ )).* “Get all students that are members of some part of some organization and whose advisor teaches a course they take. Get also the courses.”:

$$Q_5 = [advisor(?v_1, !v_4), teacherOf(!v_4, ?v_2), Employee(!v_4), \\ takesCourse(?v_1, ?v_2), memberOf(?v_1, !v_3)]$$

In addition to these queries, we augment the test set with  $Q_1$  to complete the analysis in Example 1. The test results are shown in Table 16.

Both the LUBM and UOB performance results show that the more distinguished variables the query contains, the worse the performance of the original execution is and that the core optimization technique overtakes the optimized strategy by at least an order of magnitude.

The importance of undistinguished variables is presented by queries  $Q_3$  and  $Q_4$ . For both LUBM and UOB the query  $Q_3$  has no results, since there is no person working for an explicitly asserted *ResearchGroup*. On the other hand,  $Q_4$  that matches also inferred *ResearchGroup* instances has more than 1000 results in both cases.

<sup>5</sup> The UOB vocabulary slightly differs from the LUBM one. First, both ontologies have different namespaces that are omitted to improve readability of the paper. Second, some classes and properties have different names, in our case *memberOf* and *advisor* in LUBM correspond to *isMemberOf* and *isAdvisedBy* in UOB. For UOB all presented queries were adjusted accordingly.

**Table 1.** Performance evaluation of the core strategy over the LUBM(1) and UOB(1) DL dataset. The rows labeled *simple* denote the simple evaluation strategy as described in Example 1, while the rows labeled with *core* denote the evaluation using cores. Each row corresponds to the average over 10 independent runs. The query evaluation took *Time* ms (without the initial consistency check), *results* denotes the number of bindings valid for the query and  $NB = (N_{IC} + |IN|N_{IR})/10^3$ , where  $N_{IC}$  is the count of *IC* calls and  $N_{IR}$  is the count of *IR* calls.

	LUBM(1)			UOB(1)			engine
	results	<i>NB</i>	time [ms]	results	<i>NB</i>	time [ms]	
$Q_1$	208	622673	74030	184	569650	132020	<i>simple</i>
		30028	1800		54562	5100	<i>core</i>
$Q_2$	1621	875340	5180	2256	1573345	10510	<i>simple</i>
		3249	910		4517	2280	<i>core</i>
$Q_3$	0	541	330	0	878	800	<i>simple</i>
		541	310		878	380	<i>core</i>
$Q_4$	1099	924238	26130	2262	1581081	15010	<i>simple</i>
		24216	1320		46838	3760	<i>core</i>
$Q_5$	208	4998812	38150	184	6015688	45512	<i>simple</i>
		30028	2360		54562	4720	<i>core</i>

## 6 Conclusion

Introduced optimization techniques made evaluation of queries with undistinguished variables efficient enough to be compared to evaluation of conjunctive queries with distinguished variables. Still, during the implementation of these optimization in the Pellet reasoner it turned out that evaluation of cores is still the major bottleneck for queries with undistinguished variables. Thus next work will study the overall impact of different core evaluation strategies to avoid as many instance checks as possible.

The presented algorithm was implemented and tested on top of a tableau algorithm for OWL2-DL consistency checking in the Pellet reasoner and the OWL2Query engine. Still, its integration with another tableau-based inference engine, like [15], to handle cycles of undistinguished variables seems beneficial and will be studied in detail in the future.

**Acknowledgements.** The authors would like to thank Evren Sirin for his help during integration of the proposed techniques into the Pellet reasoner. Financially, this work was supported partially by the grant No. MSM 6840770038 “Decision Making and Control for Manufacturing III” of the Ministry of Education, Youth and Sports of the Czech Republic and partially by the grant No. 2/2011/OVV “Defects in immovable cultural heritage objects: a knowledge-based system for analysis, intervention planning and prevention” of the Ministry of Culture of the Czech Republic.

## References

1. OWL 2 Web Ontology Language Document Overview (2009), <http://www.w3.org/TR/2009/PR-owl2-overview-20090922/>
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Peter, P.-S.: The Description Logic Handbook, Theory, Implementation and Applications, Cambridge (2003)
3. Calvanese, D., De Giacomo, G., Lenzerini, M.: On the Decidability of Query Containment under Constraints. In: Proceedings of Principles on Database Systems 1998, pp. 149–158 (1998)
4. Dolby, J., Fokoue, A., Kalyanpur, A., Ma, L., Schonberg, E., Srinivas, K., Sun, X.: Scalable Grounded Conjunctive Query Evaluation over Large and Expressive Knowledge Bases. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 403–418. Springer, Heidelberg (2008)
5. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive Query Answering in the Description Logic *SHIQ*. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007 (2007)
6. Glimm, B., Horrocks, I., Sattler, U.: Query entailment for *SHOQ*. In: Proc. of the 2007 Description Logic Workshop (DL 2007). CEUR Electronic Workshop Proceedings, vol. 250, pp. 65–75 (2007), <http://ceur-ws.org/Vol-250>
7. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics* 3(2-3), 158–182 (2005)
8. Haarslev, V., Möller, R.: On the scalability of description logic instance retrieval. *J. Autom. Reason.* 41(2), 99–142 (2008)
9. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *sroiq*. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) KR, pp. 57–67. AAAI Press, Menlo Park (2006)
10. Horrocks, I., Patel-Schneider, P.F.: Reducing OWL entailment to description logic satisfiability. *J. Web Sem.* 1(4), 345–357 (2004)
11. Horrocks, I., Sattler, U.: A tableaux decision procedure for *SHOIQ*. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 448–453. Morgan, San Francisco (2005)
12. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with individuals for the description logic *SHIQ*. In: McAllester, D. (ed.) CADE 2000. LNCS, vol. 1831, Springer, Heidelberg (2000)
13. Horrocks, I., Tessaris, S.: A conjunctive query language for description logic aboxes. In: AAAI/IAAI, pp. 399–404 (2000)
14. Ma, L., Yang, Y., Qiu, Z., Xie, G.T., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 125–139. Springer, Heidelberg (2006)
15. Ortiz, M., Calvanese, D., Eiter, T.: Data complexity of query answering in expressive description logics via tableaux. *J. Autom. Reason.* 41(1), 61–98 (2008)
16. Patel-Schneider, P.F., Hayes, P., Horrocks, I.: Owl web ontology language semantics and abstract syntax section 5. rdf-compatible model-theoretic semantics. Technical report, W3C (December 2004)
17. Sirin, E., Parsia, B.: Optimizations for answering conjunctive ABox queries. *Description Logics* (2006)

# RoSeS: A Continuous Content-Based Query Engine for RSS Feeds<sup>\*</sup>

Jordi Creus Tomàs<sup>1</sup>, Bernd Amann<sup>1</sup>, Nicolas Travers<sup>2</sup>, and Dan Vodislav<sup>3</sup>

<sup>1</sup> LIP6, CNRS – Université Pierre et Marie Curie, Paris, France

<sup>2</sup> Cedric/CNAM – Conservatoire National des Arts et Métiers, Paris, France

<sup>3</sup> ETIS, CNRS – University of Cergy-Pontoise, Cergy, France

**Abstract.** In this paper we present RoSeS (Really Open Simple and Efficient Syndication), a generic framework for content-based RSS feed querying and aggregation. RoSeS is based on a data-centric approach, using a combination of standard database concepts like declarative query languages, views and multi-query optimization. Users create personalized feeds by defining and composing content-based filtering and aggregation queries on collections of RSS feeds. Publishing these queries corresponds to defining views which can then be used for building new queries / feeds. This naturally reflects the publish-subscribe nature of RSS applications. The contributions presented in this paper are a declarative RSS feed aggregation language, an extensible stream algebra for building efficient continuous multi-query execution plans for RSS aggregation views, a multi-query optimization strategy for these plans and a running prototype based on a multi-threaded asynchronous execution engine.

## 1 Introduction

In its origins the Web was a collection of semi-structured (HTML) documents connected by hypertext links. This vision has been valid for many years and the main effort for facilitating access to and publishing web information was invested in the development of expressive and scalable search engines for retrieving pages relevant to user queries. More recently, new web content publishing and sharing applications that combine modern software infrastructures (AJAX, web services) and hardware technologies (handheld mobile user devices) appeared on the scene. The web contents published by these applications is generally evolving very rapidly in time and can best be characterized by a stream of information entities. Online media, social networks and microblogging systems are among the most popular examples of such applications, but the list of web applications generating many different kinds of information streams is increasing every day.

In our work we are interested in RSS [1] and ATOM [17] as standard formats for publishing information streams. Both formats can be considered as the continuous counterpart of static HTML documents for encoding semi-structured data streams in form of

---

<sup>\*</sup> The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ROSES (ANR-07-MDCO-011) “Really Open, Simple and Efficient Syndication”.

<sup>1</sup> RSS stands for (1) Rich Site Summary, (2) RDF Site Summary and (3) Really Simple Syndication.

dynamically evolving documents called *feeds*. They both use very similar data models and follow the design principles of web standards (openness, simplicity, extensibility, genericity) for generating advanced web applications.

The following example illustrates the usage of RSS<sup>2</sup> in the context of social networks. Consider some user Bob who is registered to various different social media sites like *Facebook*, *Twitter*, *Blogger*, *Flickr*, *YouTube* etc. as a publisher, as a reader or both. All of these sites propose different kinds of services for publishing and annotating any kind of web contents (web pages, images, videos). Bob rapidly feels the need of a unique interface for observing the different information streams at a glance. This kind of service can be provided by social web sites or other mashup interfaces like *NetVibes* or *iGoogle* in form of widgets for building mashup pages. A more flexible data-centric solution consists in aggregating RSS streams. Instead of building mashup pages merging different information streams, Bob can use an RSS aggregator like for example *Google Reader* to subscribe to different kinds of RSS services proposed by the various social networks and media sites. This aggregator provides a uniform interface for creating new personalized information streams composing and filtering items published by collections of source feeds. A more detailed description of how RSS aggregation can be used for combining social information streams is described in Section 4.1.

Another widely used application of RSS concerns news aggregation. RSS has become the standard format used by professional information publishers like journals and magazines for broadcasting news on the web. Combined with an adapted publish-subscribe middle-ware, this offers an efficient way for the personalized delivery of news. News aggregators like *Google News* enable personalization by defining and publishing RSS views over collections of news feeds. Each such view is defined by a declarative query which merges and filters news of a possibly important number of source feeds (for example all major French journals). Following the publish-subscribe principle, these views can be reused for building other streams and the final result is an acyclic graph of union/filtering queries on all the sources. One of the issues tackled in this paper concerns the optimization of such plans.

In this paper we present *RoSeS* (Really Open Simple and Efficient Syndication), a generic framework for content-based RSS feed querying and aggregation. Our framework is based on a data-centric approach, based on the combination of standard database concepts like declarative query languages, views and multi-query optimization. Users create personalized feeds by defining and composing content-based filtering and aggregation queries on collections of RSS feeds. Publishing these queries corresponds to defining views which can then be used for building new queries / feeds. This naturally reflects the publish-subscribe nature of RSS applications. *RoSeS* is based on a simple but expressive data model and query language for defining continuous queries on RSS streams. Combined with efficient web crawling strategies and multi-query optimization techniques, it can be used as a continuous RSS stream middle-ware for dynamic information sources. The main contributions presented in this paper are:

- A declarative RSS feed aggregation language for publishing large collections of structured queries/views aggregating RSS feeds,

---

<sup>2</sup> In the following we will use the term *RSS* for both formats, *RSS* and *ATOM*.

- An extensible stream algebra for building efficient continuous multi-query execution plans for RSS streams,
- A flexible and efficient cost-based multi-query optimization strategy for optimizing large collections of publication queries,
- A running prototype based on multi-threaded asynchronous execution of query plans.

The rest of the paper is organized as follows. Section 2 compares our system with other existing RSS aggregation systems and introduces related work on data stream processing and multi-query optimization. Section 3 describes the overall RoSeS architecture. The RoSeS data model, language and algebra are presented in Section 4 and correspond to the first important contribution of our paper. Section 5 is devoted to the second important contribution about query processing and multi-query optimization. Section 6 discusses future work.

## 2 Related Work

*Content-based Feed Aggregation:* There exists a huge number of tools and portals providing different kinds of services for using RSS feeds. These services range from simple web browser extensions for subscribing to and consulting RSS feeds to rich news aggregation web sites like Google News for building personalized information spaces by filtering and clustering news generated by hundreds or thousands of media sites. Feed registries like FeedZilla, Search4rss and Syndic8 maintain collections of RSS feeds which can be accessed by simple keyword search interfaces and a non-personalizable set of hierarchically organized categories for describing and retrieving feeds. Aggregation is often limited to the visual agregation of widgets in a web page (Netvibes, Feedzilla) or the creation of collections (Google Reader).

Opposed to graphical and procedural feed agregation techniques, we are interested in content-based feed filtering and aggregation for building personalized feeds that can be shared with other users. GoogleReader<sup>3</sup> is a RSS feed registry and feed reader which allows Google users to create a personalized hierarchy of RSS feed collections. Feeds can be discovered by keyword search and new feeds can be added manually. GoogleReader also integrates the Google social network features for sharing and recommending feeds. With respect to content-based feed aggregation, Google Reader allows users to publish all items generated by a collection of feeds as a new ATOM feed and provides a search interface for searching items in user defined collections. It is worth mentioning that it is not possible to publish such filtered collections as new feeds.

The aggregation approach proposed by YahooPipes!<sup>[1]</sup> is probably the most similar to ours. YahooPipes! is a web application for building so-called pipes generating *mashups* from several RSS feeds and other external web sources/services. A pipe is a visual representation of the *Yahoo! Query Language* (YQL) <sup>[2]</sup> which is an SQL-like language for building tables aggregating information from external web sources (and in particular RSS feeds). To our knowledge, pipes are relational expressions querying a database storing the items of each feed. Whereas YQL is more expressive than our

<sup>3</sup> Google reader is available at <http://www.google.com/reader>.

algebra, all queries are evaluated independently on demand which excludes MQO techniques as proposed by our solution.

*Data stream processing:* RoSeS is to our knowledge the first system based on continuous query processing for aggregating feeds. There is a large amount of previous work on processing data streams and providing new continuous algebras and query languages [9][13][21][23][3][29]. Most of these languages are based on a *snapshot semantics* (the result of a continuous query at some time instant  $t$  correspond to the result of a traditional one-shot query on a snapshot of its argument) and redesign relational operators with a pipe-lining execution model using inter-operator queues. They also introduce different kinds of time-based and count-based window operators (sliding, tumbling, landmark) for computing aggregates and joins [13]. The semantics of our data stream model and algebra is strongly influenced by this work. The main originality of our algebra with respect to existing algebra concerns our definition of annotation join which facilitates the rewriting and optimization of the algebraic query plans (see section 4.2).

Since RSS and Atom are encoded with XML, we also explored existing approaches for querying XML streams. XML streaming systems are concerned with the continuous evaluation of XPath [14][25] and XQuery [19][4] expressions. The rich semi-structured semantics of XML increases the complexity of the underlying query languages and their implementation. RSS feeds are simple streams of flat XML fragments which do not need highly expressive XML path expressions and we decided to follow a simple attribute/value approach which is more efficient and sufficiently expressive for encoding RSS data.

*Multi-query optimization:* Publications are stored continuous queries which might be similar in the sense that they share filtering conditions on the same input streams. This naturally calls for the application of efficient multi-query optimization techniques. Multi-query optimization (MQO) has first been studied in the context of DBMS where different users could request complex (one-shot) queries simultaneously to the system [27]. Most MQO techniques exploit the fact that multiple queries can be evaluated more efficiently together than independently, because it is often possible to share state and computation [11]. Solutions using this observations are based on predicate indexing [28], sharing states in global NFA [11]), join graphs [15] and sub-query factorization [9][3][7]. We followed the latter approach which appeared to be the most promising for a cost-based multi-query optimization solution.

Publish/subscribe systems aim to solve the problem of filtering an incoming stream of items (generally generated by a large collection of source streams) according to topic-based or content-based subscriptions. Different techniques propose efficient topic-clustering algorithms [22][24], appropriate subscription index structures [20][12][6], and distributed stream processing [26][18]. These techniques mainly focus on the parallel processing and efficient indexing of conjunctive keyword queries, which are less expressive than our aggregation queries. However, they share some issues and solutions which are conceptually similar to the MQO problem mentioned before.

### 3 RoSeS Architecture

The RoSeS system is composed of five modules for processing RSS feeds and managing meta-data about users, publications and subscriptions. As shown in Figure 1, RSS feeds are processed by a three layered architecture where the top layer (acquisition) is in charge of crawling the collection of registered RSS feeds (in the following called source feeds), the second layer (evaluation) is responsible for processing a continuous *query plan* which comprises all publication queries and the third layer (diffusion) deals with publishing the results according to the registered subscriptions (see Section 4). The remaining two modules (catalog and system manager) provide meta-data management services for storing, adding, updating and deleting source feeds, publication queries and subscriptions.

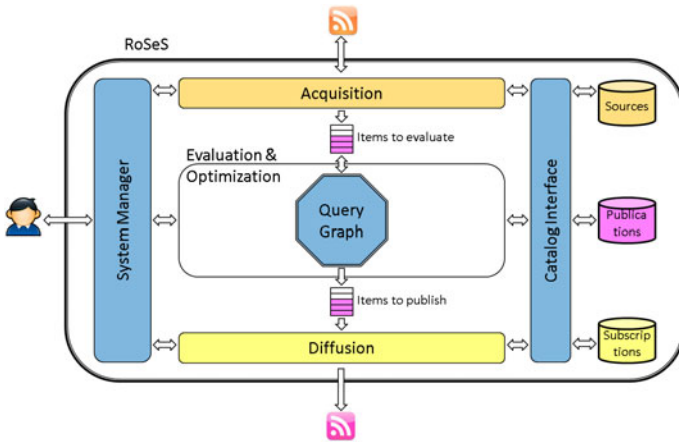


Fig. 1. RoSeS System architecture

*Acquisition:* The main task of this module is to transform evolving RSS documents into a continuous stream of RoSeS items which can be processed by the evaluation module. This transformation includes an efficient refresh strategy optimizing the bandwidth usage. In [16], we propose a best-effort strategy for refreshing RSS documents under limited bandwidth, which introduces the notion of saturation for reducing information loss below a certain bandwidth threshold. The freshness efficiency is not in the scope of this paper.

*Evaluation:* The core of the RoSeS system is an *algebraic multi-query plan* which encodes all registered publication queries. The evaluation of this query plan follows an asynchronous pipe-lined execution model where the evaluation module (1) continuously evaluates the set of algebraic operations according to the incoming stream of RoSeS items, and (2) manages the addition, modification and deletion of publication queries.



*Diffusion*: This module is responsible for transforming RoSeS items into different output formats and notifying new items to corresponding user subscriptions. The goal of this module is to define the way items are rewritten with annotations, in given formats (SMS, email, RSS/Atom feed...).

## 4 RoSeS Data Model and Language

### 4.1 The RoSeS Language

The language we have implemented in the RoSeS system provides instructions for *registering new source feeds* (register), *defining new publications* (publish) and *creating subscriptions* (subscribe). We will shortly describe these language components completed by an example. The component we are most interested in this article are publication queries, which will be studied in more detail in section [4.2](#).

The *publication language* has been designed to respond to several desiderata: to be expressive but simple to use, to facilitate most common operations and to be appropriate for large scale multi-query optimization. In RSS syndication, the most commonly used aggregation is based on large unions combined with filtering. RoSeS enforces the simplicity in use of its declarative publication language, by favoring the expression of large unions, combined with targeted filtering and joins.

A publication query contains three clauses:

- A mandatory **from** clause, which specifies the input feeds that produce output items, called *main feeds*.
- Zero, one or several **join** clauses, each one specifying a join with a *secondary feed*. Secondary feeds only produce annotations (no output) to main feed elements.
- An optional **where** clause for filtering conditions on main or secondary feeds.

The *registering language* allows defining new source feeds coming either from external RSS/Atom feeds, or from internal materialized publications. Note that RoSeS does not allow item transformation in publications (virtual feeds). However, transformations are possible if the resulting feed is materialized and registered as a new source feed at the acquisition level. Transformations are expressed through XSLT stylesheets that transform a RoSeS item structure into another. Transformations may use annotations produced by joins (e.g., include corresponding links to photos of feed *MusePhotoStream* at materialization time, in Example 2 below).

The *subscription language* allows defining subscriptions to existing publication/-source feeds. A subscription specifies a feed, a notification mode (RSS, mail, etc.), a periodicity and possibly a transformation. Subscription transformations are expressed by XSLT stylesheets, but unlike registering transformations, the output format is free.

*Example 1.* Suppose Bob regularly organizes with his friends outings to rock concerts. He therefore defines a publication *RockConcertStream*, including items about concerts from his friends messages (feeds *FriendsFacebookStream* and *FollowedTwitterStream*), and rock concert announces from feed *EventAnnounces*.

For this, he first registers the corresponding source streams in the system and creates a new publication *RockConcertStream*:

```

register feed http://www.facebook.com/feeds/friends_notes.php?id=x&key=y&format=rss20
  as FriendsFacebookStream;
register feed http://www.infoconcert.com/rss/news.xml as EventAnnounces;
register feed http://twitter.com/statuses/user_timeline/174451720.rss FollowedTwitterStream;
create feed RockConcertStream
  from (FriendsFacebookStream | EventAnnounces as $ca | FollowedTwitterStream) as $r
  where $ca[title contains 'rock'] and $r[description contains 'concert'];

```

Notice that the **from** clause allows defining groups (unions) of feeds, identified by variables and used then to express conditions on the group in the **where** clause. Here filtering conditions are expressed on *EventAnnounces* (title contains 'rock') and on grouped feeds (description contains 'concert').

*Example 2.* Then Bob, who is a fan of the Muse rock group, creates feed *MusePhotoStream* with items about Muse, annotated with photos. Items come from feeds *RockConcertStream* (those talking about Muse) and *MuseNews*, while photos come from two secondary feeds: *FriendsPhotos* with photos published by his friends and *MusicPhotos* (only for category 'rock'). Annotation is realized by join, each item from the main feed (\$main) is annotated with photos in the last 3 months from secondary feed items having similar titles. Notice that a join specifies a window on a group of secondary feeds, a main feed (through a variable) and a join predicate.

```

register feed http://muse.mu/rss/news.rss as MuseNews;
create feed MusePhotoStream
  from (RockConcertStream as $r | MuseNews) as $main
  join last 3 months on (MusicPhotos as $m | FriendsPhotos)
    with $main[title similar window.title]
  where $r[description contains 'Muse'] and $m[category = 'rock'];
register feed MusePhotoStream apply 'IncludePhotos.xml' as MuseWithPhotos;

```

The final example shows two subscriptions to the *RockConcertStream* publication: the first one extracts item titles ('Title.xml' transformation) and sends them by mail every 3 hours, the second one simply outputs an RSS feed refreshed every 10 minutes.

```

subscribe to RockConcertStream apply 'Title.xml' output mail 'me@mail.org' every 3 hours;
subscribe to RockConcertStream output file 'RockConcertStream.rss' every 10 minutes;

```

## 4.2 Data Model and Algebra

The RoSeS data model borrows from state-of-the-art data stream models, while proposing specific modeling choices adapted to RSS/Atom syndication and aggregation.

A *RoSeS feed* corresponds to either a registered external RSS/Atom (source) feed, or to a publication (virtual) feed. A feed is a couple  $f = (d, s)$ , where  $d$  is the feed description and  $s$  is a RoSeS stream. Description is a tuple, representing usual RSS/Atom feed properties: title, description, URL, etc.

A *RoSeS stream* is a data stream of annotated RoSeS items. More precisely, a RoSeS stream is a (possibly infinite) set of elements  $e = (t, i, a)$ , where  $t$  is a timestamp,  $i$  a RoSeS item, and  $a$  an annotation, the set of elements for a given timestamp being finite. Annotation links joined items to an element. An annotation is a set of couples  $(j, A)$ , where  $j$  is a join identifier and  $A$  is a set of items - the annotation is further detailed in the join operator below.

*RoSeS items* represent data content conveyed by RSS/Atom items. Despite the adoption of an XML syntax, RSS and Atom express rather flat text-oriented content structure. Extensions and deeper XML structures are very rarely used, therefore we made the choice of a flat structure, as a set of typed attribute-value couples, including common RSS/Atom properties: title, description, link, author, publication date, etc. Extensibility may be handled by including new, specific attributes to RoSeS items - this enables both querying any feed through the common attributes and addressing specific attributes (when known) of extended feeds.

A *RoSeS window* expresses subsets of a stream's items valid at various moments. More precisely, a window  $w$  on a stream  $s$  is a set of couples  $(t, I)$ , where  $t$  is a timestamp and  $I$  is the set of items of  $s$  valid at  $t$ . Note that (i) a timestamp may occur only once in  $w$ , and (ii)  $I$  contains only items that occur in  $s$  before (or at) timestamp  $t$ . We note  $w(t)$  the set of items in  $w$  for timestamp  $t$ . Windows are used in RoSeS only for computing joins between streams. RoSeS uses sliding windows of two types: time-based (last  $n$  units of time) and count-based (last  $n$  items).

Publication definition is based on *five main operators* for composing RoSeS streams. We distinguish *conservative operators* (filtering, union, windowing, join), that do not change the content of the input items, i.e. they output only already existing items, from *altering operators* (transformation), that produce new items.

Subscribed publications are translated into algebraic expressions as shown in the following example for *RockConcertStream* and *MusePhotoStream* (for space reasons abbreviated feed names and a simplified syntax are used).

$$\begin{aligned} RConcert &= \sigma_{concert' \in desc}(F\text{Facebook} \cup \sigma_{rock' \in title}(E\text{Announces}) \cup FT\text{twitter}) \\ MPhoto &= (\sigma_{Muse' \in desc}(RConcert) \cup M\text{News}) \bowtie_{title \sim w.title} \\ &\quad \omega_{last\ 3\ m}(\sigma_{cat=rock'}(M\text{Photos}) \cup F\text{Photos}) \end{aligned}$$

The algebra is defined in the rest of this section and the evaluation of algebraic expressions is detailed in section 5.

A central design choice for the RoSeS language is to express *only conservative operators in the publication language*. The advantage is that conservative operators have good query rewriting properties (commutativity, distributivity, etc.), which favor both query optimization and language declarativeness (any algebraic expression can be rewritten in a normalized form corresponding to the declarative clauses of the language). Expressiveness is preserved first by allowing join, the most powerful operator, filtering, union and windowing in the publication language. Next, transformation can be used in defining new materialized source feeds. Notice that transformations may use join annotations, and consequently improve (indirectly) the expressive power of joins.

*Filtering* outputs only the stream elements that satisfy a given item predicate, i.e.  $\sigma_P(s) = \{(t, i, a) \in s \mid P(i)\}$ . *Item predicates* are boolean expressions (using

conjunctions, disjunctions, negation) of atomic item predicates that express a condition on an item attribute; depending on the attribute type, atomic predicates may be:

- for simple types: comparison with value (equality, inequality).
- for date/time: comparison with date/time values (or year, month, day, etc.).
- for text: operators *contains* (word(s) contained into a text attribute), *similar* (text similar to another text).
- for links: operators *references/extends* (link references/extends an URL or host), *shareslink* (attribute contains a link to one of the URLs in a list).

Note that RoSeS allows applying text and link predicates *to the whole item* - in this case the predicate considers the whole text or all the links in the item’s attributes.

*Union* outputs all the elements in the input streams, i.e.  $\bigcup(s_1, \dots, s_n) = s_1 \cup \dots \cup s_n$ . Union may be expressed explicitly, by enumerating input streams, or implicitly, through a query over the existing feeds.

*Windowing* produces a window on the input stream conforming to the window specification, i.e.  $\omega_{t,spec}(s)$  and  $\omega_{c,spec}(s)$  define a time-based, respectively a count-based sliding window, where *spec* expresses a duration, respectively a number of items.

*Join* takes a (main) stream and a window on a (secondary) stream. RoSeS uses a conservative variant of the join operation, called *annotation join*, that acts like a semi-join (main stream filtering based on the window contents), but keeps a trace of the joining items by adding an annotation entry. A join  $\bowtie_P(s, w)$  of identifier *j* outputs the elements of *s* for which the join predicate *P* is satisfied by a non-empty set *I* of items in the window, and adds to them the annotation entry (*j*, *I*). More precisely,  $\bowtie_P(s, w) = \{(t, i, a') \mid (t, i, a) \in s, I = \{i' \in w(t) \mid P(i, i')\}, |I| > 0, a' = a \cup \{(j, I)\}\}$ .

*Transformation* modifies each input element following a given transformation function, i.e.  $\tau_T(s) = \{T(t, i, a) \mid (t, i, a) \in s\}$ . It is the only altering operator, whose use is limited to produce subscription results or new source feeds, as explained above.

## 5 Query Processing

### 5.1 Query Graphs

Query processing consists in continuously evaluating a collection of publication queries. This collection is presented by a *multi-query plan* composed of different physical operators reflecting the algebraic operators presented in Section 4.2 (union, selection, join and window). Query execution is based on a pipe-lined execution model where a query plan is transformed into a graph connecting sources, operators and publications by inter-operator queues or by window buffers (for blocking operators like join). A query plan for a set of queries *Q* can then be represented as a directed acyclic graph  $G(Q)$  as shown in Figure 2. Graph in Figure 2 represents one possible physical query plan for the following set of publications:  $p_1 = \sigma_1(s_1 \cup s_2)$ ,  $p_2 = (s_3 \cup s_4) \bowtie_1 \omega_1(s_5)$ ,  $p_3 = \sigma_2(p_1 \cup s_6)$ . As we can see, window operators produce a different kind of output, window buffers, which are consumed by join operators. View composition is illustrated by an arc connecting a publication operator to an algebraic operator ( $p_1$  is connected to union  $\cup_3$ ). Observe also that transform operators are applied after publication operators.

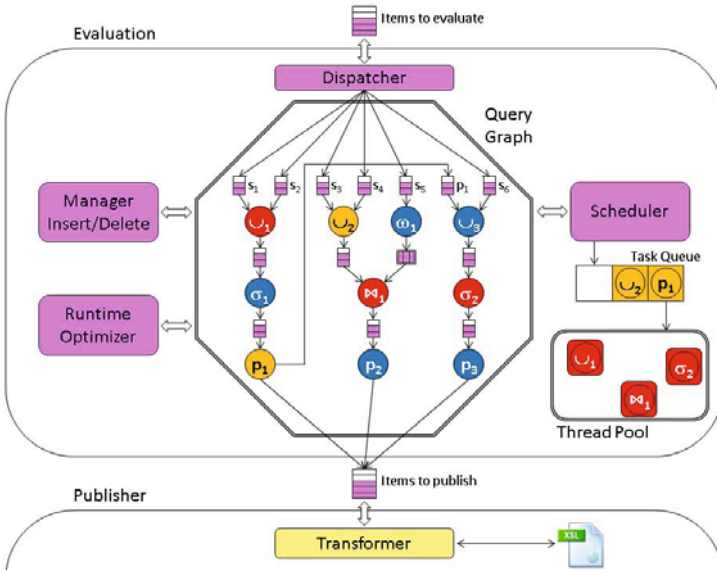


Fig. 2. Evaluation module architecture

The next section presents query graph processing and the underlying cost-model that is implemented in the current system. Section 5.3 introduces our optimization techniques (not yet implemented) that improve processing performances.

### 5.2 Query Evaluation and Cost-Model

As explained in the previous section, a set of publication queries is translated into a multi-query plan composed of operators connected by read/write queues or by window buffers. New items are continuously arriving to this graph and have to be consumed by the different operators. We have adopted a multi-threaded pipe-lining execution model which is a standard approach in continuous query processing architectures. Query execution is done as follows. The query graph is observed by a scheduler that continuously decides which operators (tasks) must be executed (see Figure 2). The scheduler has at his disposal a pool of threads for executing in parallel a fixed number of threads (the naive solution of attaching one thread to each operator rapidly becomes inefficient / impossible due to thread management overhead or system limitations). The choice of an inactive operator to be evaluated is influenced by different factors depending on the input buffer of each operator (the number and/or age of the items in the input queue).

Based on this execution model, we define a cost model for estimating the resources (memory, processor) necessary for the execution of a query plan. Compared to the cost estimation of a traditional query plan which is based on the size of the input data, the estimation parameters of a continuous plan must reflect the streaming nature of the data. We adapt a simplified version of the model presented in [5] and define the cost of each operator  $op$  as a function of the publishing rate  $R(b)$  of its input buffer(s)  $b$  (and the size  $S(w)$  of input windows  $w$ , for join operators).

**Table 1.** Cost model

Operator	Output rate	Memory	Processing cost
$\sigma_p(b)$	$sel(p) * R(b)$	$const$	$const * R(b)$
$\cup(b_1, \dots, b_n)$	$\sum_{1 \leq i \leq n} R(b_i)$	0	0
$\bowtie_p(b, w)$	$sel(p) * R(b)$	$const$	$R(b) * S(w)$
$\omega_d(b)$	0	$S = const$ or $S = d * R(b)$	$const$

As we can see in table 1, the cost of each operator strongly depends on the publishing rate of its input buffer(s). The selection operator assumes a constant execution cost for each item (a more precise model could take account of the size of each item, but since items are in general small text fragments we ignore this detail in our model). Memory cost of selection is also constant (the size of the operator plus the size of one item). The publishing rate of the selection operator reduces the rate of its input buffer by the selectivity factor  $sel(p) \in [0, 1]$  depending on the selection predicate  $p$ . Union generates an output buffer with a publishing rate corresponding to the sum of its input buffer rates. We assume zero memory and processing cost since each union can be implemented by a set of buffer iterators, one for each input buffer. The join operator generates an output buffer with a publishing rate of  $sel(p) * R(b)$  where  $R(b)$  is the publishing rate of the primary input stream and  $sel(p) \in [0, 1]$  corresponds to the probability that an item in  $b$  joins with an item in window  $s$  using join predicate  $p$ . This is due to the behavior of the annotation join: one item is produced by the join operator when a new item arrives on the main stream and matches at least one item in the window. Then the item is annotated with all matching window items. So the processing cost of the join operator corresponds to the product  $R(b) * S(w)$ , where  $S(w)$  is the size of the join window  $w$ . The window operator transforms the stream into a window buffer where the size depends on the size of the window (count-based window) or on the time-interval  $d$  and the input buffer rate  $R(b)$  (time-based window). It is easy to see that the input buffer rate of each operator strongly influences the global cost of the execution plan (the sum of the cost of all operators) and we will describe in the following section how we can reduce this rate by pushing selections and joins towards the source feeds of the query plan.

### 5.3 Query Graph Optimization

An important originality of our framework with respect to other multi-query optimization solutions lies in the explicit integration of a cost model. This makes it more expressive than other approaches without cost model. As mentioned before, our optimization strategy is based on the heuristic that selections and joins should be applied as early as possible in order to reduce the global cost of the plan. We use traditional rewriting rule for algebraic expressions (distributivity of selection over union, commutativity of selection with join and transforming selections into a cascade of selections). Observe that commutativity with join is possible because of the particular nature of our annotation joins which do not modify the input items and guarantee that all subsequent selections only apply to these items. We will describe the whole process in the following.

The optimization process can be decomposed into two phases: (a) a *normalization* phase which applies all rewriting rules for pushing selections towards their source feeds and for distributing joins over union and (b) a *factorization* phase of the selection predicates of each source based on a new cost-based factorization technique.

*Query normalization:* The goal of the first phase is to obtain a normalized query plan where all filtering selections are applied first to each source before applying joins and unions. This is possible by iteratively applying distributivity of selections on unions and commutativity between selection and join. It is easy to show that we can obtain an equivalent plan which is a four level tree where the leaves of the tree are the sources involved in the query (first level), the second level nodes are the selection operators which can be applied to each source (leaf), the third level are window/join operators operators applied to the results of the selections and the final (fourth) level are unions applied to the results of the selections/windowed-joins to build the final results. Normalization also flattens all cascading selection paths into a single conjunctive selection. This modification might increase the cost of the resulting normalized graph with respect to the original graph. However, as we will show at the end of this section this increase is only temporary and compensated by the following factorization phase.

We will explain our approach by a simple example without join operations. Suppose the three publication queries shown in Figure 3, publication  $p_3$  ( $p_3 = \sigma_d(p_2 \cup s_5)$ ) is defined over another publication  $p_2$ , with a filtering operation ( $\sigma_{a \wedge c}$ ). Here, the normalization process consists in pushing all the selection operations through the publication tree to the sources. This lets us obtain the normal form shown in Figure 4:

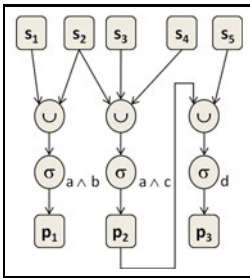
$$p_3 = \sigma_{a \wedge c \wedge d}(s_2) \cup \sigma_{a \wedge c \wedge d}(s_3) \cup \sigma_{a \wedge c \wedge d}(s_4) \cup \sigma_d(s_5)$$


Fig. 3. Query graph

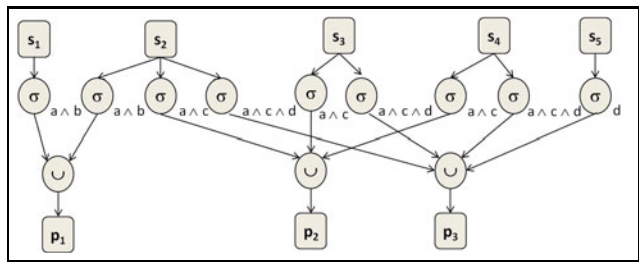


Fig. 4. Normalized query graph

*Query factorization:* Normalization generates a global query plan where each source  $s$  is connected to a set of selection predicates  $P(s)$ . Factorization consists in building a minimal *filtering plan* for each source. To find a best operator factorization, we proceed in two steps: we first generate for each source a *predicate subsumption graph* which contains *all* predicates subsuming the set of predicates in  $P(s)$ . Each subsumption link in this graph is labeled by a weight corresponding to the output rate of the source node (source or filtering operation). It is easy to show that any sub-tree of this graph covering the source  $s$  (root) and all predicates in  $P(s)$  corresponds to

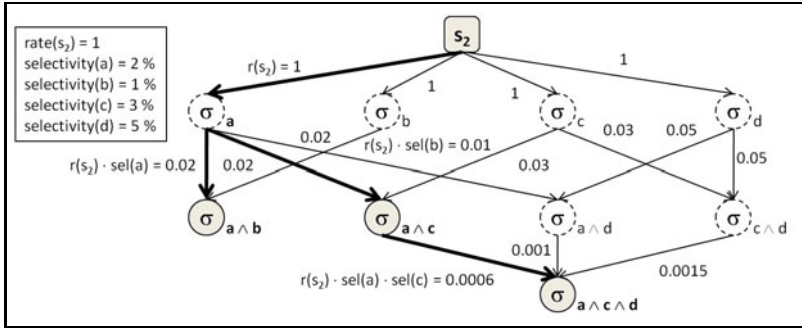


Fig. 5. Subsumption Graph for  $s_2$  and Steiner Tree

a filtering plan which is equivalent to the original plan where the cost is obtained by the sum of all arc weights in the tree. Based on this observation we then try to find a Steiner tree [8], which corresponds to a sub-tree of minimal cost covering all initial predicates. The idea of this procedure is illustrated in Figure 5 showing a subsumption graph for the source  $s_2$  and a minimal Steiner tree of this graph (in bold). The selection operators corresponding to  $s_2$  are (fig. 4)  $\sigma_{a \wedge b}$ ,  $\sigma_{a \wedge c}$ ,  $\sigma_{a \wedge c \wedge d}$ . The subsumption graph is composed of these three operators, the operators corresponding to the sub-expressions of the three initial operators:  $\sigma_a$ ,  $\sigma_b$ ,  $\sigma_c$ ,  $\sigma_d$ ,  $\sigma_{a \wedge d}$  and  $\sigma_{c \wedge d}$ , and the subsumption arcs between all these operators ( $\sigma_a \rightarrow \sigma_{a \wedge b}$ ,  $\sigma_b \rightarrow \sigma_{a \wedge b}$ , ...). Subsumption graph arcs are populated with weights, which represent the estimated evaluation cost of using that arc. They correspond to the product between the involved source’s publishing rate and the estimated selectivity of the operator predicate at the tail of the arc (e.g.,  $cost(\sigma_a \rightarrow \sigma_{a \wedge b}) = rate(s_2) \cdot selectivity(a)$ ). Then a Steiner tree algorithm can be run over the subsumption graph to find the best factorization tree. In our example with selection operators for source  $s_2$ , the resulting tree can be seen in Figure 5.

The Steiner tree problem [8] is known to be NP-complete and we have developed an approximate algorithm which exploits the particular properties of a filtering plan (all outgoing arcs of a node have the same weight and the weight is monotonically decreasing with the depth of the node in the graph). This algorithm is based on a local heuristic for dynamically building the subsumption graph by choosing the most selective subsuming predicates. Whereas it will find only an approximate solution, it can incrementally add new predicates generated by new publication queries. A detailed description of the whole process is outside the scope of this paper.

It is easy to show that the final filtering plan has less cost than the initial filtering plan. Normalization can increase the cost of the filtering plan by replacing cascading selection paths by a conjunction of all predicates on the path. However, it can be shown that the subsumption graph regenerates all these paths and the original plan is a sub-tree of this graph. Since the Steiner tree is a minimal sub-tree for evaluating the initial set of predicates, its cost will be at most be the cost of the initial graph. For example, the cost for source  $s_2$  in the original plan (Figure 3) roughly is twice the publishing rate of



$s_2$  (both selections  $\sigma_{a \wedge b}$  and  $\sigma_{a \wedge c}$  are applied to all items generated by  $s_2$ ). This cost is reduced to its half in the final Steiner tree by introducing the additional filter  $a$ .

## 6 Conclusion and Future Work

In this paper we have presented RoSeS, a large-scale RSS feed aggregation system based on a continuous multi-query processing and optimization. Our main contributions are a simple but expressive aggregation language and algebra for RSS feeds combined with an efficient cost-based multi-query optimization technique. The whole RoSeS architecture, feed aggregation language and continuous query algebra have been implemented [10] and first experiments show that the system is able to manage thousands of publications within reasonable system resources. We are currently extending this system by the multi-query optimization strategy described in Section 5.3 and some preliminary experiments show that the optimization phase scales well with respect to the number of filters. A deeper discussion concerning scalability is outside the scope of this paper.

The most important issue we are addressing now concerns two intrinsic dimensions of dynamicity in continuous query processing systems like RoSeS. Users who continuously modify the query plan by adding, deleting and updating publication queries introduce the first dimension of dynamicity. The second one is brought by sources, which generally have a time-varying publishing behavior in terms of publishing rate and contents. Both dimensions strongly influence the evaluation cost of a query plan and need a continuous re-optimization strategy in order to compensate performance loss.

A standard approach [31,30] to this problem consists in periodically replacing a query plan  $P$  by a new optimized plan  $O$  (query plan migration). The problem here is to estimate this difference efficiently without rebuilding the complete optimal plan. We are currently studying such a cost-based threshold re-optimization approach adapted to our context. The basic idea is to keep the subsumption graphs generated for each source during optimization. These subsumption graphs can be maintained at run-time by adding and deleting source predicates and updating the statistics concerning each source (selectivity, publishing rate). Using an appropriate threshold measure, it is then possible to recompute the optimal Steiner trees and update the corresponding query plan fragments.

## References

1. The yahoo! pipes feed aggregator, <http://pipes.yahoo.com>
2. Yahoo! query language, <http://developer.yahoo.com/yql>
3. Arasu, A., Babu, S., Widom, J.: The CQL continuous Query Language: Semantic Foundations and Query Execution. In: VLDB, pp. 121–142 (2006)
4. Botan, I., Fischer, P., Florescu, D., Kossman, D., Kraska, T., Tamosevicius, R.: Extending XQuery with Window Functions. In: VLDB, pp. 75–86 (2007)
5. Cammert, M., Krmer, J., Seeger, B., Vaupel, S.: A cost-based approach to adaptive resource management in data stream systems. In: TKDE, vol. 20, pp. 230–245 (2008)
6. Chandramouli, B., Phillips, J.M., Yang, J.: Value-based notification conditions in large-scale publish/subscribe systems. In: VLDB, pp. 878–889 (2007)

7. Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M., Hellerstein, J., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., Shah, M.A.: TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In: CIDR (2003)
8. Charikar, M., Chekuri, C., Cheung, T., Dai, Z., Goel, A., Guha, S., Li, M.: Approximation algorithms for directed steiner problems. In: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms SODA 1998, pp. 192–200. Society for Industrial and Applied Mathematics (1998)
9. Chen, J., DeWitt, D.J., Tian, F., Wang, Y.: NiagaraCQ: A Scalable Continuous Query System for Internet Databases. In: SIGMOD Record, pp. 379–390 (2000)
10. Creus, J., Amann, B., Travers, N., Vodislav, D.: Un agrégateur de flux rss avancé. In: 26<sup>e</sup> Journées Bases de Données Avancées, demonstration (2010)
11. Demers, A., Gehrke, J., Hong, M., Riedewald, M., White, W.: Towards expressive publish-/Subscribe systems. 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. 627–644. Springer, Heidelberg (2006)
12. Fabret, F., Jacobsen, A., Llirbat, F., Pereira, J., Ross, K., Shasha, D.: Filtering algorithms and implementation for very fast publish/subscribe systems. In: SIGMOD Record, pp. 115–126 (2001)
13. Golab, L., Özsu, M.T.: Issues in Data Stream Management. SIGMOD Record 32(2), 5–14 (2003)
14. Gupta, A.K., Suci, D.: Stream Processing of XPath Queries with Predicates. In: SIGMOD Record, pp. 419–430 (2003)
15. Hong, M., Demers, A.J., Gehrke, J., Koch, C., Riedewald, M., White, W.M.: Massively Multi-Query Join Processing in Publish/Subscribe Systems. In: SIGMOD Record, pp. 761–772 (2007)
16. Horincar, R., Amann, B., Artières, T.: Best-effort refresh strategies for content-based RSS feed aggregation. In: Chen, L., Triantafyllou, P., Suel, T. (eds.) WISE 2010. LNCS, vol. 6488, pp. 262–270. Springer, Heidelberg (2010)
17. I.E.T.F. IETF. Atompub status pages. URL retrieved (2011), -02-12., <http://tools.ietf.org/wg/atompub>
18. Jun, S., Ahamad, M.: FeedEx: Collaborative Exchange of News Feeds. In: WWW, pp. 113–122 (2006)
19. Koch, C., Scherzinger, S., Schweikardt, N., Stegmaier, B.: FluXQuery: An Optimizing XQuery Processor for Streaming XML Data. In: VLDB (2004)
20. König, A.C., Church, K.W., Markov, M.: A Data Structure for Sponsored Search. In: ICDE, pp. 90–101 (2009)
21. Li, J., Maier, D., Tufte, K., Papadimos, V., Tucker, P.A.: Semantics and Evaluation Techniques for Window Aggregates in Data Streams. In: SIGMOD Record, pp. 311–322 (2005)
22. Li, X., Yan, J., Deng, Z., Ji, L., Fan, W., Zhang, B., Chen, Z.: A Novel Clustering-Based RSS Aggregator. In: WWW, pp. 1309–1310 (2007)
23. Luo, C., Thakkar, H., Wang, H., Zaniolo, C.: A Native Extension of SQL for Mining Data Streams. In: SIGMOD Record, pp. 873–875 (2005)
24. Milo, T., Zur, T., Verbin, E.: Boosting topic-based publish-subscribe systems with dynamic clustering. In: SIGMOD Record, pp. 749–760 (2007)
25. Peng, F., Chawathe, S.: XPath Queries on Streaming Data. In: SIGMOD Record, pp. 431–442 (2003)
26. Rose, I., Murty, R., Pietzuch, P.R., Ledlie, J., Roussopoulos, M., Welsh, M.: Cobra: Content-based filtering and aggregation of blogs and rss feeds. In: NSDI (2007)
27. Sellis, T.K.: Multiple-query optimization. ACM Trans. Database Syst. 13, 23–52 (1988)
28. Whang, S.E., Garcia-Molina, H., Brower, C., Shanmugasundaram, J., Vassilvitskii, S., Vee, E., Yerneni, R.: Indexing boolean expressions. VLDB Endow 2, 37–48 (2009)

29. Wu, E., Diao, Y., Rizvi, S.: High-Performance Complex Event Processing over Streams. In: SIGMOD Record, pp. 407–418 (2006)
30. Yang, Y., Krämer, J., Papadias, D., Seeger, B.: Hybmig: A hybrid approach to dynamic plan migration for continuous queries. TKDE 19(3), 398–411 (2007)
31. Zhou, Y., Salehi, A., Aberer, K.: Scalable delivery of stream query result. VLDB Endow 2, 49–60 (2009)

# The Linear Combination Data Fusion Method in Information Retrieval

Shengli Wu<sup>1,2</sup>, Yaxin Bi<sup>2</sup>, and Xiaoqin Zeng<sup>3</sup>

<sup>1</sup> School of Computer Science and Telecommunication Engineering  
Jiangsu University, Zhenjiang, China

<sup>2</sup> School of Computing and Mathematics  
University of Ulster, Northern Ireland, UK

<sup>3</sup> College of Computer and Information Engineering  
Hehai University, Nanjing, China

**Abstract.** In information retrieval, data fusion has been investigated by many researchers. Previous investigation and experimentation demonstrate that the linear combination method is an effective data fusion method for combining multiple information retrieval results. One advantage is its flexibility since different weights can be assigned to different component systems so as to obtain better fusion results. However, how to obtain suitable weights for all the component retrieval systems is still an open problem.

In this paper, we use the multiple linear regression technique to obtain optimum weights for all involved component systems. Optimum is in the least squares sense that minimize the difference between the estimated scores of all documents by linear combination and the judged scores of those documents. Our experiments with four groups of runs submitted to TREC show that the linear combination method with such weights steadily outperforms the best component system and other major data fusion methods such as CombSum, CombMNZ, and the linear combination method with performance level/performance square weighting schemas by large margins.

## 1 Introduction

In information retrieval, the data fusion technique has been investigated by many researchers and quite a few data fusion methods such as CombSum [78], CombMNZ [78], the linear combination method [219,20,21,23], the multiple criteria approach [6], the correlation method [26,27], Borda count [1], Condorcet fusion [14], Marcov chain-based methods [4,16], and others [5,9,11,17,28], have been presented and investigated. Previous research demonstrates that, generally speaking, data fusion is an effective technique for obtaining better results.

Data fusion methods in information retrieval can be divided into two categories: score-based methods and rank-based methods. These two different types of methods apply to different situations. If some or all component systems only provide a ranked list of documents as the result of a query, then rank-based methods can be applied. If every component system provides scores for

all retrieved documents, then score-based methods can be applied. In all these methods, CombSum [7,8], CombMNZ [7,8], Borda count [1] <sup>1</sup>, the correlation method [26,27], the linear combination method [2,19,20,21,23] are score-based methods, while Condorcet fusion [14], Marcov chain-based methods [4,16], and the multiple criteria approach [6] are rank-based methods.

Among all those score-based data fusion methods, the linear combination method is very flexible since different weights can be used for different component systems. It is especially beneficial when component systems involved vary considerably in performance and dissimilarity between each other. However, how to assign suitable weights to the component systems involved is a crucial issue that needs to be carefully investigated.

Some previous investigation (e.g., in [11,18]) took a simple performance level policy. That is, for every component system, its average performance (e.g., average precision) is measured using a group of training queries, then the value obtained is used as the weight for that component system correspondingly. Some alternatives to the simple performance level weighting schema was investigated in [23]. On the other hand, both system performance and dissimilarities between systems were considered in [27] to determine all systems' weights. Although these weighting schemas can usually bring performance improvement over CombSum and CombMNZ – the two commonly used data fusion methods that assign equal weights to all component systems, it is very likely that there is still room for further improvement since all these weighting schemas are heuristic in nature.

In this paper we investigate a new approach, multiple linear regression, to training system weights for data fusion. Our experiments demonstrate that this approach is more effective than other approaches. To our knowledge, multiple linear regression has not been used for such a purpose before, though it has been used in various other tasks in information retrieval.

A related issue is how to obtain reliable scores for those retrieved documents. Sometimes scores are provided by component retrieval systems, but usually those raw scores from different retrieval systems are not comparable and some kind of score normalization is needed before fusing them. An alternative is to convert ranking information into scores if raw scores are not available or not reliable. Different models [1,3,12,15,22] have been investigated for such a purpose. In this piece of work, we use the logistic regression model for it. This technique has been investigated for distributed information retrieval [3,15], but not for data fusion before.

The rest of this paper is organized as follows: in Section 2 we review some related work on data fusion. Then in Section 3 we discuss the linear combination method for data fusion and especially the two major issues involved. Section 4 presents the experiment settings and results for evaluating a group of data fusion methods including the one presented in this paper. Section 5 is the conclusion.

---

<sup>1</sup> Borda count is a little special. It does not require that scores are provided by component systems for retrieved documents. It converts ranking information into scores instead. Since it uses the same combination algorithm as CombSum (or CombMNZ) for fusion, we classify it as a score-based method.

## 2 Related Work

In this section we review some previous work on the linear combination method and score normalization methods for data fusion in information retrieval.

Probably it was Thompson who reported the earliest work on the linear combination data fusion method in [18]. The linear combination method was used to fuse results submitted to TREC 1. It was found that the combined results, weighted by performance level, performed no better than a combination using uniform weights (CombSum). Weighted Borda count [1] is a variation of performance level weighting, in which documents at every rank are assigned corresponding scores using Borda count, and then the linear combination method with the performance level weighting is used for fusion. Another variation is, every system's performance can be estimated automatically without any training data, then the linear combination method can be applied [24].

In a piece of work done by Wu et. al. [23], their experiment shows that, using the power function of performance with a power value between 2 and 6 can lead to slightly better fusion results than performance level weighting. On the other hand, in Wu and McClean's work [26,27], both system performance and dissimilarities among systems were considered. In their weighting schema, any system's weight is a compound weight that is the product of its performance weight and its dissimilarity weight. The combination of performance weights and dissimilarity weights was justified using statistical principles and sampling theories by Wu in [21].

Bartell and Cottrell [2] used a numerical optimisation method, conjugate gradient, to search for good weights for component systems. Because the method is time-consuming, only 2 to 3 component systems and the top 15 documents returned from each system for a given query were considered in their investigation.

Vogt and Cottrell [19,20] analysed the performance of the linear combination method. In their experiments, they used all possible pairs of 61 systems submitted to the TREC 5 ad-hoc track. Another numerical optimisation method, golden section search, was used to search for good system weights. Due to the nature of the golden section search, only two component systems can be considered for each fusion process. Generally speaking, these brute-force search methods are very time-consuming and their usability in many applications are very limited.

An associated problem is how to obtain reliable scores for retrieved documents. This problem is not trivial by any means. Sometimes component information retrieval systems provide scores for their retrieved documents. However, those raw scores from different retrieval systems may not be comparable. Some kind of normalization is required for those raw scores. One common linear score normalization method is: for any list of scores (associated with a ranked list of documents) for a given topic or query, we map the highest score into 1, the lowest score into 0, and any other scores into a value between 0 and 1 by using the formula.

$$n\_score = \frac{(raw\_score - min\_score)}{(max\_score - min\_score)} \quad (1)$$

Here *max\_score* is the highest score, *min\_score* is the lowest score, *raw\_score* is the score to be normalized and *n\_score* is the normalized score of *raw\_score*. This normalization method was used by Lee [10] and others in their experiments.

The above zero-one normalization method can be improved in some situations. For example, in the TREC workshop, each system is usually required to submit 1000 documents for any given query. In such a situation, the top-ranked documents in the list are not always relevant, and the bottom-ranked documents are not always irrelevant. Therefore, Wu, Crestani, and Bi [25] considered that  $[a, b]$  ( $0 < a < b < 1$ ) should be a more suitable range than  $[0, 1]$  for score normalization. Experiments with TREC data were conducted to support this idea.

Two other linear score normalization methods, SUM and ZMUV (Zero-Mean and Unit-Variance), were investigated by Montague and Aslam [13]. In SUM, the minimal score is mapped to 0 and the sum of all scores in the result to 1. In ZMUV, the average of all scores is mapped to 0 and their variance to 1.

If only a ranked list of documents is provided without any scoring information, then we cannot use the linear combination method directly. One usual way of dealing with this is to assign a given score to documents at a particular rank. For example, Borda count [1] works like this: for a ranked list of  $t$  documents, the first document in the list is given a score of  $t$ , the second document in the list is given a score of  $t - 1$ , ..., the last document in the list is given a score of 1. Thus all documents are assigned corresponding scores based on their rank positions and CombSum or the linear combination method can be applied accordingly.

Some non-linear methods for score normalization have also been discussed in [3, 12, 15, 22]. In [12], an exponential distribution for the set of non-relevant documents and a normal distribution for the set of relevant documents were used, then a mixture model was defined to fit the real score distribution. The cubic regression model for the relation between documents' rank and degree of relevance was investigated in [22]. The logistic regression model were investigated in [3, 15] in the environment of distributed information retrieval. However, this model can also be used in data fusion without any problem.

### 3 The Linear Combination Method

Suppose we have  $n$  information retrieval systems  $ir_1, ir_2, \dots, ir_n$ , and for a given query  $q$ , each of them provides a result  $r_i$ . Each  $r_i$  comprises a ranked list of documents with associated scores. The linear combination method uses the following formula to calculate score for every document  $d$ :

$$M(d, q) = \sum_{i=1}^n \beta_i * s_i(d, q) \quad (2)$$

Here  $s_i(d, q)$  is the normalized score of document  $d$  in result  $r_i$ ,  $\beta_i$  is the weight assigned to system  $ir_i$ , and  $M(d, q)$  is the calculated score of  $d$  for  $q$ . All the documents can be ranked according to their calculated scores.

There are two major issues that need to be considered for using the linear combination method properly. Firstly, for every document retrieved by a retrieval system, it should have a reliable score that indicates the degree of relevance of that document to the query. Secondly, appropriate weights need to be assigned to all component systems for best possible fusion results. Next let us discuss these two issues one by one.

In this research, we use the binary logistic regression model to estimate the relation between ranking position of a document and its degree of relevance, because we consider it is a good, reliable method. However, some other score normalization methods may also be able to achieve similar results presented in this paper.

The logistic model can be expressed by the following equation

$$score(t) = \frac{e^{a+bt}}{1 + e^{a+bt}} = \frac{1}{1 + e^{-a-bt}} \tag{3}$$

In Equation 3,  $score(t)$  is the degree of the document at rank  $t$  being relevant to the query.  $a$  and  $b$  are two parameters. As in [3], we use  $ln(t)$  to replace  $t$  in the above equation 3:

$$score(t) = \frac{e^{a+b*ln(t)}}{1 + e^{a+b*ln(t)}} = \frac{1}{1 + e^{-a-b*ln(t)}} \tag{4}$$

The values of parameters  $a$  and  $b$  can be decided by some training data, as we shall see later in Section 4.

The second issue is how to train weights for all the component systems involved. In our work we use the multiple linear regression model to do this.

Suppose there are  $m$  queries,  $n$  information retrieval systems, and a total of  $r$  documents in a document collection  $D$ . For each query  $q^i$ , all information retrieval systems provide scores for all the documents in the collection. Therefore, we have  $(s_{1k}^i, s_{2k}^i, \dots, s_{nk}^i, y_k^i)$  for  $\{i=(1, 2, \dots, m), k=(1, 2, \dots, r)\}$ . Here  $s_{jk}^i$  stands for the score assigned by retrieval system  $ir_j$  to document  $d_k$  for query  $q^i$ ;  $y_k^i$  is the judged relevance score of  $d_k$  for query  $q^i$ . If binary relevance judgment is used, then it is 1 for relevant documents and 0 otherwise.

Now we want to estimate

$$Y = \{y_k^i; i = (1, 2, \dots, m), k = (1, 2, \dots, r)\}$$

by a linear combination of scores from all component systems. The least squares estimates of the  $\beta$ 's are the values  $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \dots,$  and  $\hat{\beta}_n$  for which the quantity

$$q = \sum_{i=1}^m \sum_{k=1}^r [y_k^i - (\hat{\beta}_0 + \hat{\beta}_1 s_{1k}^i + \hat{\beta}_2 s_{2k}^i + \dots + \hat{\beta}_n s_{nk}^i)]^2 \tag{5}$$



is a minimum. This is partly a matter of mathematical convenience and partly due to the fact that many relationships are actually of this form or can be approximated closely by linear equations.  $\beta_0, \beta_1, \beta_2, \dots$ , and  $\beta_n$ , the multiple regression coefficients, are numerical constants that can be determined from observed data.

In the least squares sense the coefficients obtained by multiple linear regression can bring us the optimum fusion results by the linear combination method, since they can be used to make the most accurate estimation of the relevance scores of all the documents to all the queries as a whole. In a sense this technique for fusion performance improvement is general since it is not directly associated with any special metrics. In information retrieval, all commonly used metrics are rank-based. Therefore, efforts have been focused on how to improve rankings. This can partially explain why a method like this has not been investigated. However, theoretically, this approach should work well for any reasonably defined rank-based metrics including average precision or recall-level precision and others, since more accurately estimated scores for all the documents are able for us to obtain better rankings of those documents.

In the linear combination method, those multiple regression coefficients,  $\beta_1, \beta_2, \dots$ , and  $\beta_n$ , are be used as weights for  $ir_1, ir_2, \dots, ir_n$ , respectively. Note that we do not need to use  $\beta_0$  to calculate scores, since the relative ranking positions of all the documents are not affected if a constant  $\beta_0$  is added to all the scores.

## 4 Experiments

In this section we present our experiment settings and results with four groups of runs submitted to TREC (TRECs 5-8, ad hoc track). Their information is summarized in Table 1.

**Table 1.** Information of the four groups of runs submitted to the ad-hoc track in TREC

Group	Total Number	Number of of runs selected	Number of queries
TREC 5	61	53	50
TREC 6	74	57	50
TREC 7	103	82	50
TREC 8	129	108	50

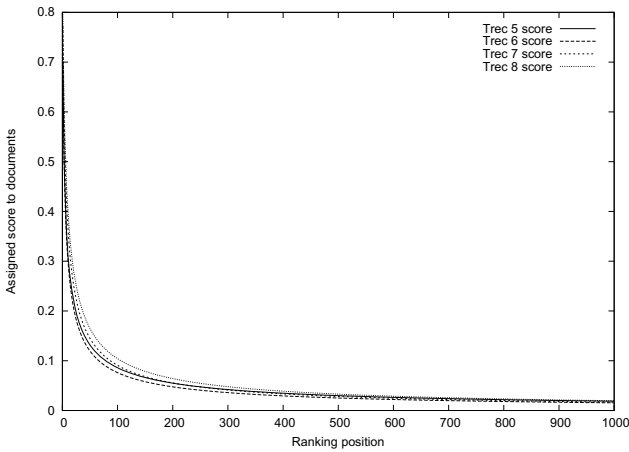
We only use those runs that include 1000 documents for every query. In each year group, some runs include fewer than 1000 documents for some queries. Removing those runs with fewer documents provides us a homogeneous environment for the investigation. Anyhow, we are still left with quite a large number of runs in each year group.

In some runs, the raw scores assigned to documents are not reasonable. Therefore, we do not use those raw scores in our experiment. Instead, we use the binary

**Table 2.** Coefficients obtained using binary logistic regression

Group	<i>a</i>	<i>b</i>
TREC 5	.821	-.692
TREC 6	.823	-.721
TREC 7	1.218	-.765
TREC 8	1.362	-.763

logistic model to generate estimated scores for documents at different ranks. In each year group, all the runs are put together. Then we use  $\ln(t)$  as the independent variable and  $score(t)$  as the dependant variable to obtain the coefficients of the logistic model. The coefficients we obtain are shown in Table 2.



**Fig. 1.** Corresponding scores of documents at different ranks based on the logistic model (coefficients are shown in Table 2)

Note that in a year group, for any query and any result (run), the score assigned to any document is only decided by the ranking position of the document. For example, for TREC 7, we obtain  $a=1.218$  and  $b=-.765$ . It follows from Equation 4 that

$$score(t) = \frac{1}{1 + e^{-1.218+0.765*\ln(t)}} \tag{6}$$

Therefore,  $score(1) = 0.7717$ ,  $score(2) = 0.6655, \dots$ , and so on. On the one hand, such scores are reliable since they come from the logistic regression model with all the data. On the other hand, such scores are not optimum by any means since

we treat all submitted runs and all queries equally. As a matter of fact, there is considerable variance from query to query, from one submitted run to another.

In Table 2, it can be noticed that the two pairs of parameters obtained in TREC 5 and TREC 6 are very close to each other, the two pairs in TREC 7 and TREC 8 are very close to each other, while it seems that the parameters in TRECs 5 & 6 are quite different from that in TRECs 7 & 8. However, the actual difference between TRECs 5 & 6 and TRECs 7 & 8 may not be as big as those parameters look like. In order to let us have a more intuitive understanding of them, we use a graph (Figure 1) to present the calculated scores based on those parameters. In Figure 1, we can see that all four curves are very close to each other. This demonstrates that all of them are very similar indeed.

With such normalized scores for the runs submitted, the next step is to use multiple linear regression to train system weights and carry out fusion experiments. For all 50 queries in each year group, we divided them into two groups: odd-numbered queries and even-numbered queries. First we used odd-numbered queries to train system weights and even-numbered queries to test data fusion methods; then we exchanged their positions by using even-numbered queries to train system weights and odd-numbered queries to test data fusion methods.

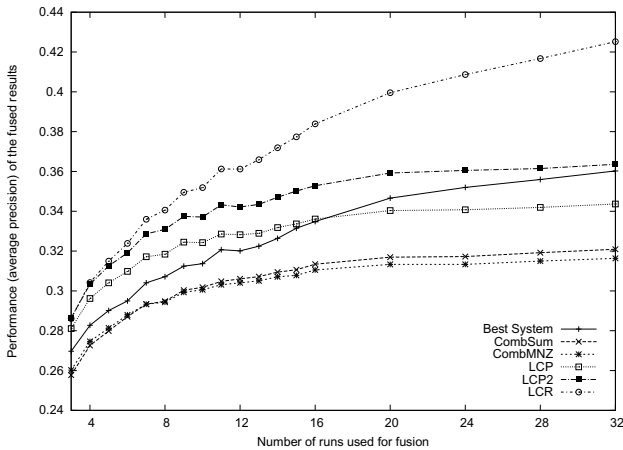
The file for linear multiple regression is a  $m$  row by  $(n + 1)$  column table. Here  $m$  is the total number of documents in all results and  $n$  is the total number of component systems involved in the fusion process. In the table, each record  $i$  is used for one document  $d_i$  ( $1 \leq i \leq m$ ) and any element  $s_{ij}$  ( $1 \leq i \leq m$ ,  $1 \leq j \leq n$ ) in record  $i$  represents the score that document  $d_i$  obtains from component system  $ir_j$  for a given query. Any element  $s_{i(n+1)}$  is the judged score of  $d_i$  for a given query. That is, 1 for relevant documents and 0 for irrelevant documents. The file can be generated by the following steps:

1. For each query, go through steps 2-5.
2. If document  $d_i$  occurs in at least one of the component results, then  $d_i$  is put into the table as a row.
3. If document  $d_i$  occurs in result  $r_j$ , then the score  $s_{ij}$  is calculated out by the logistic model (parameters are shown in Table 2).
4. If document  $d_i$  does not occur in result  $r_j$ , then a score of 0 is assigned to  $s_{ij}$ .
5. Ignore all those documents that are not retrieved by any component systems.

The above step 5 implies that any document that is not retrieved will be assigned a score of 0. Apart from the linear combination method with the trained weights by multiple regression (referred to as LCR later in this paper), CombSum, CombMNZ, the linear combination method with performance level weighting (referred to as LCP), and the linear combination method with performance square weighting (referred to as LCP2) are also involved in the experiment. For LCP and LCP2, we also divide all queries into two groups: odd-numbered queries and even-numbered queries. A run's performance measured by average precision on odd-numbered queries ( $AP(odd)$ ) is used for the weighting of even-numbered group ( $weight_{LCP}(even)$ ) and vice versa.

## 4.1 Experiment 1

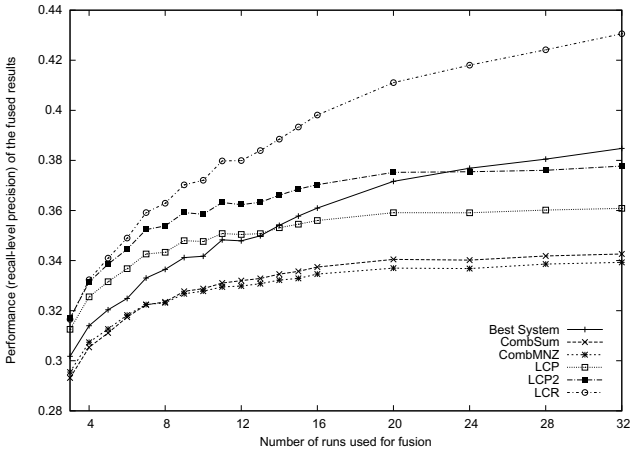
We first investigated the situation with various different number of component systems, from 3, 4, ..., to up to 32 component systems. For each given number  $k$  ( $3 \leq k \leq 32$ ), 200 combinations were randomly selected and tested. Eight metrics, including average precision, recall-level precision, precision at 5, 10, 15, 20, 30, and 100 document level, were used for performance evaluation. But in the rest of this paper, we only report three of them, which are average precision, recall-level precision, and precision at 10 document level. Results with other metrics are not presented because they are very similar to ones presented.



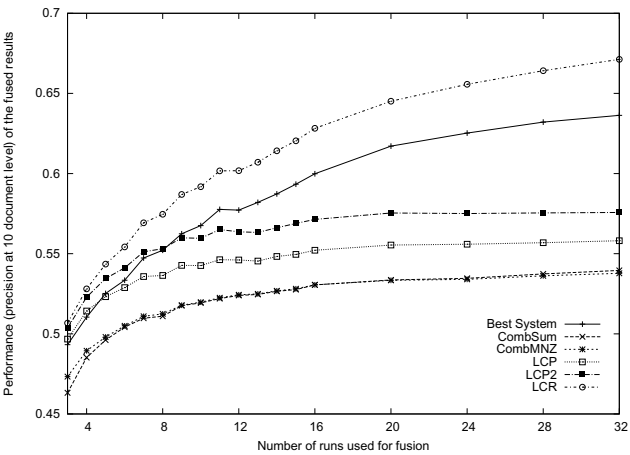
**Fig. 2.** Performance comparison of different data fusion methods (average of 4 groups and 200 combinations for each given number of component systems, average precision)

Figures 2, 3, and 4 show average precision (AP), recall-level precision (RP), and precision at 10 document level (P10), respectively, of all data fusion methods involved over 4 year groups. From Figures 2-4, we can see that LCR is the best. In the following, we compare them using the metric of average precision (AP) and the average of 4 year groups and 200 combinations for each given number of component systems. When 3 component systems are fused, LCR outperforms LCP, LCP2, CombSum, CombMNZ, and the best component systems by 1.78%, -0.14%, 11.02%, 9.95%, and 6.08%; when 16 component systems are fused, LCR outperforms LCP, LCP2, CombSum, CombMNZ, and the best component systems by 14.22%, 8.82%, 22.50%, 23.64%, and 14.67%; when 32 component systems are fused, LCR outperforms LCP, LCP2, CombSum, CombMNZ, and the best component systems by 23.71%, 16.94%, 32.50%, 34.39%, and 18.05%, respectively.

From Figures 2-4, we have a few further observations. Firstly, we can see that LCR consistently outperforms all other methods and the best component system



**Fig. 3.** Performance comparison of different data fusion methods (average of 4 groups and 200 combinations for each given number of component systems, recall-level precision)



**Fig. 4.** Performance comparison of different data fusion methods (average of 4 groups and 200 combinations for each given number of component systems, precision at 10 document level)

by a large margin. The only exception is LCP2, which is slightly better than LCR when fusing 3 component systems (AP: 0.14% improvement for 3 component systems, -0.30% for 4 component systems, and -0.80% for 5 component systems). As the number of component systems increase, the difference in performance between LCR and all other data fusion methods increase accordingly.

Secondly, LCP2 is always better than LCP, which confirms that performance square weighting is better than performance level weighting [23]. The difference between them is increasing with the number of component systems for some-time, and then becomes stable. Let us take average precision as an example: the differences are 1.92% for 3 component systems, 2.40% for 4 component systems, 2.76% for 5 component systems,..., 4.96% for 16 component systems. Then it stabilizes at 5.50% to 5.80% when the number reaches 20 and over. Two-tailed  $t$  test was carried out to compare the difference between these two methods. It shows that the differences are always highly significant ( $p$ -value  $< 2.2e-16$ ) whether 3, 4, 5, or more component systems are fused.

Thirdly, compared with the best component system, LCP and LCP2 perform better when a small number of component systems are fused. However, there are significant difference when different metrics are used for evaluation. Generally speaking, average precision is the metric that favours LCP and LCP2, while precision at 10 document level favours the best component system. For LCP2, if average precision is used, then it is always better than the best component system; if recall-level precision is used, then it is better than the component system when 20 or fewer component systems are fused; if precision at 10 document level is used, then the number of component systems decreases to 8 or 9 for it to outperform the best component system. For LCP, 16, 12, and 4 are the maximum numbers of component systems can be involved for it to beat the best component system, if AP, RP, and P10 are used, respectively.

Fourthly, CombSum and CombMNZ perform badly compared with all the linear combination data fusion methods and the best component system. It demonstrates that equally treating all component systems is not a good fusion policy when a few poorly performed component systems are involved.

Fifthly, as to effectiveness, the experimental results demonstrate that we can benefit from fusing a large group of component systems no matter which fusion method we use. Generally speaking, the more component systems we fuse, the better fusion result we can expect. This can be seen from all those curves that are increasing with the number of component systems in all 3 figures.

Finally, it demonstrates that LCR can cope well with all sorts of component systems that are very different in effectiveness and reciprocal similarity, even some of the component systems are very poor. It can bring effectiveness improvement steadily for different groups of component systems no matter which rank-based metric is used.

## 4.2 Experiment 2

Compared with some numerical optimisation methods such as conjugate gradient and golden section search, the multiple linear regression technique is much

more efficient and much more likely to be used in real applications. However, if a large number of documents and a large number of information retrieval systems are involved, then efficiency might be an issue that needs to be considered. For example, in some dynamic situations such as the Web, documents are updated frequently. Weights may need to be calculated quite frequently so as to obtain good performance. Therefore, we consider how to use the linear regression technique in a more efficient way.

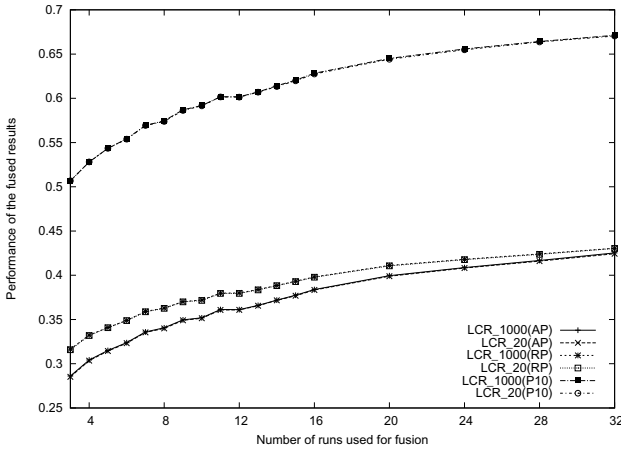
In Experiment 1, we used all 1000 documents for every query to form two training data sets: odd-numbered group and even-numbered group. Then the multiple linear regression applied to these data sets to obtain weights. This time we did not use all 1000 documents, but the top 20 documents for each query to form training data sets. Then the multiple linear regression was applied as before.

We carried out the experiment to compare the performance of the linear combination data fusion method with two groups of different weights. One group uses the training data set of all 1000 documents, as in Experiment 1; the other group uses the training data set that only includes top 20 documents for each of the queries. Apart from that, the methodology in this experiment is just the same as in Experiment 1. As a matter of fact, we reused the result in Experiment 1. We just rerun the experiment with weights obtained by using the 20-document training set. The number of combinations and all the component systems involved in each combination run are just the same as in Experiment 1. The experimental result is shown in Figure 5. In Figure 5, LCR\_1000 represents the LCR method using the 1000-document training set and LCR\_20 represents the LCR method using the 20-document training set. Note that LCR\_1000 is exactly the same as LCR in Experiment 1. In Figure 5, for AP, RP, and P10, the curves of LCR\_1000 and LCR\_20 overlap with each other and are totally indistinguishable. In almost all the cases, the difference between them is less than 0.3%. This demonstrates that using the top-20 documents for each of the queries as training data set can do the same job, no better and no worse, as using all 1000 documents. Obviously, using fewer documents in training data set is beneficial if considering the aspect of efficiency.

We also measured the time needed for weighting calculation by the multiple linear regression. A personal computer <sup>2</sup>, which installed R with Windows interface package “Rcmdr”, and two groups of data, TREC 5 and TREC 7 (even-numbered queries) were used for this part of the experiment. The time needed for the multiple linear regression is shown in Table 3. Note that in this experiment, for a year group, we imported data of all component systems into the table and randomly selected 5, 10, 15, and 20 from them to carry out the experiment. It is probably because the TREC 7 group comprises more component systems (82) than the TREC 5 group does (53), it took a little more time for the multiple regression to deal with TREC 7 than TREC 5 when the same number of component systems (5, 10, 15, or 20) are processed. We can see that roughly the time

---

<sup>2</sup> It is installed with Intel Duo core E8500 3.16GHz CPU, 2 GB of RAM and Windows operating system.



**Fig. 5.** Performance comparison of the linear combination fusion methods with two groups of different weights trained using different data sets (average of 4 groups and 200 combinations for each given number of component systems)

**Table 3.** Time (in seconds) needed for each multiple linear regression run

No. of variables	TREC 5		TREC 7	
	1000	20	1000	20
5	1.20	0.06	1.40	0.07
10	1.60	0.08	1.70	0.09
15	2.00	0.10	2.80	0.12
20	2.70	0.14	3.60	0.14

needed for the 20-document data set is 1/20 of the time for the 1000-document data set, though the number of records in the 20-document data set is 1/50 of that in the 1000-document data set. But a difference like this may still be a considerable advantage for many applications. It is also possible to speed up the process if we only upload those data needed into the table.

## 5 Conclusion

In this paper we have investigated a new data fusion method that uses the multiple linear regression to obtain suitable weights for the linear combination method. The weights obtained in this way is optimum in the least squares sense of estimating relevance scores of all related documents by linear combination. The extensive experiments with 4 groups of runs submitted to TREC have demonstrated that it is a very good data fusion method. It outperforms other major data fusion methods such as CombSum, CombMNZ, the linear combination method with performance level weighting or performance square weighting, and



the best component system, by large margins. We believe this piece of work is a significant progress to the data fusion technique in information retrieval and can be used to implement more effective information retrieval systems.

## References

1. Aslam, J.A., Montague, M.: Models for metasearch. In: Proceedings of the 24th Annual International ACM SIGIR Conference, New Orleans, Louisiana, USA, pp. 276–284 (September 2001)
2. Bartell, B.T., Cottrell, G.W., Belew, R.K.: Automatic combination of multiple ranked retrieval systems. In: Proceedings of ACM SIGIR 1994, Dublin, Ireland, pp. 173–184 (July 1994)
3. Calvé, A.L., Savoy, J.: Database merging strategy based on logistic regression. *Information Processing & Management* 36(3), 341–359 (2000)
4. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: Proceedings of the Tenth International World Wide Web Conference, Hong Kong, China, pp. 613–622 (May 2001)
5. Efron, M.: Generative model-based metasearch for data fusion in information retrieval. In: Proceedings of the 2009 Joint International Conference on Digital Libraries, Austin, USA, pp. 153–162 (June 2009)
6. Farah, M., Vanderpooten, D.: An outranking approach for rank aggregation in information retrieval. In: Proceedings of the 30th ACM SIGIR Conference, Amsterdam, The Netherlands, pp. 591–598 (July 2007)
7. Fox, E.A., Koushik, M.P., Shaw, J., Modlin, R., Rao, D.: Combining evidence from multiple searches. In: The First Text REtrieval Conference (TREC-1), Gaithersburg, MD, USA, pp. 319–328 (March 1993)
8. Fox, E.A., Shaw, J.: Combination of multiple searches. In: The Second Text REtrieval Conference (TREC-2), Gaithersburg, MD, USA, pp. 243–252 (August 1994)
9. Juárez-González, A., Montes y Gómez, M., Pineda, L., Avendaño, D., Pérez-Coutiño, M.: Selecting the n-top retrieval result lists for an effective data fusion. In: Proceedings of 11th International Conference on Computational Linguistics and Intelligent Text Processing, Iasi, Romania, pp. 580–589 (March 2010)
10. Lee, J.H.: Analysis of multiple evidence combination. In: Proceedings of the 20th Annual International ACM SIGIR Conference, Philadelphia, Pennsylvania, USA, pp. 267–275 (July 1997)
11. Lillis, D., Zhang, L., Toolan, F., Collier, R., Leonard, D., Dunnion, J.: Estimating probabilities for effective data fusion. In: Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Geneva, Switzerland, pp. 347–354 (July 2010)
12. Manmatha, R., Rath, T., Feng, F.: Modelling score distributions for combining the outputs of search engines. In: Proceedings of the 24th Annual International ACM SIGIR Conference, New Orleans, USA, pp. 267–275 (September 2001)
13. Montague, M., Aslam, J.A.: Relevance score normalization for metasearch. In: Proceedings of ACM CIKM Conference, Berkeley, USA, pp. 427–433 (November 2001)
14. Montague, M., Aslam, J.A.: Condorcet fusion for improved retrieval. In: Proceedings of ACM CIKM Conference, USA, pp. 538–548 (November 2002)
15. Nottelmann, H., Fuhr, N.: From retrieval status values to probabilities of relevance for advanced ir applications. *Information Retrieval* 6(3-4), 363–388 (2003)

16. Renda, M.E., Straccia, U.: Web metasearch: rank vs. score based rank aggregation methods. In: Proceedings of ACM 2003 Symposium of Applied Computing, Melbourne, USA, pp. 847–452 (April 2003)
17. Shokouhi, M.: Segmentation of search engine results for effective data-fusion. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECiR 2007. LNCS, vol. 4425, pp. 185–197. Springer, Heidelberg (2007)
18. Thompson, P.: Description of the PRC CEO algorithms for TREC. In: The First Text REtrieval Conference (TREC-1), Gaithersburg, MD, USA, pp. 337–342 (March 1993)
19. Vogt, C.C., Cottrell, G.W.: Predicting the performance of linearly combined IR systems. In: Proceedings of the 21st Annual ACM SIGIR Conference, Melbourne, Australia, pp. 190–196 (August 1998)
20. Vogt, C.C., Cottrell, G.W.: Fusion via a linear combination of scores. *Information Retrieval* 1(3), 151–173 (1999)
21. Wu, S.: Applying statistical principles to data fusion in information retrieval. *Expert Systems with Applications* 36(2), 2997–3006 (2009)
22. Wu, S., Bi, Y., McClean, S.: Regression relevance models for data fusion. In: Proceedings of the 18th International Workshop on Database and Expert Systems Applications, Regensburg, Germany, pp. 264–268 (September 2007)
23. Wu, S., Bi, Y., Zeng, X., Han, L.: Assigning appropriate weights for the linear combination data fusion method in information retrieval. *Information Processing & Management* 45(4), 413–426 (2009)
24. Wu, S., Crestani, F.: Data fusion with estimated weights. In: Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, pp. 648–651 (November 2002)
25. Wu, S., Crestani, F., Bi, Y.: Evaluating score normalization methods in data fusion. In: Ng, H.T., Leong, M.-K., Kan, M.-Y., Ji, D. (eds.) AIRS 2006. LNCS, vol. 4182, pp. 642–648. Springer, Heidelberg (2006)
26. Wu, S., McClean, S.: Data fusion with correlation weights. In: Proceedings of the 27th European Conference on Information Retrieval, pp. 275–286. Santiago de Composite, Spain (2005)
27. Wu, S., McClean, S.: Improving high accuracy retrieval by eliminating the uneven correlation effect in data fusion. *Journal of American Society for Information Science and Technology* 57(14), 1962–1973 (2006)
28. Zhou, D., Lawless, S., Min, J., Wade, V.: A late fusion approach to cross-lingual document re-ranking. In: Proceedings of the 19th ACM Conference on Information and Knowledge Management, Toronto, Canada, pp. 1433–1436 (October 2010)

# Approaches and Standards for Metadata Interoperability in Distributed Image Search and Retrieval

Ruben Tous, Jordi Nin, Jaime Delgado, and Pere Toran

Universitat Politècnica de Catalunya (UPC BarcelonaTech), Barcelona, Spain  
rtous@ac.upc.edu, nin@ac.upc.edu, jaime.delgado@ac.upc.edu,  
ptoran@ac.upc.edu

**Abstract.** This paper addresses the general problem of how building a distributed image search&retrieval system that copes with metadata heterogeneity. Firstly, We analyze the usage of metadata in current image repositories, the different metadata schemas for image description and their semantic relationships. After, we provide a general classification for the different approaches which provide a unified interface to search images hosted in different systems without degrading query expressiveness. This paper analyzes how these approaches can be implemented on top of the latest standards in the area, ISO/IEC 15938-12 (MPEG Query Format) for the query interface interoperability and ISO/IEC 24800 (JPSearch) for the definition and management of translations between metadata schemas. Finally, we provide insights into an example of a real distributed image search&retrieval system which provides real time access to Flickr, Picassa and Panoramio.

**Keywords:** metadata, interoperability, image, information retrieval, standards, jpsearch, mpqf.

## 1 Introduction

Nowadays, digital images are being generated, distributed and stored worldwide at an ever increasing rate. Consequently, in the recent years, image Search&Retrieval tasks arise as an important issue. There are multiple systems, however, almost every one provides a different search interface and multimedia metadata description format. This fact prevents users from experiencing a unified access to the repositories. Systems aiming to provide a unified query interface to search images hosted in different systems without degrading query expressiveness need to address several questions which include but are not limited to the following:

- Is the system going to harvest all the metadata and store them locally? How frequently do the data change and how is the system going to cope with data volatility?
- Which metadata schema or schemas are going to be exposed to user queries? Is the system going to expose a mediated/pivot schema?

- How the mappings among the underlying target metadata schemas are going to be generated?
- Which formalism is going to be used to describe the mappings?
- How is the system going to use the mappings during querying?

Currently many standardization efforts are trying to provide answers to some of these questions. Two of the most relevant initiatives are the ISO/IEC 15938-12:2008 (MPEG Query Format or simply MPQF) [5][10][13] and ISO/IEC 24800 (JPEG's JPSearch framework) [14][15]. While MPQF offers a solution for the query interface interoperability, JPSearch (whose Part 3 makes use of MPQF) faces the difficult challenge to provide an interoperable architecture for images metadata management.

This paper provides an analysis of the current approaches and standards for metadata interoperability in distributed image search&retrieval systems. In order to better feature the problem being faced, firstly the paper examines the main metadata schemas currently used for image description and identifies their semantic relationships. Secondly, the paper provides a classification of the different approaches and evaluates their advantages and drawbacks. Thirdly, the paper analyzes how distributed image search&retrieval systems can be implemented on top of the ISO/IEC 15938-12 and ISO/IEC 24800 standards. Finally, the paper provides insights into an example real distributed image search&retrieval system which provides real time access to Flickr, Picassa and Panoramio.

## 2 Topology of Digital Image Description Metadata Models

In order to better feature the image metadata heterogeneity problem, let's first study which are the main metadata schemas used for image description and which are their relationships. We have selected the following list of metadata schemas, which is not comprehensive, but allows the reader to obtain some conclusions that can be easily extrapolated:

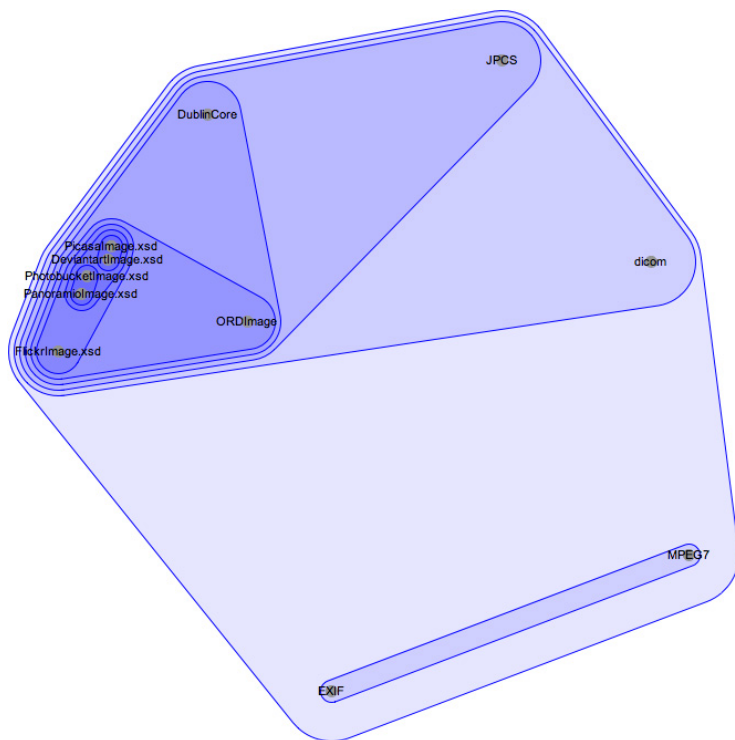
- *Flickr*. Metadata schema [19] representing the image description model of Flickr, the most popular image hosting and video hosting website.
- *Picasa*. Metadata schema [22] representing the image description model of Picasa, Google's image organizer, image viewer an integrated photo-sharing website.
- *Panoramio*. Metadata schema [20] representing the image description model of Panoramio. A geolocation-oriented photo sharing website.
- *Photobucket*. Metadata schema [21] representing the image description model of Photobucket, an image hosting, video hosting, slideshow creation and photo sharing website usually used for personal photographic albums, remote storage of avatars displayed on internet forums, and storage of videos.
- *DeviantART*. Metadata schema [18] representing the image description model of DeviantART, an online community showcasing various forms of user-made artwork.

- *ORDImage*. Metadata schema of the Oracle Multimedia’s *ORDImage* type, which supports the storage, management, and manipulation of image data.
- *DICOM*. Metadata schema of the Digital Imaging and Communication in Medicine (DICOM), a standard for handling, storing, printing, and transmitting information in medical imaging.
- *MPEG-7*. Metadata schema of ISO/IEC 15938 (Multimedia Content Description Interface).
- *Dublin Core*. Metadata schema of Dublin Core (ISO Standard 15836, and NISO Standard Z39.85-2007).
- *JPSearch Core Schema*. Metadata schema of ISO/IEC 24800-2 (JPSearch Core Schema).
- *EXIF*. Exchangeable image file format is a specification for the image file format used by digital cameras (including smartphones) and scanners. its latest version dated April 2010 (2.3) was jointly formulated by JEITA (Japan Electronic Industries Development Association) and CIPA (Camera & Imaging Products Association).

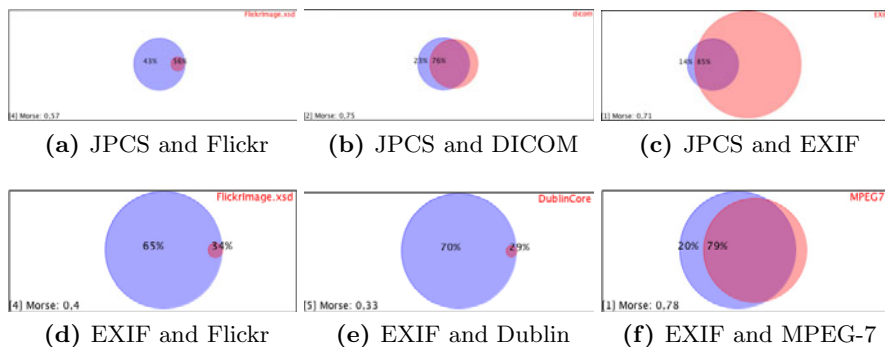
To visualize and compare the different schemas, we have used OpenII [24], a novel schema management tool. OpenII is a suite of open-source tools for information integration (II). The suite includes the following components:

- *Affinity* is a clustering tool for entire schemas. It groups schemas using hierarchical clustering techniques.
- *Harmony* is a semi-automated tool that finds the correspondences across two data schemas using a set of semantic matchers.
- *Proximity* visualizes in a graphical way the inclusion/exclusion relations between a selected reference model and all the others.
- *Unity* semi-automatically produces a common vocabulary, based on aligning the schema elements in a set of source schemas. The vocabulary consists of a list of canonical terms the source schemas agree on.
- *Yggdrasil* is a repository implementing an entity relationship metamodel (called M3) for both schemas and mappings. This repository is implemented on top of a Postgres database. Yggdrasil allows for importing / exporting XML Schemas.

We have used OpenII to find the relationships among the different schemas, and to visualize the topology of the different image description metadata models. Figure 1 shows the results of clustering all the metadata models with the the OpenII’s *Affinity* tool. *EXIF* and *MPEG-7*, being the more comprehensive and general schemas, appear clustered together at the bottom of the figure. Schemas from generic image hosting websites, such as the ones from Flickr, Picasa, Panoramio, Photobucket or DevianART, appear clustered together at the top-left of the figure, with *ORDImage* not far away. These schemas are flat and simple, and they have significant overlappings. *DICOM*, a specialized schema for medical image tagging, appears at the right of the figure, while the *JPSearch Core Schema* appears at the top.



**Fig. 1.** Relations found for all the schemas using the OpenII's Affinity tool



**Fig. 2.** Relationship between different schemas obtained using the OpenII's Proximity tool

We have also used OpenII to find one-to-one relationships between schemas. The OpenII's Harmony tool allows to automatically obtain the correspondences between two given metadata schemas. Due to the lack of space, we do not include the produced mappings, however, in figure 2 we show the inclusion/exclusion relations between some selected schemas using the harmony tool. Figures 2(a) and 2(d) depict that the Flickr metadata information is completely included in the JPSearch Core Schema and EXIF, two general image tagging schemas. Figure 2(c) shows that EXIF metadata is more general and includes more information than JPCS schema. Additionally, this figure shows that most of the JPCS information (85%) is also included in EXIF. Similar conclusions can be extracted from the Figure 2(f) where we compare EXIF and MPEG-7 schemas, in this case we also see that MPEG-7 includes more information than JPCS (by comparing the relative sizes of both schemas in Figures 2(c) and 2(f)).

Figure 2(b) illustrates that only the 76% of the DICOM metadata information is shared by JPCS, this has sense because DICOM schema has medical specific information such as patient position or patient orientation. Finally, we would like to comment Figure 2(e), where we show the relation between EXIF and Dublin core schema. Both schemas are general but the Dublin schema clearly has less information than EXIF, and Dublin information is completely included in EXIF.

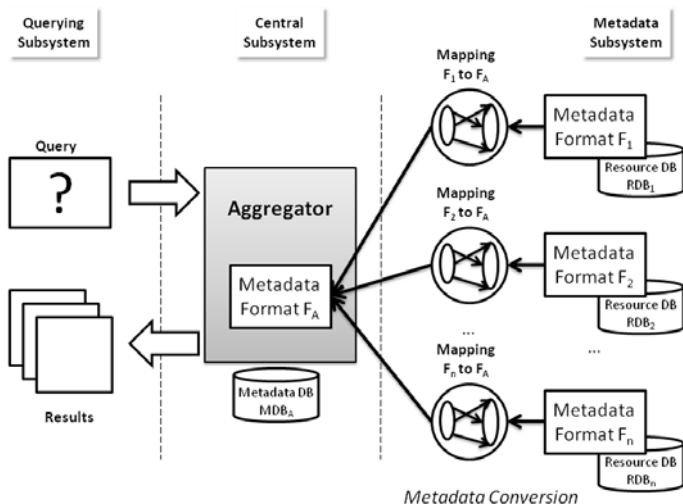
### 3 Approaches to Metadata Interoperability in Distributed Image Search&Retrieval

In this section we provide an overview to the three approaches that we propose to classify the different solutions for the image metadata interoperability problem. Each approach is labelled with a double title. The first part of the title always refers to the most characteristic concept that identifies the architecture of each proposal: Simple Aggregator, Multiple Aggregator and Broker. The second part of the title illustrates the implementation issues implied in each approach. In the Simple Aggregator approach it is necessary to deal with Metadata Conversion issues, while in the Multiple Aggregator approach it is necessary to focus on Inference. Finally, in the Broker approach it is necessary to face Query Rewriting related topics. The architecture in each of the approaches is subdivided into three subsystems: the Querying subsystem, the Central subsystem and the Metadata subsystem. The Querying subsystem manages the queries that the user inputs. The Metadata subsystem represents the different external systems that the Central subsystem makes transparently interoperable. We have focused mainly in the Central and Metadata subsystems.

#### 3.1 Simple Format Metadata Aggregator/Metadata Conversion

The main idea of this approach is that we have a centralized system, called Aggregator, which holds all the metadata needed to satisfy the user queries. The key issue is that we have an intermediate metadata format for the Aggregator.

Thus, all the supported formats need to provide a mapping mechanism to convert from their own metadata format to the Aggregator metadata format. Figure 3 presents the basic Simple Format Metadata Aggregator (SFMA) architecture diagram. The Aggregator (that is part of the Central subsystem) receives a query. As we said, in order to answer to that query the Aggregator needs to have all the external metadata formats available and converted to its own one. All supported external Metadata Formats ( $F_1, F_2, \dots, F_n$ ) need to provide some mapping mechanism ( $F_1$  to  $F_A, F_2$  to  $F_A, \dots, F_n$  to  $F_A$ ) to map from their own format to the one supported by the Aggregator ( $F_A$ ). Thus, assuming that the Aggregator has all the information obtained from the external metadata formats converted to its own metadata format, the system uses this metadata database to obtain the results demanded and send back them to the user.



**Fig. 3.** Basic architecture for the "Simple Format Metadata Aggregator" (SFMA) approach

In Figure 3 we have also added the different databases distribution used by the system. It is interesting to include this information in the figure to compare the different distributions of the databases that every approach implies. Basically, we will have two types of databases: (1) resource databases and (2) metadata databases. In the resource database is where the actual resources we search for are located, i.e. the image file. On the other hand, the metadata database contains all the metadata information associated to the images of the resource database. In the SFMA approach, the metadata database is located at the Central subsystem. It aggregates the different metadata databases into a single one labelled  $MDB_A$ , which is the unique metadata database that the system uses. However, the resource databases ( $RDB_1, RDB_2, \dots, RDB_n$ ) may remain at the



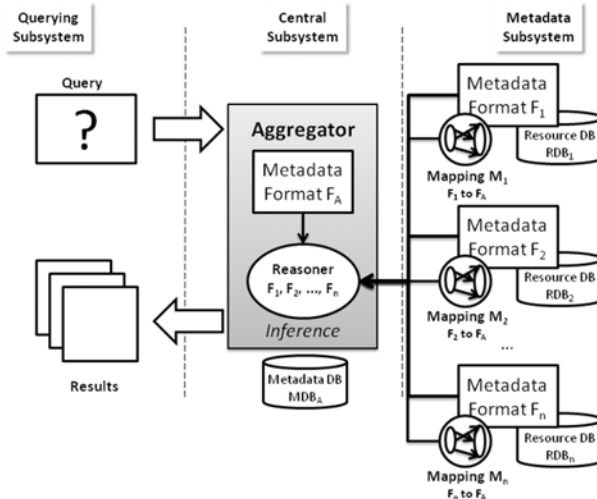
external systems and the Aggregator obtains the resource directly from there after the query  $Q$  has been solved. This approach greatly simplifies the query evaluation process, but does still require a solution for metadata conversion.

### 3.2 Multiple Format Metadata Aggregator/Inference

In this model, the Aggregator is able to answer queries using previously loaded information (metadata and mapping information) from the external metadata-based systems that the Central subsystem supports.

The main characteristic of the Multiple Format Metadata Aggregator (MFMA) approach (see Figure 4) is that it is entirely based on ontologies and an ontology Reasoner [2]. Thus, the information that the Aggregator expects from every external system (and needs to be preloaded before receiving any query) is an ontology representing each external metadata format ( $F_1, F_2, \dots, F_n$ ) and another ontology that provides the mapping from a given external metadata format to the internal metadata format of the Aggregator ( $F_1$  to  $F_A, F_2$  to  $F_A, \dots, F_n$  to  $F_A$ ). In addition to that, the Aggregator also needs to have the metadata information of the actual content that the user is querying for preloaded. This metadata is stored in a metadata database ( $MDB_A$ ) in the form of ontology individuals (extracted from or provided by the external systems in the Metadata subsystem).

So, when the Aggregator receives a query, the Reasoner expands all the ontology information using the mappings provided and infers the results before sending them back to the user. Typically, the results will be direct links to the



**Fig. 4.** Basic architecture for the "Multiple Format Metadata Aggregator" (MFMA) approach

external Resource Databases ( $RDB_1, RDB_2, \dots, RDB_n$ ) of the supported systems, where the actual content (i.e. images, videos, music...) is stored.

Examples of the MFMA approach include all the projects providing semantic search capabilities over Linked Data [7] related datasets. Linked Data is a way of exposing RDF metadata and interlinking them. In the years significant amounts of data have been generated, increasingly forming a globally connected, distributed data space. DBpedia [8] is an example for such a source in the Web of Data. In the case of multimedia there are examples such as the The Linked Movie DataBase (LinkedMDB) [16], which contains semantic metadata from several movie-related data sources, or DBtune [4], a collection of music-related data sets.

such as DBpedia [8]

### 3.3 Broker/Query Rewriting

The Broker acts as an intermediary agent communicating the system user to several external systems. The Broker approach is the only one that does not rely on the local storage of significant amounts of metadata (even though it needs to temporally store certain metadata for query reprocessing). It is based on the definition of a mediated schema,  $F_B$ , query rewriting and query reprocessing. This approach can be seen as a reformulation of some issues already faced by the Data Integration community, such as the classic *global-as-view* approach or GAV [1] (TSIMMIS) [3] [11] and the classic *local-as-view* approach or LAV [17] [12] [6]. However, there's a more recent definition of the Broker approach in [25]. Figure 4 shows the basic Broker architecture diagram.

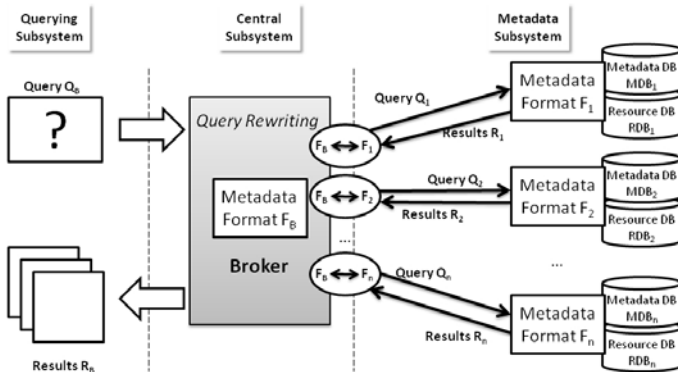


Fig. 5. Basic architecture for the "Broker" approach

The Broker receives a query ( $Q_B$ ) in the metadata format  $F_B$ , it rewrites it once for every metadata format that is supported ( $F_1, F_2, \dots, F_n$ ) via the corresponding interface ( $F_B \leftrightarrow F_1, F_B \leftrightarrow F_2, \dots, F_B \leftrightarrow F_n$ ). The queries  $Q_1, Q_2, \dots, Q_n$  are generated and sent to every external system. The resulting data

is processed back again by the corresponding interface, mapped back to the metadata format  $F_B$  and presented to the user. The local reprocessing of results allows answering queries addressing properties not available at the source's query interfaces but included in the result sets. When a source does not include a metadata field neither in the query interface nor in the result set the source cannot be used for a query addressing this metadata field and the user must be properly informed.

In the Broker approach, we have all the metadata databases ( $MDB_1, MDB_2, \dots, MDB_n$ ) and all the resource databases ( $RDB_1, RDB_2, \dots, RDB_n$ ) at the metadata subsystem, so there is no need to maintain any kind of central database at the Central subsystem.

As we can see, from the architectural point of view this approach could be classified as a Broker based model, but from the implementation point of view we could talk about a Query Rewriting scenario [9]. The reason is that the main responsibility of the Broker system is to rewrite every query that receives and propagate it to all supported external systems.

### 3.4 Approaches Comparison

The Simple Format Metadata Aggregator is strongly related to the Metadata Conversion topic. All the incorporated metadata formats are converted to the internal metadata format of the aggregator before the system is ready to answer to user queries. This approach has probably a more static and deterministic nature in the sense that its behavior is easier to predict because of the absolute control of the metadata information that it offers. *SFMA* systems are easy to design and implement and offer better performance than the other approaches. However, the need to harvest and locally store all the metadata can become a problem if the amount of metadata or their volatility is too big. This approach would probably be suitable for closed applications such as local multimedia repositories for multimedia players or file indexers.

The Broker approach requires a more complex design and implementation, it avoids the problems related to scalability and volatility. However, its performance is sensible to eventual problems (e.g. delays or unavailability) in the underlying databases. This approach would be suitable for distributing queries among big Internet image hosting services such as Flickr, Picassa or Panoramio.

The Multiple Format Metadata Aggregator is entirely based on the concept of Inference in ontologies through a reasoner. In this approach we also need to build a centralized metadata database in the Central subsystem. However, this time we do not need to perform metadata conversion before including the metadata information into the central database, the semantic correspondences are considered by the inference reasoner on-the-fly. *MFMA* systems have small design complexity, being the main task the ontology mapping design. Besides, the inclusion of support new formats is straightforward, only requiring to be fed the knowledge base with the corresponding new ontologies and mappings.

## 4 Standards for Metadata Interoperability in Distributed Image Search&Retrieval

In this section we describe two metadata interoperability standards for image search&retrieval. The first one, MPQF, proposed by the MPEG working group and the second one, JPsearch, proposed by the JPEG committee.

### 4.1 ISO/IEC 15938-12:2008 Standard (MPEG Query Format or MPQF)

A key element in all the different approaches to distributed image search&retrieval is the interchange of queries and API calls among all the involved parties. The usage of different proprietary interfaces for this task makes extremely difficult the deployment of distributed image search services without degrading the query expressiveness. The progressive adoption of an unified query interface would greatly alleviate this problem. For its features, we conclude that the ISO/IEC 15938-12:2008 standard (MPEG Query Format or MPQF) is the most suited language for this purpose. MPQF is an XML-based query language that defines the format of queries and replies to be interchanged between clients and servers in a distributed multimedia information search&retrieval context. MPQF is an XML-based in the sense that all MPQF instances (queries and responses) must be XML documents. Formally, MPQF is Part 12 of ISO/IEC 15938, "Information Technology - Multimedia Content Description Interface" (MPEG-7 [23]). However, the query format was technically decoupled from MPEG-7 and it is now metadata-neutral. So, MPQF is not coupled to any particular metadata standard.

Example in Code 1 shows an input MPQF query asking for JPEG images taken after the 2011/01/15 with the keyword "Tokyo" somewhere in their metadata.

### 4.2 ISO/IEC 24800 Standard (JPSearch)

The selection of a unified query interface is not enough to guarantee interoperability if it is not accompanied with a proper mechanism to manage metadata heterogeneity. The need of dealing with the management of metadata translations is a common factor in all the approaches to distributed image search&retrieval. Currently there is a standard solution to this problem proposed by the JPEG Committee, named JPSearch (ISO/IEC 24800). JPSearch provides a set of standardized interfaces of an abstract image retrieval framework. On one hand, JPSearch specifies the pivot JPSearch's Core Metadata Schema as the main component of the metadata interoperability strategy in ISO/IEC 24800. The core schema contains a set of minimal core terms which serve as metadata basis supporting interoperability during search among multiple image retrieval systems. The core schema is used by clients to formulate, in combination with the MPEG Query Format, search requests to JPSearch compliant search systems. In addition to the definition of JPSearch Core Metadata Schema, ISO/IEC 24800 provides a mechanism which allows a JPSearch

---

**Code 1.** Example MPQF input query

---

```

<MpegQuery>
  <Query>
    <Input>
      <OutputDescription>
        <ReqField>title</ReqField>
        <ReqField>date</ReqField>
      </OutputDescription>
      <QueryCondition>
        <TargetMediaType>image/jpg
        </TargetMediaType>
        <Condition xsi:type="AND">
          <Condition xsi:type="QueryByFreeText">
            <FreeText>Tokyo</FreeText>
          </Condition>
          <Condition xsi:type="GreaterThanOrEqual">
            <DateTimeField>date</DateTimeField>
            <DateValue>2011-01-15</DateValue>
          </Condition>
        </Condition>
      </QueryCondition>
    </Input>
  </Query>
</MpegQuery>

```

---

compliant system taking profit from proprietary or community-specific metadata schemas. A translation rules language is defined, allowing the publication of machine-readable translations between metadata terms belonging to proprietary metadata schemas and metadata terms in the JPSearch Core Metadata Schema. Users can choose which metadata language to use in a JPSearch-based interaction (annotation, querying, etc.) if the proper translations are available.

On the other hand, JPSearch specifies JPSearch Translation Rules Declaration Language (JPTRDL). JPTRDL allows the publication of machine-readable translations between metadata terms belonging to proprietary metadata schemas and metadata terms in the JPSearch Core Metadata Schema. Users can choose which metadata language to use in a JPSearch-based interaction if the proper translations are available. Code 2 shows a one-to-many translation rule which maps the JPSearch Core Schema date element into three fields.

### 4.3 JPSearch Registration Authority

According to the JPSearch specification, ISO/IEC 24800 compliant systems can manage multiple proprietary or community-specific metadata schemas, besides the JPSearch Core Metadata Schema. The multiplicity of schemas is solved by allowing the publication of machine-readable translations between metadata terms belonging to proprietary metadata schemas and metadata terms in the

**Code 2.** Example JPSearch translation rule

---

```

<?xml version="1.0" encoding="iso-8859-1"?>
<TranslationRules>
  <TranslationRule xsi:type="OneToManyFieldTranslationType">
    <FromField xsi:type="FilteredSourceFieldType">
      <XPathExpression>date</XPathExpression>
      <FilterWithRegExpr>(\d\d)/(\d\d)/(\d\d\d\d)</FilterWithRegExpr>
    </FromField>
    <ToField xsi:type="FormattedTargetFieldType">
      <XPathExpression>day</XPathExpression>
      <ReplaceWithRegExpr>£1</ReplaceWithRegExpr>
    </ToField>
    <ToField xsi:type="FormattedTargetFieldType">
      <XPathExpression>month</XPathExpression>
      <ReplaceWithRegExpr>£2</ReplaceWithRegExpr>
    </ToField>
    <ToField xsi:type="FormattedTargetFieldType">
      <XPathExpression>year</XPathExpression>
      <ReplaceWithRegExpr>£3</ReplaceWithRegExpr>
    </ToField>
  </TranslationRule>
</TranslationRules>

```

---

JPSearch Core Metadata Schema. In order to rationalize the usage of schemas and translation rules across different JPSearch systems, Subclause 3.3.3 of Part 2 of ISO/IEC 24800-2 specifies that a global authority for schemas and their translation rules will be established where all JPSearch compliant retrieval applications can obtain the information needed.

The establishment of a JPSearch Registration Authority (JPSearch RA) was formally approved during the 54th JPEG meeting in Tokyo, Japan, in February 2011, and will be operative in July 2011. The JPSearch RA will maintain a list of Metadata Schemas together with their related Translation Rules, if any. Those schemas and rules will be directly stored in the JPSearch RA web site or the JPSearch RA web site will provide a link to an external organization in charge of keeping that information updated. Registration forms will be available from the Registration Authority. Any person or organization will be eligible to apply. More information about the JPSearch RA can be obtained at [www.iso.org/iso/maintenance\\_agencies/](http://www.iso.org/iso/maintenance_agencies/) or directly at the JPEG home page ([www.jpeg.org](http://www.jpeg.org)).

## 5 Example Real Distributed Image Search&Retrieval System

As a proof of concept, now we will describe an example real system that we have developed at the Distributed Multimedia Applications Group (DMAG, UPC BarcelonaTech). This experience is of special interest to the scope of this paper

because it consists on a real implementation of the most complex approach, the only one which suits the requirements of a large scale web image hosting services aggregator, i.e. the Broker approach. The software we have developed provides a centralized place for searching images from Panoramio, Picasa and Flickr. Our system is compliant with the current version of the ISO/IEC 15938-12:2008 and ISO/IEC 24800 standards. Because the system follows the Broker architectural approach, there is a subsystem that receives MPQF queries addressing meta-data in JPsearch format (extended with some image specific fields and with some EXIF fields, like the camera make and model) and rewrites them once for every metadata format that is supported (Panoramio, Picasa and Flickr). The system is split into two modules: a central application that implements a broker-based metadata aggregator and a web portal as the user front-end. Modules communicate each other by using MPQF queries.

Figure 6 shows a screenshot of the system’s web front-end. In order to enable the maximum query expressiveness, the search form can be duplicated, as it is shown in this figure, allowing launching multi-queries. In fact, they will be all assembled with an OR operand and sent to the application using a single MPQF query, but the applications MPQF interpreter will split it into several searches, one by each form, and the broker will launch them in parallel.

The screenshot shows a web search interface with the following components:

- Search Header:** A blue bar with the word "Search" in a yellow box on the left and a question mark icon on the right.
- Display Section:** A light blue bar with a question mark icon on the right.
- Form Fields:**
  - Timeout:
  - Results by page:
  - Required Fields:
  - Order by:
  - Search in:
- Multi-Form Structure:** Two identical "Basic" search sections are shown side-by-side, separated by a green plus sign and a red minus sign. Each "Basic" section has a blue header with a question mark icon.
  - Basic Section 1:**
    - Title:
    - Keywords:
    - Date:
    - Location:
    - Photo:
    - License:
  - Basic Section 2:**
    - Title:
    - Keywords:
    - Date:
    - Location:
    - Latitude:
    - Longitude:
    - Photo:
    - License:
- Search Button:** A yellow button with the text "Search!" at the bottom left.

Fig. 6. Example web form

## 6 Conclusions

This paper has analyzed the current approaches and standards for metadata interoperability in distributed image search&retrieval. From the analysis of the main metadata schemas currently used for image description we conclude that they contain significant overlappings, but their specificity and constant evolution disallows any approach relying on a unified model. We have classified the several solutions that face this problem into three different approaches, the Simple Format Metadata Aggregator, the Multiple Format Metadata Aggregator and the Broker. We conclude that only the Broker approach suits the requirements of a large scale web image hosting services aggregator, while implying more complexity and performance constraints. Both the Simple Format Metadata Aggregator and the Multiple Format Metadata Aggregator suit situations with less challenges in terms of scalability and volatility, but offer better performance and are more simple to design and manage. We have also analyzed how a distributed image search&retrieval system, independently of the approach it uses, can be implemented on top of the ISO/IEC 15938-12 and ISO/IEC 24800 standards. Finally, we have described an example real distributed image search&retrieval system which provides real time access to Flickr, Picassa and Panoramio.

**Acknowledgments.** This work has been partly supported by the Spanish government (TEC2008-06692-C02-01).

## References

1. Adali, S., Candan, K.S., Papakonstantinou, Y., Subrahmanian, V.S.: Query caching and optimization in distributed mediator systems. In: SIGMOD Conference, pp. 137–148 (1996)
2. Amann, B., Beerl, C., Fundulaki, I., Scholl, M.O.: Ontology-Based Integration of XML Web Resources. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 117–131. Springer, Heidelberg (2002)
3. Sudarshan, S., Chawathe, S.S., Hector, G.-M., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J.D., Widom, J.: The tsimmis project: Integration of heterogeneous information sources. In: IPSJ, pp. 7–18 (1994)
4. Dbtune - serving music-related rdf since (2007), <http://dbtune.org/>
5. Döller, M., Tous, R., Gruhne, M., Yoon, K., Sano, M., Burnett, I.S.: The MPEG Query Format: On the Way to Unify the Access to Multimedia Retrieval Systems. IEEE Multimedia 15(4) (2008); ISSN: 1070-986X
6. Duschka, O.M., Genesereth, M.R.: Answering recursive queries using views. In: PODS 1997: Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 109–116. ACM Press, New York (1997)
7. Schandl, B., et al.: Linked data and multimedia: The state of affairs. Multimedia Tools and Applications, 1–34 (2011)
8. Bizer, C., et al.: Dbpedia - a crystallization point for the web of data. J. Web Sem. 7(3), 154–165 (2009)



9. Gergatsoulis, M., Bountouri, L., Gaitanou, P., Papatheodorou, C.: Query Transformation in a CIDOC CRM Based Cultural Metadata Integration Environment. In: Lalmas, M., Jose, J., Rauber, A., Sebastiani, F., Frommholz, I. (eds.) ECDL 2010. LNCS, vol. 6273, pp. 38–45. Springer, Heidelberg (2010)
10. Gruhne, M., Tous, R., Döller, M., Delgado, J., Kosch, H.: MP7QF: An MPEG-7 Query Format. In: Proceedings of the 3rd International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS 2007), Barcelona, Spain, pp. 15–18 (November 2007)
11. Haas, L.M., Kossmann, D., Wimmers, E.L., Yang, J.: Optimizing queries across diverse data sources. In: Proceedings of the Twenty-third International Conference on Very Large Databases, pp. 276–285. VLDB Endowment, Athens (1997), <http://citeseer.ist.psu.edu/article/haas97optimizing.html>
12. Halevy, A.Y., Rajaraman, A., Ordille, J.J.: Querying heterogeneous information sources using source descriptions. In: Proceedings of the Twenty-Second International Conference on Very Large Databases, Bombay, India, pp. 251–262. VLDB Endowment, Saratoga (1996), <http://citeseer.ist.psu.edu/levy96querying.html>
13. ISO/IEC 15938-12:2008 Information Technology - Multimedia Content Description Interface - Part 12: Query Format (2008)
14. ISO/IEC 24800-3:2010 Information technology - JPSearch - Part 3: JPSearch Query format (2010)
15. ISO/IEC 24800-2:2011 Information technology - JPSearch - Part 2: Registration, identification and management of schema and ontology (2011)
16. The linked movie database (linkedmdb), <http://www.linkedmdb.org/>
17. Manolescu, I., Florescu, D., Kossmann, D.K.: Answering XML queries over heterogeneous data sources. In: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 241–250 (2001)
18. Deviantart webpage, <http://www.deviantart.com/>
19. Flickr webpage, <http://www.flickr.com/>
20. Panoramio webpage, <http://www.panoramio.com/>
21. Photobucket webpage, <http://photobucket.com/>
22. Picasa webpage, <http://picasa.google.com/>
23. ISO/IEC 15938 Version 2. Information Technology - Multimedia Content Description Interface, MPEG-7 (2004)
24. Seligman, L., Mork, P., Halevy, A., Smith, K., Carey, M., Chen, K., Wolf, C., Madhavan, J., Kannan, A., Burdick, D.: Openii: an open source information integration toolkit. In: ACM Int. Conf. on Management of data (SIGMOD), pp. 1057–1059 (2010)
25. Tous, R., Delgado, J.: Advanced Meta-Search of News in the Web. In: 6th International ICC/IFIP Conference on Electronic Publishing (elPub 2002), Karlovy Vary, Czech Republic (November 2002)

# A Distributed Architecture for Flexible Multimedia Management and Retrieval

Mihaela Brut, Dana Codreanu, Stefan Dumitrescu,  
Ana-Maria Manzat, and Florence Sedes

Universite de Toulouse – IRIT – UMR 5505,  
31062 Toulouse, France  
{Firstname.lastname}@irit.fr

**Abstract.** Developing an efficient system that manages distributed multimedia content supposes to minimize resource consumption while providing the most relevant results for a user's query in the shortest time. This paper presents LINDO, a generic architecture framework for distributed systems that acquires efficiency in multimedia indexing and retrieval. Three characteristics particularize it: (1) it differentiates between implicit algorithms executed over all the multimedia content at the acquisition time, and explicit algorithms, executed on demand for answering a specific need; (2) it stores and processes multimedia content and metadata locally, instead of transferring and indexing it on a central server; (3) it selects a set of relevant servers for query execution based on the user query semantic processing and on the system knowledge, including descriptions of distributed servers, multimedia content and indexing algorithms. The paper relies on a concrete implementation of the LINDO framework in order to validate this contribution.

## 1 Introduction

The development of a distributed multimedia system must balance the efficiency principle to minimize resource consumption while providing the most relevant results for a user's query in the shortest time. The system needs to locate relevant multimedia contents in an environment that consists of an increasing number of machines with different capabilities, each hosting large multimedia collection. The efficient content indexation is a key issue for the management and retrieval of relevant information. Indexing is based on a set of algorithms, which generate diverse and heterogeneous multimedia metadata, but which are usually highly resources consuming.

Designing a distributed multimedia system requires a number of choices [1]: indexing based on a fixed or variable set of algorithms, algorithms executed over the entire multimedia collection or only over a filtered sub-collection, the whole set of algorithms being itself filtered or not before their effective execution, indexing in a distributed manner, on the same location as the content, or in a centralized one, by transferring the content to an indexation server (e.g., Web services), a decision regarding the distributed or the centralized placement of the multimedia metadata.

The LINDO<sup>1</sup> project (Large scale distributed INDEXation of multimedia Objects), specifies a generic architectural solution that guides the design and the development of any distributed multimedia information system relying on indexing facilities. The paper illustrates how LINDO differentiates from other multimedia distributed systems by capitalizing and improving the state of the art results concerning the above mentioned decisions. Thus, three characteristics of the LINDO framework are elicited:

- it differentiates between implicit algorithms executed over all the multimedia content at acquisition time, and explicit algorithms, executed on demand for answering a specific need;
- it processes multimedia content locally, instead of transferring and indexing it on a central server;
- it selects a set of relevant servers for query execution based on the user query semantic processing and on the system knowledge, including descriptions of distributed servers, multimedia content and indexing algorithms.

In this way, the solution we adopted in LINDO prevents from executing at once all indexing algorithms by defining a method that determines the relevant set of algorithms for a user's query. These algorithms are executed only on a multimedia sub-collection, which is also selected according to the query. Indexing algorithms will only be run on the multimedia content location. Algorithms and multimedia content filtering is done with respect to a developed centralized knowledge repository. This repository gives an overview of the system, including semantic descriptions of the distributed servers and indexing algorithms functionalities, as well as summaries of the multimedia metadata extracted and stored on each remote server. It also enables the selection of the relevant remote servers where the user's query will be executed.

Similar approaches adopted by distributed multimedia systems are exposed in Section 2. The architecture, as well as the content indexation and retrieval mechanisms of these systems are presented, while emphasizing the characteristics that differentiate them from LINDO. The LINDO framework is described in Section 3 through its generic architecture, as well as through its indexing and querying mechanisms. A testing implementation of the LINDO system is presented in Section 4, also detailing the architecture topology and the indexing and retrieval mechanisms. Finally, conclusions and future work directions are provided.

## 2 Related Work

The requirements to design an information system that manages distributed multimedia contents are:

- R1. *Fixed or variable set of indexing algorithms (IA) for multimedia contents indexation;*
- R2. *Algorithms executed at acquisition time or at user's query;*
- R3. *Selection of algorithms or not, according to the user's query;*

---

<sup>1</sup> <http://www.lindo-itea.eu/>

- R4. *Distributed executing*, in the same location as the multimedia contents storage, or *centralized*, on an indexation server where the multimedia contents are transferred;
- R5. *Filtering multimedia content* or *not* before indexing;
- R6. *Management of multimedia metadata* obtained as results of indexing process in *distributed way*, on each server that stores multimedia content, or in *centralized one*, through a unique metadata collection;
- R7. At the query moment, *selection or not of the relevant remote servers (RS)* according to the query, in order to only send the query to these servers.

In the design of the LINDO framework we considered all these aspects after a careful study of the existing state of the art. Systems and approaches in which multimedia contents are distributed adopt various techniques to accomplish content indexing and retrieval. A large part of these approaches addresses only partially the above mentioned issues according to their main objective.

The CANDELA project (Content Analysis and Network DELivery Architectures)<sup>2</sup> proposes a generic distributed architecture for video content analysis and retrieval [2]. Multiple domain specific instantiations are realized (e.g., personal mobile multimedia management [3], video surveillance [4]). The indexation is done on the distributed servers at acquisition time. The resulting metadata can be distributed over the network. However, the indexation algorithms are a priori selected and pre installed.

The KLIMT project (Knowledge InterMediation Technologies) [5] proposes a Service Oriented Architecture middleware for easy integration of heterogeneous content processing applications over a distributed network. The indexing algorithms are considered as web services. The query is limited to pre-defined patterns that match a set of rules for the algorithms' execution sequence. After such a sequence selection, the content is analyzed and the metadata is stored in a centralized database.

The WebLab<sup>3</sup> project proposes an integration infrastructure that enables the management of indexation algorithms as web services in order to be used in the development of multimedia processing applications [6]. These indexing services are handled manually through a graphical interface. For each specific application a fixed set of indexing tools is run. The obtained metadata is stored in a centralized database.

The VITALAS<sup>4</sup> project (Video & image Indexing and retrieval in the Large Scale) capitalizes the WebLab infrastructure in a distributed multimedia environment [7]. The architecture enables the integration of partner's indexation modules as web services. The multimedia content is indexed off-line, at acquisition time, on different indexing servers. No selection of indexing algorithms based on user query is done.

In [8], the authors propose a system that implements a scalable distributed architecture for multimedia content processing. The architecture is service oriented allowing the integration of new indexing tools. The indexation is distributed and the metadata produced are attached to the multimedia document.

In order to avoid the transfer of the multimedia content in the context of a distributed search engine, [9] propose to use mobile agents that migrate from one server to another for indexing the content. The resulted metadata can be either

<sup>2</sup> <http://www.hitech-projects.com/euprojects/candela>

<sup>3</sup> <http://weblab-project.org/>

<sup>4</sup> <http://vitalas.ercim.org/>

centralized or distributed over the network. In the latter case, a user's query is sent to all the remote servers. The authors prove that transferring the indexing algorithms at the content location is more efficient than transferring the content to a central indexing facility.

**Table 1.** A comparative overview of some representative systems

System name	Set of IA	IA execution moment	IA selection	IA execution location	MM content filtering	Metadata management	RS selection
Candela	Fixed	Acquisition time	Not done	Distributed servers	Not done	Distributed DB	Not specified
KLIMT	Variable	Query moment	Done	Indexation servers	Not specified	Centralized DB	Not done
Weblab	Fixed	Query moment	Done manually	Indexation servers	Not specified	Centralized DB	Not done
Vitalas	Variable	Acquisition time	Not done	Indexation servers	Not specified	Distributed DB	Not specified
[8]	Variable	Acquisition time	Not specified	Distributed servers	Not done	Distributed	Not specified

[10] propose to store on a central server a hierarchy of interest concepts that describe the content stored in the distributed servers. This hierarchy is used to select the servers that are relevant to a query. It is a priori constructed and maintained manually. The authors prove that sending the query to some servers only answers faster than sending the query to each server, while the same precision is maintained.

As can be noticed, each information system for distributed multimedia management considers only a part of the above mentioned issues that contribute to the overall system efficiency. This is the reason why the LINDO framework was developed such as to provide solutions for each issue.

Further we present the LINDO framework and explain how it provides support for acquiring efficiency for the mentioned requirements.

### 3 The LINDO Framework Architecture

The LINDO project's idea was not to define yet another multimedia information indexing solution but rather to reuse existing indexing frameworks into a common architecture. As illustrated latter, this architecture was designed such as to provide efficient solutions to the mentioned issues in order to enable reduced resource consumption and to enhance the context for giving relevant results to the user query.

We have defined the LINDO generic architecture over two main components: (1) *remote servers* (§3.1) which acquire, index and store multimedia contents, and (2) a *central server* (§3.2) which has a global view of the overall system. Even though our proposal is based on this classical approach for distributed systems, it presents two advantages. First, each remote server is independent, i.e., it can perform uniform as well as differentiated indexations of multimedia contents. For instance, some remote servers may index in real time acquired multimedia contents, while others may

proceed to an off-line indexation. Secondly, the central server can send relevant indexation routines or queries to relevant remote servers, while the system is running.

In the following, the role of each framework’s component in the fulfillment of the requirements R1 to R7 mentioned in the beginning of Section 2 is presented.

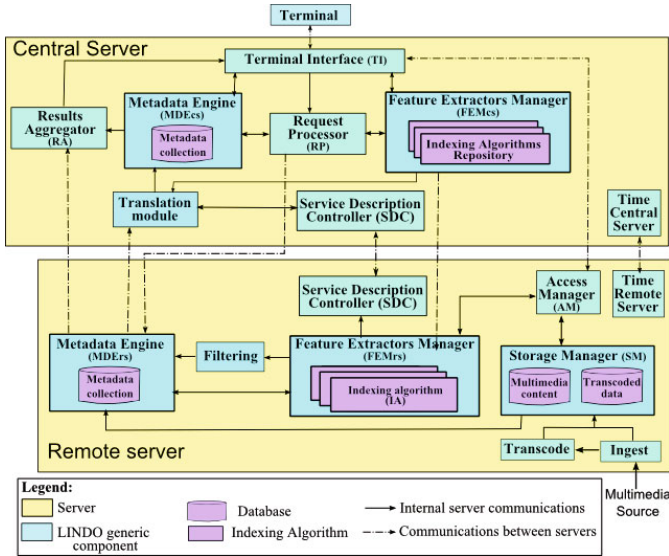


Fig. 1. LINDO Framework Architecture

### 3.1 The Remote Server Components

The remote servers in LINDO-based systems store and index all acquired multimedia contents, to provide answers to user queries. Hence, several modules have been defined and linked together in order to cover all these tasks:

- The *Storage Manager (SM)* stores the acquired multimedia contents. Through the *Transcode* module, acquired multimedia contents can be transcoded into several formats. This allows a user to download different encodings of the desired content.
- The *Access Manager (AM)* provides methods for accessing multimedia contents stored in the SM. Apart from accessing an entire content, different fragments of one multimedia content can be selected (for multimedia filtering, in case of R5).
- The *Feature Extractors Manager (FEMr)* is in charge of managing and executing a set of indexing algorithms over the acquired multimedia contents. At any time, new algorithms can be uploaded into this module, while others can be removed or updated. It can permanently run the algorithms over all the acquired contents or it can execute them on demand only on certain multimedia contents (thus enabling the deployment of the necessary algorithms for a user query for R1 and R2).
- The *Time Remote Server* handles time synchronization with the central server.
- The *Metadata Engine (MDEr)* collects and aggregates all extracted metadata about the multimedia contents stored in the SM. Naturally, the metadata stored in

this module can be queried in order to retrieve some desired information (thus, the distributed management of metadata is enabled for R6).

- The *Service Description Controller* (SCD) stores the remote server description, e.g., its location, its capacities, the acquisition context (useful for enabling the remote servers selection for R5 and R7).

### 3.2 The Central Server Components

The central server can control the remote indexing processes, and it can answer or forward user queries. Thus, a central server is composed of the following components:

- The *Terminal Interface* (TI) enables a user to specify queries and displays the obtained results. Other functionalities are included in the TI, such as visualization of metadata collections and management of indexing algorithms (thus a variable set of indexing algorithms is possible for R1).
- The *Metadata Engine* (MDEcs) gives a global view of the system. It can contain some extracted metadata about multimedia contents, some contextual information about the system, the remote servers' descriptions, the descriptions of the available indexing algorithms, etc. It is a system knowledge repository that enables efficient solutions for multiple issues: algorithm selection according to a user query (for R2 and R3); filtering of the multimedia content for R5; distributed metadata management for R6; the selection of relevant remote servers for R7.
- The *Feature Extractors Manager* (FEMcs) manages the entire set of indexing algorithms available in the system. This module communicates with its equivalent on the remote server side in order to install new indexing algorithms if it is necessary or to ask for the execution of a certain indexing algorithm on a multimedia content, or part of multimedia content. Thus, the management of a variable algorithms set is possible for R1. Their remote deployment and execution is also possible for R4.
- The *Request Processor* (RP) treats some queries on the MDEcs or forwards them to specific remote server metadata engines. Moreover, through the FEMcs, it can decide to remotely deploy some indexing algorithms. Thus, the RP has an essential contribution to the content filtering and to the selection of the relevant algorithms and remote servers for R3, R5 and R7.
- The *Results Aggregator* (RA) aggregates the results received from all the queried metadata engines and sends them to the TI, which displays them.
- The *Translation* module homogenizes the data stored into the MDEcs coming from the MDErs, the remote SDCrs and the FEMcs. Hence, this module unifies all descriptions in order to provide the system global view.
- The *Time Central Server* provides a unique synchronization system time.
- The *Service Description Controller* (SDCcs) collects all remote server descriptions (useful for enabling the remote servers selection for R5 and R7). It manages the integration in the system of new remote servers, their removal and the change in their functioning state (e.g., if the server is temporarily down or it is active).

### 3.3 Indexing and Querying Mechanisms

In order to reduce resource consumption, the architecture allows the multimedia contents indexation to be accomplished at acquisition time (i.e., *implicit indexation*) and on demand (i.e., *explicit indexation*). This avoids executing all indexing algorithms at once (thus a solution for R2 issue is available).

When a remote server acquires new multimedia content, the SM stores it and then the FEMrs starts its implicit indexation by executing a predefined set of indexing algorithms. This algorithm set is established according to the server particularities.

Once the execution of an indexing algorithm is achieved, the obtained metadata is forwarded to the Filtering module. The filtered metadata is then stored by the MDErs in its metadata collection. In order to avoid the transmission of the whole collection of metadata computed on the remote servers, the MDErs only sends, at a given time interval, a summary of these metadata to the Translation Module on the central server. [11] (the distributed metadata management is adopted for R6). Once translation is done, the metadata are sent to the MDEcs to be stored and further used in the querying process. Thus, the *implicit indexation* process is achieved.

The *query process* begins with the query specification through the TI. The user's query is sent to the RP module in order to be executed over the metadata collections. In this process, the RP analyses the query in order to select, based on the metadata summaries from MDEcs, the active remote servers that could provide answers to the query. Among the servers that were not thus selected there could be some servers that contain relevant information, but that has not been indexed with the suitable algorithms (the servers' selection relies upon their metadata summary, obtained mainly from the implicit algorithms' metadata; so, maybe among these algorithms there are not the most relevant for the current query). For this reason, our solution detects such supplementary algorithms [12] and starts their execution (i.e., *explicit indexation*) on a sub-collection of multimedia contents (developing thus efficient solution for R3 and R5). The query is sent for execution to all the selected servers. The top-ranked relevant results obtained from these remote servers are sent to the RA, which combines them in order to obtain a global ranked results list that is displayed to the user in the TI.

An important remark is that the two kinds of indexation can be mixed in the LINDO system, i.e., on some remote servers only the implicit indexation can be accomplished, while on others only the explicit indexation is done, and finally on others both indexation processes can be performed. The implementation of these two workflows will be detailed in the next section.

## 4 LINDO System Evaluation

As illustrated before, the LINDO framework was conceived to provide support for efficient handling of all the seven design requirements. The aim of the project was to build a real system, so we could evaluate our framework in different scenarios.

We further present the topology of the system employed in the evaluation. We will also illustrate with some examples how the multimedia indexing and the query processes are flexibly accomplished on this topology:



- (1) Multimedia indexing is performed locally, on each remote server, while being coordinated at the central server level;
- (2) Explicit indexing is employed only when necessary, namely when a query doesn't receive satisfactory results. Thus, for a certain query, all the suitable results are located and retrieved.

We will emphasize, while presenting this concrete implementation, how all the seven issues receive an efficient solution.

### 4.1 The LINDO System Topology Used for Evaluation

In the development of the testing system architecture, we considered multiple remote servers, located in different countries that instantiate modules of the proposed architecture, and that concern different domains (video surveillance, broadcast).

The topology of the LINDO testing system is composed of a central server and three remote servers, located in Paris and in Madrid. Two remote servers are dedicated to video surveillance and they store, index and query video contents acquired in real time. The third remote server manages multimedia contents for the broadcast domain.

In the following, we detail the particularities of each architecture module instantiation on each one of these servers, either remote or central.

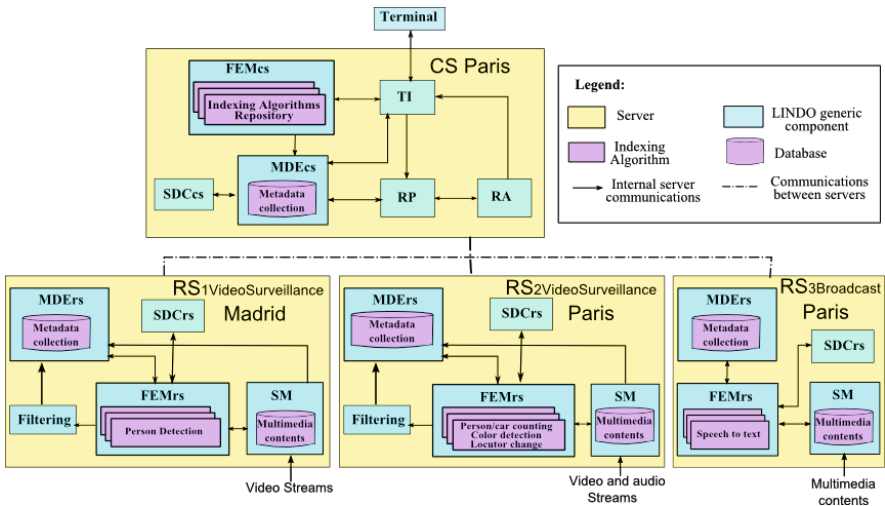


Fig. 2. The LINDO testing system topology

The generic architecture of a remote server was instantiated for each one of the three remote servers. The instantiations maintained the architecture's modules, while adopting a different implementation of their functionalities:

- For the SM module, a proprietary software developed in C language by one of the partners was adopted for a video surveillance remote server as well as for the

broadcast remote server; for the other video surveillance remote server, a software produced by another partner was employed. It manages the splitting, naming and storing manner of the multimedia content (Thales CCTV, WiLix).

- Similarly, two different implementations (in Java and C#) of the *FEM* module were adopted for the two video surveillance remote servers, while the broadcast and the central server employed the Java implementation;
- The same *MDE* module was integrated in all the three remote servers. This module was developed in Java and uses the XML native Oracle Berkley DB XML<sup>5</sup> database for storing the metadata;
- The *Filtering* module was not included in the broadcast remote server;
- A Java implementation of the *Service Description Controller* was instantiated on each remote server and on the central server as well.



This topology proves that the LINDO architecture enables each partner to develop his own implementation of each module, while respecting the interfaces and data format requirements.

The characteristics of each remote server in terms of multimedia content and implicit indexing algorithms are presented in the following.

**First video surveillance remote server, installed in Paris:**

- Manages multimedia contents acquired from two video surveillance cameras situated in a train station and watching the main hall and parking.
- Stores audio and video contents acquired in real time in the *SM* module, which in this case is the software developed in C language.
- Contains implicit indexing algorithms managed by the *FEMrs*. The indexing algorithms (executed on Windows and Linux environments) for video content are in charge with person and car counting and intrusion detection for indoor and outdoor environments, as illustrated in Table 1. For audio content, the speaker change detection is available.

**Table 2.** Indexing algorithms for video content on the Paris video surveillance remote server

	<b>Indoor</b>	<b>Outdoor</b>
<b>Intrusion</b>	- Presence of people	- Presence of people & vehicles
<b>Counting</b>	- Number of people - Main color of the upper part of the people	- Number of people, number of vehicles - Main color of the people upper part. - Main color of vehicles
		

<sup>5</sup> <http://www.oracle.com/technology/products/berkeley-db/xml/index.html>

- Handles the metadata provided by the indexing algorithms in a uniform XML data format [13] as well as the descriptions of the installed indexing algorithms. All this information is stored by the *MDErs*. A fragment of the person detection indexing algorithm description is shown in Table 2.

**Table 3.** XML algorithm description

```

<AlgorithmModel AlgoName="Person Detection" MediaType="Video">
  <InputParameters>
    <InputParamFileFormat>xml</InputParamFileFormat>
  </InputParameters>
  <ImageParameters/>
  <Feature>Local Semantic Features</Feature>
</InputParameters>
  <OutputObject Type="Metadata">
    <MetadataObject>
      <MetadataObjectDescription>location of the detected persons</MetadataObjectDescription>
    </MetadataObject>
  </OutputObject></AlgorithmModel>

```

- The *SDC* module contains an XML based description of the specific characteristics and context for this remote server (e.g., the IP address, the deployed indexing algorithms, the spatial topology of the location, the installed cameras and their characteristics). An example of such description is provided in Table 3.

**Table 4.** Remote server description

```

<RemoteServer id="rs1" name="Remote Server 1">
  <localisation>train station, Paris, France</localisation>
  <description>Manages content from cameras located in the main hall of the station and in the parking of the station</description>
  <devices>
    <camera id="c1Paris"> <description>located in the main hall </description> </camera>
  </devices>
  <indexingAlgorithms>
    <indexingAlgorithm id="ia2rs1" name="pedestrian detection" mediaType="video">
      <description>Detects pedestrians in a parking and their predominant color</description>
    </indexingAlgorithm>
  </indexingAlgorithms>
</RemoteServer>

```

**The second video surveillance remote server, installed in Madrid:**

- Manages and stores video contents acquired in real time from a video surveillance camera situated at the entrance into a security control room, using a software produced by a local partner;
- Contains an indexing algorithm for person and color detection in indoor environments. The description of this algorithm and its output follow the same formats as the algorithms installed on the Paris remote server;

- The *SDC* stores locally the description of the remote server, which is similar with the one provided in Table 3.

**The third remote server**, designed for the broadcast domain:

- Stores video content resulted from BBC journals using the same software for the SM as the video surveillance remote server from Paris;
- Contains a speech-to-text indexing algorithm based on Microsoft technology, which processes the audio stream of video files. The output of this algorithm follows the metadata format defined in the project.

The **Central Server** complies with the architecture presented in Figure 1 and has the following characteristics:

- FEMCs manages the indexing algorithm global collection where, alongside with all the implicit indexing algorithms installed on the remote servers, a supplementary set of explicit indexing algorithms are installed (abandoned luggage detection [14], shape detection, color detection, shout detection, etc. [15])
- MDECs manages multiple data: descriptions of each remote server, abstracts of the multimedia metadata from each remote server, descriptions of indexing algorithms.
- Contains also the TI, the RP and the RA modules.

## 4.2 Multimedia Indexing

The indexing algorithms enumerated above for each remote server are implicit indexing algorithms that are selected according to each server's characteristics. These algorithms index all the multimedia contents at the acquisition time. For example, on the remote server from Madrid only the algorithm for person detection in indoor environments is installed because it is a priori enough for processing the video captured with a camera at the entrance of a security control room. On the contrary, all the indexing algorithms presented in Table 1 are necessary on the video surveillance remote server in Paris because this server manages content from the main hall and parking of a train station.

These implicit video indexing algorithms produce metadata for each video frame. In order to reduce the size of the generated metadata, the *Filtering* module aggregates the metadata associated with consecutive frames that refer to the same detection. For example, for the Paris server, Table 4 contains the metadata obtained after the Filtering process was applied on the metadata generated by the person detection algorithm.

**Table 5.** Metadata aggregation result

```
<document src="stream1">
  <video capturedBy="cam1_Paris">
    <object type="Person" id="0">
      <localisation confidence="100">
        <period start_time="2010-07-28T11:07:35" end_time="2010-07-28T11:07:55"/>
        <area>control room</area>
      </localisation>
      <property name="color">red</property>
    </object>
  </video> </document>
```

Periodically, a Web service sends to the central server a metadata summary that contains the essential detected information on each remote server (thus the R6 is handled). In our experiments we send this abstract each hour. Because we are dealing with multimedia contents from two different domains (i.e., video surveillance and broadcast) the metadata summary is built differently, according to the domain:

- for video surveillance: the summary consists in statistics based on the metadata obtained in the last hour of recording;
- for broadcast: the summary will be also accomplished on the metadata that was generated by the indexation of the multimedia content during the last hour, but will consist in the titles and participants for each broadcast content (article, show, etc.).

This summary is concatenated to the other information in the MDEcs, and thus a complete view of the system is obtained on the central server. This overview is the basis for further treatment of the user’s query, in the context of explicit indexation, as detailed in the next section.

### 4.3 Query Processing

The user formulates the query in the TI through a graphic interface that enables him to specify five query components: the query itself (as free text), the location (free text), time span (calendar-based), domain (checkbox list) and the media format (video, image, audio or text).

As further detailed, the query is processed in order to select the remote servers (according R7) that are currently in a functional state (active), the sub-set of explicit indexing algorithms (R3), as well as the sub-collection of the multimedia content (R5). Figure 3 shows the logical steps that happen when a user queries the system.

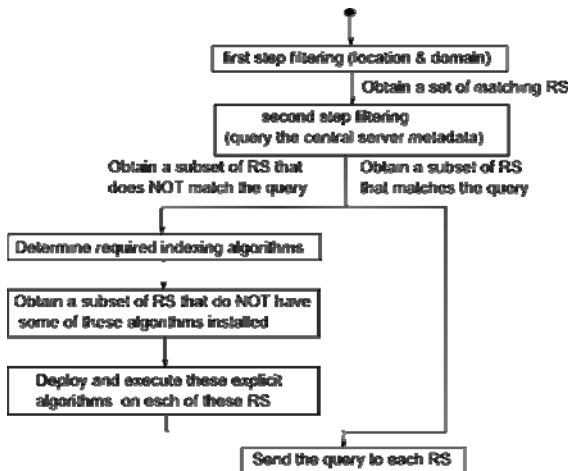


Fig. 3. Query processing diagram

In order to process the query, the first task of the RP is to select a set of active nodes on which the query will be executed. The selection is done into two steps. In

the first step every node that does not match for its location and domain with the user query (location and domain fields) is rejected. The set of remaining nodes then goes into the second filter. In this step, the user's query is applied to the metadata summary stored on the central server that corresponds to all of these remote servers. Two subsets of nodes will result, the first containing nodes that match the query, and the second one containing the remaining nodes that do not match the query.

The query is then directly sent to the first set of matching nodes. For the nodes in the second set, a list of relevant explicit algorithms is first determined. The required algorithms are found by means of similarity between each algorithm description and the user query. For each remote server, only the algorithms that have not been applied will be deployed and executed. This ensures that these nodes will also be able to provide a final answer to the query.

For illustrating these filtering operations during querying processing, we examine in the following the query mechanism on some concrete query examples.

*Q1. Location: Paris; Domain: broadcast; Time: 14 July 2010; Query content: Sarkozy speech.*

The first filtering step will select only the broadcast remote server from Paris (location is matched directly). During the second filtering step, the query content is searched inside the metadata summaries for the date of 14 July 2010 on the central server (the text "Sarkozy speech" is matched over these metadata, based on a semantic processing as will be presented at Q3). Supposing this search is successful, the query is further sent to the metadata collection from the Paris broadcast server. Based on this search, the concrete corresponding audio and video BBC news are located and provided as results.

*Q2. Domain: video surveillance; Time: 8 March 2011; Query content: woman in red.*

The first filtering step will select the two video surveillance remote servers, from Paris and Madrid. During the second filtering step, the query content is searched inside the metadata summaries for the date of 8 March 2011 from the central server (the text "woman in red" is matched over the metadata summaries corresponding to the 8 of March, according to the semantic processing described at Q3). Supposing the both remote servers confirm the existence of such information, the query is sent to Paris and Madrid. The returned results are merged by the *RA* and presented to the user via the *TI*. It can be noticed that in both Q1 and Q2, the right branch of the diagram represented in Figure 3 is followed.

*Q3. Location: Paris; Domain: video surveillance; Time: 8 March 2011; Query content: abandoned bag by women in red.*

After first filtering, the Paris server is selected. In the second filtering step, the metadata summaries are queried (using the same technique as described below), but no result is obtained. This means that either no results actually exist, or on the Paris server the algorithms that detect persons and static objects have not been executed.

We have to determine the appropriate algorithms to be run based solely on the user's query. For this purpose, an analysis is first performed on the query to obtain so-called query chunks (usually, a chunk is a noun phrase composed of a noun and its

modifiers). A shallow parsing of the query will obtain two distinct chunks: “abandoned bag” and “women in red”. Then, for each chunk, plural nouns are inflected (“women”->”woman”).

Next, the query chunks are matched to every algorithm description available on the central server, which are themselves pre-processed in the same way. The matching is done separately for every query chunk. For example, the algorithm that has the description “stationary left or abandoned luggage or object” will match query chunk “abandoned bag” due to the exact adjective match “abandoned” and to the fact that “luggage” matches “bag” because both have “container” as hypernym. The chunk-chunk match score is obtained by computing similarity between words, using the JCN - Jiang and Conrath [16] similarity measure between their synonyms and also by matching adjectives’ and adverbs’ synsets using WordNet<sup>6</sup>. JCN was chosen among other similarity measures because of its better performance [17]. The same applies for the person detection algorithm having the description “person and color detection algorithm” that matches to the query chunk “women in red” due to the fact that “woman” is a “person” and “red” is a “color” (direct hypernym relations mean good JCN score). The final score for each algorithm is the sum of the highest similarity scores between algorithm chunks and query chunks. This avoids score imbalance due to variable algorithm description lengths. The top candidate algorithms are chosen.

A check is performed to see if they had already run on the selected remote server. If every candidate algorithm has already run, that means that the initial search in the metadata on the central server yielded correctly no results. In our case, the algorithm that detects stationary objects was not run, and it is deployed for processing on the remote server.

## 5 Conclusions

In this paper we presented a framework that supports the design of an efficient distributed multimedia system by minimizing resource consumption while providing the most relevant results in the shortest time.

This framework was developed in the context of the LINDO project, and acquires efficiency in multimedia indexing and retrieval through three particularities: (1) it differentiates between implicit and explicit indexation; (2) it processes multimedia content locally, instead of transferring and indexing it on a central server; (3) it selects a set of relevant servers for query execution. The paper presented also a concrete implementation of the LINDO framework, which validates this contribution.

In the future, we will study how the LINDO indexing and retrieval mechanisms could be applied on some existing multimedia distributed repositories in order to intelligently handle their knowledge. In order to improve these mechanisms, we also plan to develop semantically enhanced algorithm descriptions that will enable to define better criteria for algorithm selection.

**Acknowledgments.** This work has been supported by the EUREKA project LINDO (ITEA2 – 06011).

---

<sup>6</sup> <http://wordnet.princeton.edu/>

## References

1. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, Hardcover (2011)
2. Petkovic, M., Jonker, W.: Content-Based Video Retrieval, A Database Perspective. Multimedia Systems and Applications, vol. 25 (2003)
3. Pietarila, P., Westermann, U., Järvinen, S., Korva, J., Lahti, J., Löthman, H.: CANDELA - storage, analysis, and retrieval of video content in distributed systems. In: The IEEE International Conference on Multimedia and Expo. pp. 1557–1560 (2005)
4. Merkus, P., Desurmont, X., Jaspers, E.G.T., Wijnhoven, R.G.J., Caignart, O., Delaigle, J.-F., Favoreel, W.: Candela- Integrated storage, analysis and distribution of video content for intelligent information system. In: European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology (2004)
5. Onan, V., Ferran, I., Joly, P., Vasserot, C.: KLIMT: Intermediations Technologies and Multimedia Indexing. In: International Workshop on Content-Based Multimedia Indexing, pp. 11–18 (2003)
6. Giroux, P., Brunessaux, S., Brunessaux, S., Doucy, J., Dupont, G., Grilheres, B., Mombrun, Y., Saval, A.: Weblab: An integration infrastructure to ease the development of multimedia processing applications. In: the 21st Conference on Software & Systems Engineering and their Applications (2008) (published online)
7. Viaud, M.-L., Thievre, J., Goeau, H., Saulnier, A., Buisson, O.: Interactive components for visual exploration of multimedia archives. In: The International Conference on Content-based Image and Video Retrieval, pp. 609–616. ACM Press, New York (2008)
8. Thong, J.M.V., Blackwell, S., Weikart, C., Hasnian, A., Mandviwala, A.: Multimedia Content Analysis and Indexing: Evaluation of a Distributed and Scalable Architecture, Technical report, HPL-2003-182 (2003)
9. Roth, V., Peters, J., Pinsdorf, U.: A distributed content-based search engine based on mobile code and web service technology. Scalable Computing: Practice and Experience 7(4), 101–117 (2006)
10. Hinds, N., Ravishankar, C.V.: Managing metadata for distributed information servers. In: The 31st Hawaii International Conference on System Sciences, pp. 513–522 (1998)
11. Laborie, S., Manzat, A.-M., Sedes, F.: Managing and querying efficiently distributed semantic multimedia metadata collections. IEEE MultiMedia Special Issue on Multimedia-Metadata and Semantic Management 16, 12–21 (2009)
12. Brut, M., Laborie, S., Manzat, A.-M., Sèdes, F.: A Framework for Automatizing and Optimizing the Selection of Indexing Algorithms. In: Sartori, F., Sicilia, M.Á., Manouselis, N. (eds.) MTSR 2009. Communications in Computer and Information Science, vol. 46, pp. 48–59. Springer, Heidelberg (2009)
13. Brut, M., Laborie, S., Manzat, A.M., Sedes, F.: A Generic Metadata Framework for the Indexation and the Management of Distributed Multimedia Contents. In: The Third International Conference on New Technologies, Mobility and Security, pp. 1–5 (2009)
14. Gasteratos, A., Vincze, M., Tsotsos, J., Mieziako, R., Pokrajac, D.: Detecting and Recognizing Abandoned Objects in Crowded Environments. In: Computer Vision Systems. LNCS, pp. 241–250. Springer, Heidelberg (2008)
15. Snoek, C.G., Worring, M.: Multimodal video indexing: A review of the state of the art. Multimedia Tools and Applications 25, 5–35 (2005)
16. Varelas, G., Voutsakis, E., Raftopoulou, P., Petrakis, E.G.M., Milios, E.E.: Semantic similarity methods in wordNet and their application to information retrieval on the web. In: the 7<sup>th</sup> International Workshop on Web Information and Data Management, pp. 10–16 (2005)
17. Jiang, J., Conrath, D.: Semantic similarity based on corpus statistics and lexical taxonomy. In: International Conference on Research in Computational Linguistics, pp. 19–33 (1997)



# Deontic BPMN

Christine Natschläger

Software Competence Center Hagenberg GmbH, Austria

[christine.natschlaeger@scch.at](mailto:christine.natschlaeger@scch.at)

[www.scch.at](http://www.scch.at)

**Abstract.** The Business Process Model and Notation (BPMN) is maintained by the Object Management Group (OMG) and a widely-used standard for process modeling. A drawback of BPMN, however, is that modality is implicitly expressed through the structure of the process flow. All activities are implicitly mandatory and whenever something should be optional, a gateway or event is used to split the process flow and offer the possibility to execute the task or to do nothing. This requires a comprehensive understanding of the whole process to identify mandatory, optional and alternative activities.

The paper addresses this issue and extends BPMN with deontic logic to explicitly highlight modality. After a detailed study of modality expressed through various BPMN elements, an approach based on path exploration is introduced to support the deontic analysis. The result is an algebraic graph transformation from BPMN to Deontic BPMN diagrams, reducing the structural complexity and allowing better readability by explicitly highlighting the deontic classification. The understandability of Deontic BPMN is studied by means of a preliminary survey.

**Keywords:** BPMN, Deontic Logic, Modality, Graph Transformation.

## 1 Introduction

The Business Process Model and Notation (BPMN) [1] is a standard maintained by the Object Management Group (OMG) and aims at business analysts and technical developers. BPMN supports the graphical representation of business processes; however, if someone wants to express normative concepts like obligations, alternatives and permissions (deontic logic), (s)he may have to split the process flow with additional BPMN elements, since an explicitly optional activity is not available. This complicates the identification of mandatory and optional activities and leads to more complex process diagrams. The following issues will be addressed within this paper:

1. Study BPMN elements that express modality.
2. Extend BPMN with deontic logic.
3. Execute deontic analysis based on path exploration approach.
4. Perform algebraic graph transformation from BPMN to Deontic BPMN.
5. Evaluate the Deontic BPMN approach based on a preliminary survey.

Before continuing with the motivation, deontic logic will be introduced. Deontic logic is defined as the logical study of the normative use of language and its subject matter is a variety of normative concepts including obligation (O), prohibition (F), permission (P) and commitment (conditional obligation) [2]. These concepts can be linked with the logical connectives for negation ( $\neg$ ), conjunction ( $\wedge$ ), disjunction ( $\vee$ ) and contravalance ( $\dot{\vee}$ ). While monadic deontic logic considers unconditional normative concepts, conditional obligations are part of dyadic deontic logic (compare [3]) in which obligations and permissions are conditional on certain circumstances. In addition, the concepts of agency and deontic logic are necessary if users and roles have to be considered (compare [4]).

## 2 Motivation

In BPMN all activities are implicitly mandatory and whenever something should be optional, a gateway or event has to be used to split the process flow and offer the possibility to execute the task or to do nothing. This approach leads to the following problems:

1. Limited readability: It is difficult to identify mandatory, optional and alternative activities at first sight.
2. Complex structure: Additional elements are necessary for expressing modality and complicate the structure of the diagram.
3. Duplication: Various roles with different normative concepts for the same activity may require duplication of the activity.

Problems with implicit modality have also been observed within an industrial project in which an order execution process was designed. In this process several activities are optional or present special cases. The resulting graphical representation is complex and a distinction of mandatory and optional activities would have been advantageous. An extract of the process is described in section 5.

After studying the related work in section 3, the first goal of ongoing research is to investigate BPMN elements that express modality and to highlight the deontic concepts (see section 4). This will reduce the structural complexity, since the number of gateways and/or sequence flows in a Deontic BPMN diagram is equal or less compared to the original BPMN diagram. As it is not possible to analyze deontic concepts directly within a BPMN diagram, section 5 presents a path exploration approach. The transformation from BPMN to Deontic BPMN is then based on algebraic graph transformation and described in section 6. Afterwards, section 7 evaluates the understandability of the approach within a preliminary survey. Finally, the conclusion sums up the main results in section 8.

## 3 Related Work

Considering deontic logic, an overview of the main applications in computer science is given by Wieringa and Meyer [5]. In addition, Broersen and van der

Torre identify ten problems of deontic logic and normative reasoning in computer science [6]. One problem is how to combine legal ontologies, normative systems, business process notations and compliance checking tools. Regarding this issue, the authors recommend the Semantics of Business Vocabulary and Business Rules (SBVR) for interaction between norms and business processes. SBVR expresses modality with alethic or deontic logic [7], however, it does neither consider the influence on the process flow (e.g., readability or reduction of structural complexity) nor the deontic analysis or transformation.

According to Goedertier and Vanthienen [8], most process modeling languages like BPMN, BPEL and UML Activity Diagrams are procedural and only implicitly keep track of why design choices have been made. Therefore, this publication presents a vocabulary for declarative process modeling that supports business concerns, execution scenario, execution mechanism, modality, rule enforcement and communication. Considering modality, procedural modeling only specifies what *must* be the case while declarative process modeling supports *must*, *ought* and *can* based on deontic logic. In [9], the same authors introduce a language to express temporal rules about obligations and permissions in business interaction called *Penelope*. The publications provide a good foundation for the current research. However, the focus of the normative concepts is more on agents and temporal constraints, whereas deontic analysis, transformation or optimization capabilities are not presented at all.

Other publications focus on the formal model of normative reasoning and deontic logic in combination with business rules and process modeling. Padmanabhan et al. consider process modeling and deontic logic in [10]. They develop a logical framework based on multi-modal logic to capture the normative positions among agents in an organizational setting. Furthermore, Governatori et al. present a language for expressing contract conditions in terms of deontic concepts called Business Contract Language (BCL) [11]. However, these publications do not provide a detailed study of modality, but rather focus on agents and their contractual relationships.

Further approaches assure business process compliance based on deontic logic. According to Sadiq et al. [12], process and control modeling are two distinct specifications, but convergence is necessary to achieve business practices that are compliant with control objectives. The authors propose a Formal Contract Language (FCL) as formalism to express normative specifications. This language is a combination of defeasible logic and a deontic logic of violations. Ghose and Koliadis present an approach to enhance business process modeling notations with the capability to detect and resolve compliance related issues [13]. They define a framework for auditing BPMN process models and suggest that activity, event and decision inclusion may be defined with deontic modalities.

Furthermore, an approach by Weigand et al. provides a bridge between interoperable transactions and business process models based on deontic logic [14].

Although several other approaches use deontic logic within process modeling, none of them studies the influence on the process flow (e.g., readability and optimization capabilities) or provides a deontic analysis and transformation.

## 4 Deontic BPMN

This section analyses the possibilities of BPMN 2.0 to express that something has to be done (Fig. 11a), can be done (Fig. 11b) or that the user can choose what to do (Fig. 11c). Some BPMN gateways and events can split the process flow and thereby offer optionality and choice. The goal of Deontic BPMN is to identify optional and alternative activities and to improve the readability of BPMN diagrams by highlighting those activities. This reduces the number of BPMN elements and allows the reader to identify at first glance what is mandatory and what is optional. This extension is called Deontic BPMN.

### 4.1 Empty Task

As a first step, an empty task (*Phi* or  $\Phi$ ) is introduced to highlight that the user has the possibility to do nothing (see Fig. 11d). This empty task is inserted whenever a sequence flow directly connects a split with a merge (see Fig. 11b).

If a *Parallel Gateway* has a *Phi-Branch*, then this branch can be removed:

$$A \wedge \Phi = A \quad A \wedge B \wedge \Phi = A \wedge B$$

All other paths that follow a *Parallel Gateway* are mandatory. In Deontic BPMN a mandatory activity is highlighted with an orange background color (or dark grey) and by surrounding the text with  $O()$  for obligatory.

If a *Phi-Task* is part of another gateway, then the deontic concept of permission or alternative (see section 4.2) is fulfilled.

### 4.2 Permission and Alternative

In BPMN, an optional (or permissible) activity as well as an alternative can be expressed with gateways (exclusive, inclusive, complex or event-based) or events (boundary events). If an outgoing path provides a *Phi-Task*, then all other paths are called optional. Otherwise, if no outgoing path provides a *Phi-Task*, then, in most cases, the outgoing paths are alternatives. The respective BPMN elements, except for *Complex Gateway* (synchronization semantics similar to *Inclusive Gateway*) and *Event-Based Gateway* (similar to *Exclusive Gateway*), are described in the following subsections.

**Exclusive Gateway:** An *Exclusive Gateway* (XOR) provides a gating mechanism where only one of the outgoing paths can be taken. If one of the outgoing paths leads to a *Phi-Task* (see Fig. 11d), then all other paths are optional:

$$A \dot{\vee} \Phi = P(A) \quad A \dot{\vee} B \dot{\vee} \Phi = P(A) \dot{\vee} P(B)$$

In Deontic BPMN an optional activity is highlighted with a green background color (or middle grey) and by surrounding the text with  $P()$  for permissible. The mentioned adaptations are compliant with the BPMN Specification [1]. If there is only one other branch beneath the *Phi-Branch*, then the surrounding gateways can be removed. The semantics of an optional task is that if a token reaches the

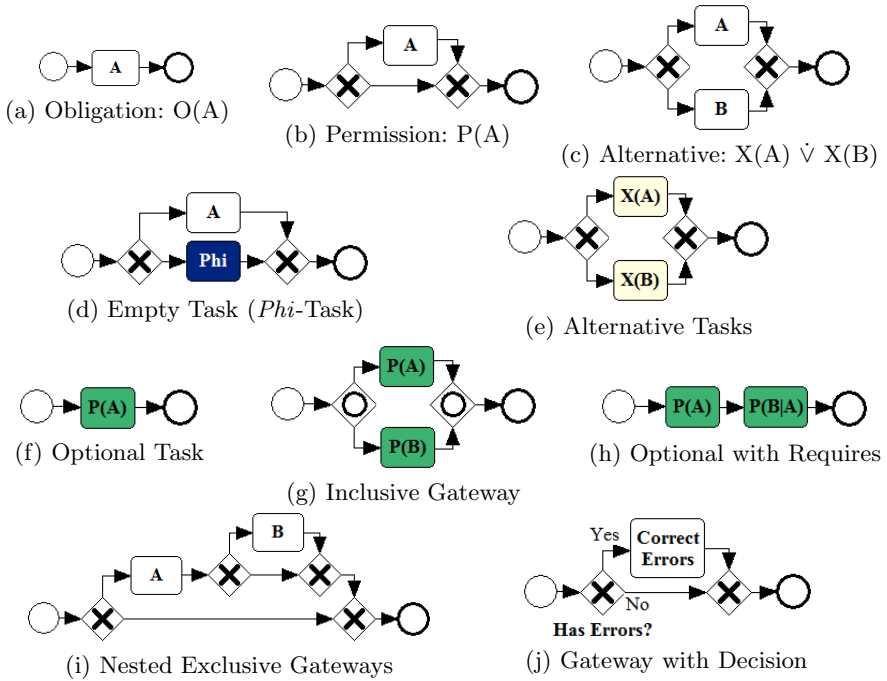


Fig. 1. BPMN and Deontic BPMN Diagrams

task it can either be executed or not. The BPMN diagram in Fig. 1d can be transformed to Fig. 1f with the same semantics.

If no outgoing path leads to a *Phi*-Task, then all paths are alternatives. The deontic logic for alternatives can be expressed as follows [15]:

$$X(A_1, A_2) = (O(A_1) \vee O(A_2)) \wedge \neg(P(A_1) \wedge P(A_2))$$

In our approach, both activities can be marked separately ( $X(A) \dot{\vee} X(B)$ ) and are highlighted with a yellow background color (or light grey) (see Fig. 1e). The semantics of an alternative task is that it is mandatory to execute the task if no alternative task was executed and forbidden if an alternative task was executed.

**Inclusive Gateway:** According to the BPMN Specification [1], after an *Inclusive Gateway* (OR) all combinations of outgoing paths may be taken, from zero to all. The BPMN Specification recommends that it should be designed that at least one path is taken, but this is not mandatory. If no outgoing path is selected, then an exception will be thrown. If at least one outgoing path is taken (e.g., default path), then an *Inclusive Gateway* can be defined as follows:

$$A \text{ OR } B \equiv (A \dot{\vee} B) \dot{\vee} (A \wedge B)$$

If one of the outgoing paths leads to a *Phi*-Task, then the structure (A OR B OR  $\Phi$ ) can be transformed as follows:

$$\begin{aligned} & \phi \dot{\vee} A \dot{\vee} B \dot{\vee} (A \wedge B) \dot{\vee} (A \wedge \phi) \dot{\vee} (B \wedge \phi) \dot{\vee} (A \wedge B \wedge \phi) \\ & \quad \phi \dot{\vee} A \dot{\vee} B \dot{\vee} (A \wedge B) \\ & \quad \quad P(A \dot{\vee} B \dot{\vee} (A \wedge B)) \\ & \quad \quad \quad P(A \vee B) \end{aligned}$$

According to theorem OK6 of [2], the last row is equal to  $P(A) \vee P(B)$ , so the result is an *Inclusive Gateway* with two optional tasks (see Fig. 1g).

If at least one outgoing path is taken but no path leads to a *Phi*-Task, then the structure (A OR B OR C) can be transformed as follows:

$$(P(A) \wedge O(A|\neg B \wedge \neg C)) \vee (P(B) \wedge O(B|\neg A \wedge \neg C)) \vee (P(C) \wedge O(C|\neg A \wedge \neg B))$$

If no default path is defined and zero outgoing paths can be taken, then the possibility of an exception has to be considered. In the following, every path has a condition, for example, if condition *C1* is true then task *A* is executed and so on. The structure (A OR B OR C) can be transformed as follows:

$$\begin{aligned} & (C1 \leftrightarrow A) \wedge (C2 \leftrightarrow B) \wedge (C3 \leftrightarrow C) \wedge ((\neg C1 \wedge \neg C2 \wedge \neg C3) \leftrightarrow \textit{Exception}) \\ & \quad (O(A|C1 \vee (\neg C2 \wedge \neg C3)) \wedge F(A|\neg C1)) \wedge \\ & \quad (O(B|C2 \vee (\neg C1 \wedge \neg C3)) \wedge F(B|\neg C2)) \wedge \\ & \quad (O(C|C3 \vee (\neg C1 \wedge \neg C2)) \wedge F(C|\neg C3)) \end{aligned}$$

If neither condition *C1*, *C2* or *C3* is true, then it is simultaneously mandatory and forbidden to execute any task. Therefore, an exception will be thrown during evaluation of the deontic tasks based on the contradictions.

**Event:** An event is something that “happens” during the course of a process [1]. Not all events lead to deontic concepts, e.g., a catching event in the process flow waits for the event and initializes no branches. If an event occurs, then several events can catch and start a new flow, however, the flows are mandatory and executed in parallel. Relevant for a deontic classification are only interrupting boundary events, since either the outgoing path of the activity or of the boundary event is taken. The deontic analysis is similar to that of an *Exclusive Gateway*.

### 4.3 Requires

In some cases, the deontic transformation of a BPMN construct needs the specification of a precondition to provide the same semantics. This precondition (also called *requires*) is defined within dyadic deontic logic [3]. The deontic notation is  $O(A|B)$  or  $P(A|B)$  (concatenation of several preconditions with  $\wedge$  or  $\vee$ ).

An example for the necessity of preconditions are nested gateways as shown in Fig. 1i. In this example task *B* is optional but can only be executed if task *A* was executed before. Therefore, task *A* is specified as precondition in Fig. 1h.

Preconditions are also necessary for situations where a user is not free to select the outgoing path. This would be the case if conditions or events are involved (*Event-Based Gateway* or *Boundary Event*). Fig. 1j, for example, shows an *Exclusive Gateway* with the decision “Has Errors?” and two outgoing flows. The first flow represents “Yes” and leads to a task “Correct Errors”. The second flow represents “No” and nothing ( $\Phi$ ) has to be done. However, the task “Correct Errors” is not optional, but mandatory under the precondition that errors exist and forbidden if no errors exist:  $(O(\text{CorrectErrors}|\text{HasErrors}) \wedge F(\text{CorrectErrors}|\neg\text{HasErrors}))$ . A pragmatic rule can further define that the forbidden part can be omitted. Note that the conditions of the example are complete (no other answer is possible) and distinct (not possible that “Yes” and “No” are simultaneously true). It is also possible to transform gateways with incomplete or overlapping conditions; however, the transformation is more complex and cannot be described in this publication due to space limitations.

Remark: The presented scenarios have one task per path. If a path has a sequence of tasks, then preconditions or sub-processes, which encapsulate the sequence of tasks and are marked according to the deontic analysis, can be used.

#### 4.4 Multiple Deontic Classifications

Multiple deontic classifications are necessary for unstructured diagrams. For example, in Fig. 2 task *A* is addressed by different split gateways. In this case, task *A* must be marked as alternative and as optional ( $X(A) \wedge P(A)$ ). The gateways of the optional structure can be removed, if preconditions are used.

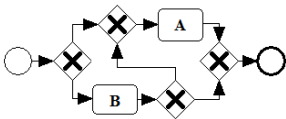


Fig. 2. Multiple Deontic Classifications

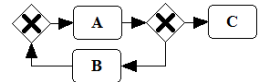
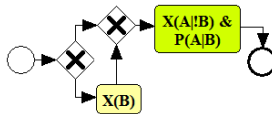


Fig. 3. Cyclic Graph

## 5 Path Exploration

A BPMN diagram is a directed cyclic graph, which can have complex structures. In path exploration all possible paths through the BPMN diagram are described within a tree structure that only includes the splits. The deontic analysis compares the paths afterwards and whenever an activity is found in every path, it is mandatory. If an activity is only found in some paths it may be optional or alternative, depending on whether the previous split has or has not a *Phi*-Task in an alternative path. In addition, references are used to cope with loops. Since path exploration duplicates the sequence flows and activities for different paths, it can neglect merging gateways and, nevertheless, reach all activities. Furthermore, this approach supports activities with multiple deontic classifications.

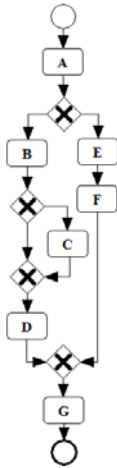


Fig. 4. Acyclic Diagram

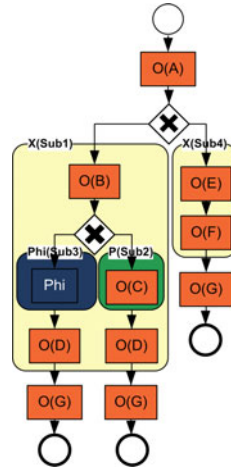


Fig. 5. Path Exploration for Fig. 4

In a first step only acyclic BPMN diagrams are studied. A simple example is shown in Fig. 4 with the resulting path exploration being displayed in Fig. 5.

Whenever a split is found in the BPMN diagram, a sub-process is used for each alternative path and marked according to the deontic logic. The sub-process ends, if a task is part of each alternative path, e.g., task *G* is mandatory in all paths, therefore, the sub-processes *Sub1* and *Sub4* end before task *G*. A sub-process can have a precondition, if it is a nested split or if the user is not free to select the outgoing paths (see section 4). If there is only one task in a sub-process, then the deontic constraint of the sub-process directly applies to the task, e.g., in the final Deontic BPMN diagram task *C* can be marked as optional and sub-process *Sub2* can be removed. A major advantage of using sub-processes is that they show a hierarchy of deontic classifications, e.g., task *C* is an optional task in an alternative path.

*Parallel Gateways* do not lead to deontic constraints since all paths are executed and, therefore, mandatory regarding deontic logic. In path exploration it is allowed to insert several paths for all possible ordering combinations, however, it is recommended to insert only one path with irrelevant order.

*Inclusive Gateways* are more complex since all combinations of paths can be taken. The different cases are distinguished according to section 4.2.

In path exploration, all paths are presented in a tree structure. Therefore, several *Start Events*, as allowed by the BPMN Specification [1], are not presentable. As a solution to this problem, a virtual *Start Event* with a splitting gateway that references all original *Start Events* is introduced. The splitting gateway must be of the same type as the merging gateway that is used to join the different branches of the original *Start Events*. In addition, BPMN allows several *End Events* within a process flow. Whenever an *End Event* finishes a branch after a split or splitting event, all other branches remain in their sub-process until they reach their own *End Event*.



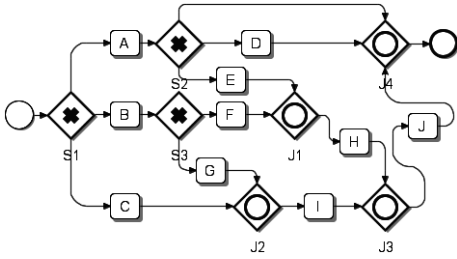


Fig. 6. Unstructured Acyclic Diagram

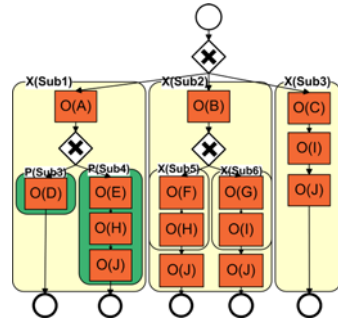


Fig. 7. Path Exploration for Fig. 6

With this approach also unstructured acyclic diagrams can be described, e.g., scenarios with merging gateways joining paths from different splits or tasks that require multiple deontic classifications (see Fig. 6). The path exploration for this example is shown in Fig. 7. Task *J* is once in an optional and several times in an alternative path thereby allowing multiple deontic classifications.

Considering cyclic graphs, iterations have no deontic influence, since the elements in between are just repeated and the only question is how often a task is executed. The only exception is a while loop (task is part of a path going backwards), which provides the possibility to execute a task 0..n times. Whenever a task in a loop might not be executed at all, this task is deontically classified similar to normal gateways as described in section 4. Considering the example shown in Fig. 3, tasks *A* and *C* are always executed and, therefore, mandatory. Task *A* might be executed 1..n times, but this has no deontic influence. However, considering the loop, a decision between tasks *B* and *C* is made until task *C* is chosen. Task *B* is not required to be executed at all and is, therefore, an alternative to task *C*. However, task *C* remains mandatory since it must be executed sooner or later.

A BPMN diagram showing an extract of the order execution process mentioned in section 2 is displayed in Fig. 8. The process comprises the following tasks: *Approve Order* (AO), *Create/Modify Appointment* (CMA), *Remove Appointment* (RA), *Approve Appointment* (AA), *Reject Appointment* (RJA), *Order in Progress* (OP) and *Execute Order* (EO). To specify all possible flows, 10 exclusive gateways and 27 sequence flows are necessary. The corresponding Deontic BPMN diagram is shown in Fig. 9 and only needs 4 exclusive gateways and 15 sequence flows to express all possible flows.

Deontic BPMN diagrams provide two advantages with respect to usability. Firstly, mandatory and optional activities can be distinguished at first sight based on the colored highlighting. Secondly, the number of gateways and sequence flows in a Deontic BPMN diagram is equal or less compared to the original BPMN diagram thereby reducing the complexity of the diagram. It is still necessary to decide whether an optional activity is executed or not but instead of describing this decision through separate gateways and alternative paths, the

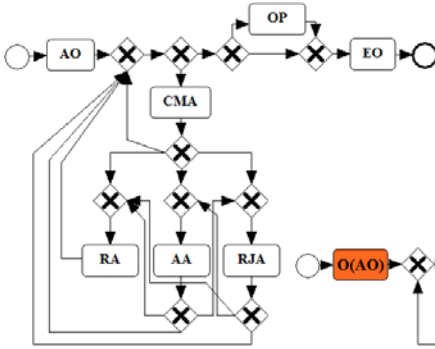


Fig. 8. BPMN Example

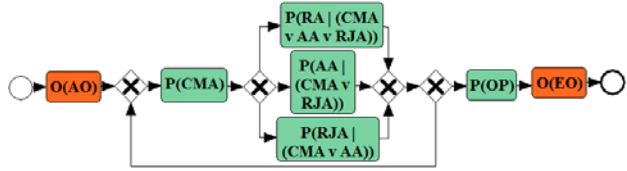


Fig. 9. Deontic BPMN based on Fig. 8

decision is described within the corresponding activity. The advantages are accompanied by additional deontic constructs requiring a basic understanding by the user. However, these constructs (e.g., preconditions) are only relevant for a more detailed understanding of the process.

## 6 Algebraic Graph Transformation

The algebraic graph transformation approach was initiated by Ehrig, Pfender, and Schneider (see [16]). The main idea of graph transformation is the rule-based modification of graphs from a source to a target graph [17]. The algebraic graph transformation approach is based on pushout constructions, where pushouts are used to model the gluing of graphs. This approach is supported by a tool called the Attributed Graph Grammar (AGG) [17].

The transformation from BPMN to Deontic BPMN (called *DeonticGTS*) is based on an algebraic graph transformation approach and specified with the help of the AGG tool. However, the transformation is currently limited to:

- structured diagrams,
- a basic set of BPMN elements (only some gateways without conditions, no intermediate events, only tasks, ...), and
- only one task per path after a gateway (also prohibiting nested gateways).

Nevertheless, *DeonticGTS* provides the most important transformations, but will be extended within further work based on the afore mentioned limitations.

In a first step, the *Type Graph* of *DeonticGTS* is specified and shown in Fig. 10. It defines a set of node and edge types as well as the generalization relationships between them. The basic element is *Node* with the derived types *Gateway*, *DeonticTask*, *BpmnTask* and *Event*. The concrete gateways (parallel, inclusive and exclusive) can either be presented as colored rectangles or as images corresponding with the BPMN element. Some *DeonticTasks* define attributes for preconditions and all are colored as suggested in section 4. The only edge

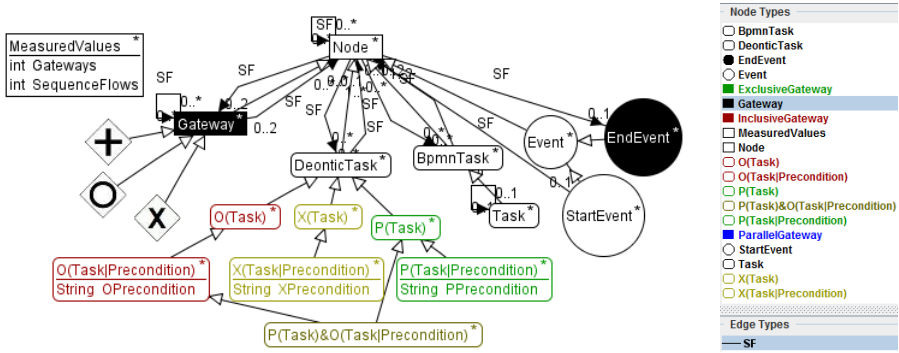


Fig. 10. Type Graph of *DeonticGTS*

type is called SF and represents sequence flows. The allowed target, source and cardinality of a sequence flow is specified by further relationships shown in the *Type Graph*. Furthermore, there is one element called *MeasuredValues* which is used to count the number of gateways and sequence flows.

Afterwards 21 transformation rules with corresponding negative application conditions are specified on four different layers. Layering offers a kind of ordering, since all rules on one layer are executed before the next layer is considered. The rules cover sequences, gateways (parallel, exclusive and inclusive) as well as iterations. Three rules concerning *Exclusive Gateways* with a *Phi*-Branch are presented in more detail and shown in Fig. III.

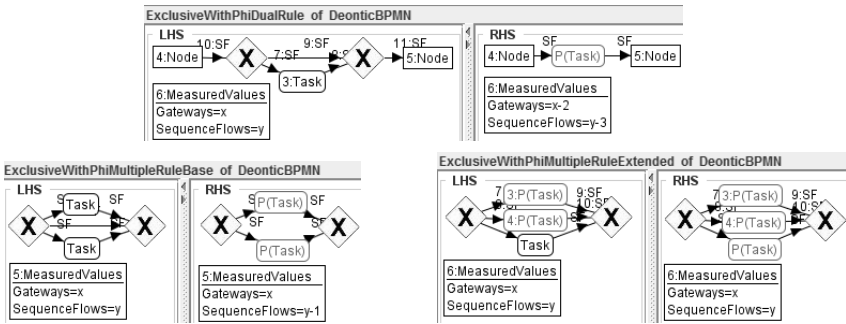


Fig. 11. Three Rules for transforming *Exclusive Gateways* with *Phi*-Branch

The first rule is called “ExclusiveWithPhiDualRule” and transforms an *Exclusive Gateway* with a task and a *Phi*-Branch into an optional task. A negative application condition forbids further alternative paths in order to avoid a violation of dangling conditions. The element *MeasuredValues* highlights that the transformation leads to a reduction of two gateways and three sequence flows.

The second rule is called “ExclusiveWithPhiMultipleRuleBase” and transforms an *Exclusive Gateway* with two tasks and a *Phi*-Branch into an *Exclusive*

*Gateway* with two optional tasks. The transformation reduces the number of sequence flows by one.

The third rule is called “ExclusiveWithPhiMultipleRuleExtended” and transforms every additional task into an optional task. This rule can be applied multiple times. If, for example, an *Exclusive Gateway* has one *Phi-Branch* and four other tasks, then the second rule is applied once and the third rule twice.

After defining the transformation rules, a graph can be created for every concrete BPMN model and is then transformed to Deontic BPMN. The AGG tool also computes the critical pairs and proves termination of the rules. This eases the proof for local confluence and, since the graph transformation system is terminating, also for global confluence.

## 7 Evaluation

The deontic artifacts are evaluated in terms of optimization and understandability based on the following methods:

*Optimization:* The optimization capabilities can be proven with algebraic graph transformation, since every rule leads to equal or less gateways and/or sequence flows and thereby reduces the structural complexity.

*Controlled Experiment:* Gemino and Wand explore differences between two modeling techniques (see [18]) and conducted an experiment with 77 participants answering problem solving questions. Interestingly, the evaluation showed that the more complex modeling technique provides better clarity. Since reduced complexity doesn’t automatically lead to better clarity, it is also necessary to study the understandability of Deontic BPMN within a controlled experiment.

The understandability of Deontic BPMN is studied within a preliminary survey, which was answered by 22 post-graduate computer scientists. The survey starts with an introduction to BPMN and Deontic BPMN including several examples as well as some pretest questions concerning the experience with (process) modeling languages in general and BPMN in particular.

Afterwards four examples are presented, each expressed with a BPMN and a Deontic BPMN model. Examples 1-3 are shown in Fig. 12, 13 and 14. Example 4 is based on the extract of the order execution process shown in Fig. 8 and 9.

To avoid a recognition of models expressing the same example, corresponding tasks received different names and the order of examples (e.g. first BPMN or Deontic BPMN diagram), elements (e.g. first parallel or exclusive gateway) and questions varies. The respondents then answered 17 questions for each model type as, for example, which tasks always have to be accomplished, or in which order the tasks can be executed.

According to Melcher et al. [19], the four aspects *concurrency*, *exclusiveness*, *order* and *repetition* are important for structural process understandability. However, the transformation to Deontic BPMN only influences the aspects *exclusiveness*, which defines whether a task is always, sometimes or never executed, and *order*. Therefore, all questions of the survey address these two aspects.

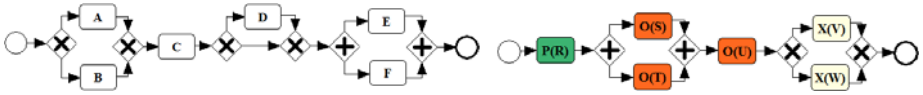


Fig. 12. Example 1

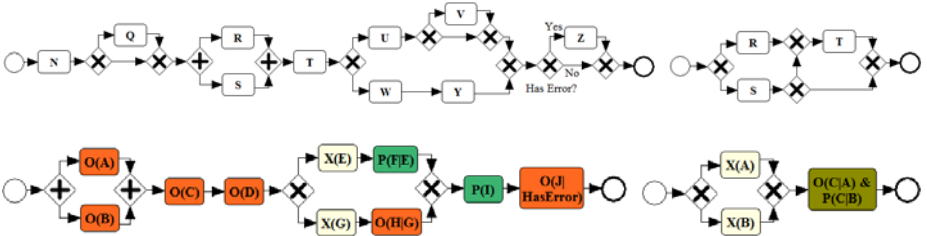


Fig. 13. Example 2

Fig. 14. Example 3

After finishing the survey, the wrong answers are divided in false-positives (FP) (answer is selected although wrong) and false-negatives (FN) (answer is not selected although true). If a question only allows a single answer (Q4-6, Q9-13), then an answer is only counted once as FP (otherwise every mistake would be doubled). An overview of all mistakes is shown in Fig. 15.

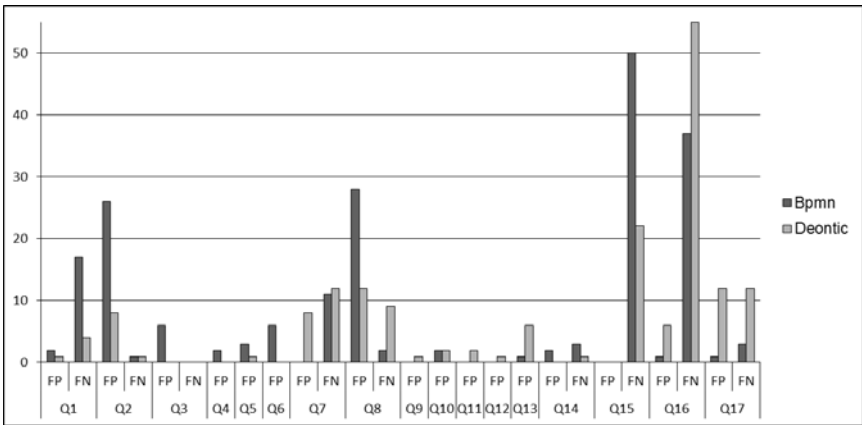
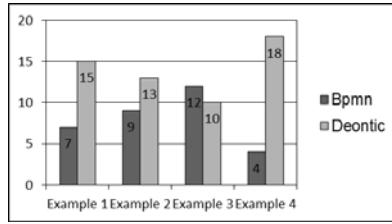


Fig. 15. Summary of Mistakes in BPMN and Deontic BPMN Diagrams

In summary, 204 mistakes emerged in the BPMN models compared to 176 mistakes in the Deontic BPMN models. So the number of mistakes could be reduced by 13.7%. However, there are two major reasons why the percentage is not higher:



**Fig. 16.** Comparison of Understandability based on Preferences

1. Known BPMN vs. unknown Deontic BPMN: According to the pretest questions most respondents have experience with BPMN but not with Deontic BPMN. Thus, several respondents mixed the colors or the deontic concepts.
2. Preconditions: The survey showed that preconditions decrease the understandability, especially in combination with multiple deontic classifications (as shown in example 3). In total, 53 mistakes were made in Deontic BPMN models due to misunderstanding of preconditions. Therefore, preconditions will be revised within further work to make them better understandable.

Considering these explanations, the results of the survey are satisfying. Furthermore, all respondents have been asked whether they prefer the BPMN or the Deontic BPMN diagram. The answers are shown in Fig. 16. The BPMN model was only favored in example 3, since this example was easy to understand in BPMN, but required multiple deontic classifications and preconditions in Deontic BPMN. In all other examples, the Deontic BPMN model was preferred, especially for more complex process flows as shown in example 4.

## 8 Conclusion

This paper presents the results of an ongoing research in which BPMN is extended with deontic logic to identify normative concepts. The Deontic BPMN diagram highlights modality, reduces the structural complexity and avoids duplication of the process flow. The automatic analysis of a BPMN diagram is based on a path exploration approach, which provides all possible paths through the BPMN diagram in a tree structure. The transformation from BPMN to Deontic BPMN is then based on algebraic graph transformation. Finally, a preliminary survey is presented to study the understandability.

The focus of further work is to revise preconditions, to extend the transformation from BPMN to Deontic BPMN and to support agent collaboration.



**Acknowledgements.** The project *Vertical Model Integration* is supported within the program “Regionale Wettbewerbsfähigkeit OÖ 2007-2013” by the European Fund for Regional Development as well as the State of Upper Austria.

## References

1. Business Process Model and Notation (BPMN) 2.0, <http://www.omg.org/spec/BPMN/2.0>
2. Åqvist, L.: Deontic Logic. In: Gabbay, D., Guenther, F. (eds.) *Handbook of Philosophical Logic*, 2nd edn., vol. 8, pp. 147–264. Kluwer Academic, Dordrecht (2002)
3. Lewis, D.: *Semantic Analyses For Dyadic Deontic Logic*. In: Stenlund, S. (ed.) *Logical Theory and Semantic Analysis*. D.Reidel Publishing Company(1974)
4. Horty, J.: *Agency and Deontic Logic*. Oxford University Press, New York (2001)
5. Wieringa, R.J., Meyer, J.-J.C.: *Applications of Deontic Logic in Computer Science: A Concise Overview*. In: *Deontic Logic in Computer Science: Normative System Specification*. Wiley, Chichester (1993)
6. Broersen, J., Van der Torre, L.: *Ten problems of deontic logic and normative reasoning in computer science*. Tutorial for ESSLLI (2010)
7. *Semantics of Business Vocabulary and Business Rules (SBVR) 1.0*, <http://www.omg.org/spec/SBVR/1.0>
8. Goedertier, S., Vanthienen, J.: *Declarative Process Modeling with Business Vocabulary and Business Rules*. In: *Proc. of Object-Role Modeling, ORM (2007)*
9. Goedertier, S., Vanthienen, J.: *Designing Compliant Business Processes from Obligations and Permissions*. In: *2<sup>nd</sup> Work. on Business Processes Design (BPD) (2006)*
10. Padmanabhan, V., Governatori, G., Sadiq, S., Colomb, R., Rotolo, A.: *Process Modelling: The Deontic Way*. In: *Asia-Pacific Conf. on Conceptual Modeling (2006)*
11. Governatori, G., Milosevic, Z.: *A Formal Analysis of a Business Contract Language*. *Int. Journal of Cooperative Information Systems (2006)*
12. Sadiq, S., Governatori, G., Namiri, K.: *Modeling Control Objectives for Business Process Compliance*. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) *BPM 2007*. LNCS, vol. 4714. Springer, Heidelberg (2007)
13. Ghose, A.K., Koliadis, G.: *Auditing Business Process Compliance*. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) *ICSOC 2007*. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
14. Weigand, H., Verharen, E., Dignum, F.P.M.: *Interoperable transactions in business models: A structured approach*. In: Constantopoulos, P., Vassiliou, Y., Mylopoulos, J. (eds.) *CAiSE 1996*. LNCS, vol. 1080, Springer, Heidelberg (1996)
15. Asirelli, P., ter Beek, M., Gnesi, S., Fantechi, A.: *A deontic logical framework for modelling product families*. In: *4<sup>th</sup> Int. Work. on Variability Modelling of Software-intensive Systems (2010)*
16. Ehrig, H., Pfender, M., Schneider, H.J.: *Graph Grammars: an Algebraic Approach*. In: *Proceedings of FOCS 1973*. IEEE, Los Alamitos (1973)
17. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: *Fundamentals of Algebraic Graph Transformation*. Springer, Heidelberg (2006)
18. Gemino, A., Wand, Y.: *Complexity and clarity in conceptual modeling: Comparison of mandatory and optional properties*. *Data&Knowledge Engineering* 55 (2005)
19. Melcher, J., Mendling, J., Reijers, H.A., Seese, D.: *On Measuring the Understandability of Process Models*. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) *BPM 2009. Lecture Notes in Business Information Processing*, vol. 43, pp. 465–476. Springer, Heidelberg (2010)

# Improving Stock Market Prediction by Integrating Both Market News and Stock Prices

Xiaodong Li<sup>1</sup>, Chao Wang<sup>2</sup>, Jiawei Dong<sup>2</sup>, Feng Wang<sup>3</sup>,  
Xiaotie Deng<sup>1,4</sup>, and Shanfeng Zhu<sup>2\*</sup>

<sup>1</sup> Department of Computer Science, City University of Hong Kong, Hong Kong  
`xiaodonli2@student.cityu.edu.hk`

<sup>2</sup> Shanghai Key Lab of Intelligent Information Processing and School of Computer Science, Fudan University, Shanghai 200433, China  
`zhusf@fudan.edu.cn`

<sup>3</sup> School of Computer and State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China

<sup>4</sup> Department of Computer Science, University of Liverpool, Liverpool, UK

**Abstract.** Stock market is an important and active part of nowadays financial markets. Addressing the question as to how to model financial information from two sources, we focus on improving the accuracy of a computer aided prediction by combining information hidden in market news and stock prices in this study. Using the multi-kernel learning technique, a system is presented that makes predictions for the Hong Kong stock market by incorporating those two information sources. Experiments were conducted and the results have shown that in both cross validation and independent testing, our system has achieved better directional accuracy than those by the baseline system that is based on single one information source, as well as by the system that integrates information sources in a simple way.

**Keywords:** Stock market prediction; Information integration; Multi-kernel learning.

## 1 Introduction

Stock market is an important and active part of nowadays financial market. Both investors and speculators in the market would like to make better profit by analyzing market information. In Efficient Market Hypothesis (EMH, proposed by Fama [1]), it is thought that stock prices have already included and revealed all the information in the market, and that random walk is the most natural and possible way the stock market should behave. However, researchers in behavioral finance argue that EMH may not be right because of irrational behavior of players who are influenced by various kinds of market information as well as their psychological interpretation of the information [2]. Although there

---

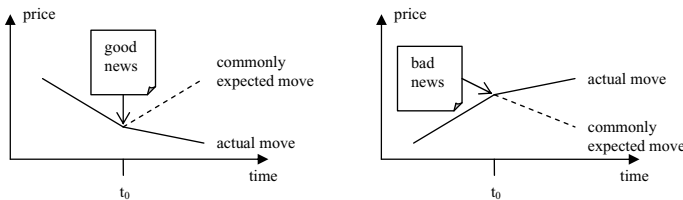
\* Corresponding author.



are differences between those two theories, neither of them ignores the effect of market information.

News articles, known as one of the most important part of market information, are widely used and analyzed by investors. With the development of Internet, both the speed of news broadcasting and the amount of news articles have been changing tremendously: 1) Bloomberg<sup>1</sup> and Thomson Reuters<sup>2</sup> could provide investors around the world with real-time market news by network; and 2) the number of online news articles could be thousands of times than that in the past newspaper-only age. With such a big volume of information, more and more institutions rely on the high processing power of modern computers for information analysis. Predictions given by support systems could assist investors to filter noises and make wiser decisions. How to model and analyze market information so as to make more accurate predictions thus becomes an interesting problem.

Researchers with computer science background have studied this problem, and some works modeled it as a classification problem [3,4,5,6]. Their algorithm could give a directional prediction (up/hold/down) based on the newly released news articles. Text classification, however, only considers news articles' impact, but ignores information hidden in the prices shortly before news is released. Taking into consideration both news articles and short-time history price, we believe that positive news may not always lead to going up the price immediately, it might just stop the price trend from falling down. Negative news does not necessarily drive the trend from up to down. Instead, it might just make price curve appear flat. Figure 1 illustrates several possible scenarios that could happen. This is different from the traditional view, that is, "good news means up, bad news means down".



**Fig. 1.** Possible scenarios of price movements based on news articles and short time history price

In order to aggregate more-than-one information sources into one system, Multi-Kernel Learning (MKL) is employed in our system. The MKL has two sub-kernels: one uses news articles and the other accepts the short-time history prices. After learning the weights for sub-kernels, the derived model gives prediction that is supposed to be more accurate than traditional methods.

<sup>1</sup> <http://www.bloomberg.com/>

<sup>2</sup> <http://thomsonreuters.com/>

The rest of this paper is organized as follows. Section 2 reviews major existing approaches related to stock market directional prediction. Section 3 gives an overview of our proposed system and brief information about experimental design. Experimental results are reported in Section 4. The conclusion and future work are given in Section 5.

## 2 Related Work

Some helpful observations and discussions about news and market prices are presented in finance domain. Ederington and Lee [7] observed that there is always a big increment of standard deviation of five-minutes returns on the day that a government announcement released at 8:30am. Engle and Ng [8] claim that positive and negative news present asymmetry impact curves, based on the empirical analysis of ARCH model family.

Analyzing news articles and market prices has also been reported in many works in computer science domain. Seo, Giampapa and Sycara [9] built a TextMiner system (a multi-agent system for intelligent portfolio management), which could assess the risk associated with companies by analyzing news articles. Fung and Yu [4] classified news articles into categories and predict newly released news articles' directional impact based on the trained model. AZFin-Text system, built by Schumaker and Chen [10], is also able to give directional forecast of prices.

As illustrated in Figure 2, the common steps of those works in general could be summarized as follows:

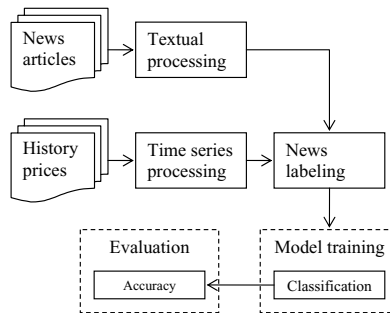


Fig. 2. Pipeline of traditional approach

1. **Representation of news articles.** News articles are basically textual documents. The vector space model from information retrieval treats textual documents as “bag of words”, where each word is in a long vector  $\langle word_1, word_2, \dots \rangle$  without any duplicate. Weights  $w$  are assigned to words (or terms, features, etc.) by measuring their term frequencies and inverse document frequencies i.e.  $w = tf \cdot idf$ . Stop words, e.g. *the*, *of* and *is*, which frequently occur but are less informative, are removed in order to reduce

the noises in corpora. However, remaining features still form a very sparse space which requires a lot of memory and computation power. To reduce the feature space, dimension reduction and feature selection methods are applied [11][12][13][14].

2. **Representation of price data.** Price data is a series of trading statistics. At each time  $t_i$ , there is a corresponding trading record  $r_i$ . History price data is of different quality: 1) Inter-day data. Generally contains open, close, high, low and volume for each trading day, and daily collected; and 2) Intra-day data. a.k.a. tick-by-tick data, the trading statistics collected in a smaller time unit, e.g. minute or second. Since price data is not smooth, time series segmentation techniques, such as the parametric spectral model [15] and fourier coefficient [16], are applied to smooth the price curve in order to emphasize the trend of prices.
3. **Alignment and news labeling.** Fung *et al.* [4] formulate the alignment of news articles and price data. They classify possible scenarios into three categories: 1) *Observable Time Lag* - a time-lag between news and price moves; 2) *Efficient Market* - No observable time-lag, price moves at almost the same time with news; 3) *Reporting* - News released after price moves, a summary or report of previous price moves. Alignment of news article with price data mainly relies on the time stamps attached with the news articles. News articles are sorted by their time stamps in ascending order and then aligned with price data in the corresponding time slots. Before training the classifier, news articles should be labeled with a tag indicating the directions of their impact. Besides simply labeling news articles by the trend of aligned price movement, linguistic methods and sentiment mining are also implemented for this purpose [17][18][19][20].
4. **Model learning.** Machine learning models, such as support vector machine, are used as classifier in this area because of their relatively short training time and comparatively high classification accuracy. Take SVM for example, SVM is fed with the labeled news articles. By selecting some points as support vectors, SVM finds a hyperplane that maximizes marginal space from hyperplane to the support vectors. After the training phase, prediction model is built up to make predictions for new coming data.
5. **Evaluation.** Besides evaluating the model by some standard benchmarks, such as precision, recall and accuracy, some researchers [3][21] conduct a preliminary simulation, which makes trades (buy/hold/sell) in a virtual market with real market data. Trading strategies are made on basis of the signals generated by prediction model. *Return rate* is calculated to measure the performance of different models. However, the performance in the simulation actually depends not only on the prediction model, but also on the trading strategies and risk management, which is beyond the discussion scope of this paper.

### 3 System and Experimental Design

Unlike the traditional way that considers only single information source, our system is designed to enable to integrate multiple information sources. The architecture of the system is shown in Figure 3.

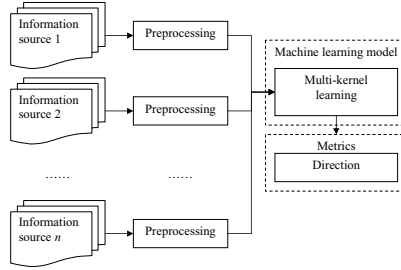


Fig. 3. Architecture of system integrating multiple sources

Two information sources are used in our system: market news and the history prices shortly before the news released, the later of which is referred to as *ex ante* prices in the rest of this paper. The processing pipeline is illustrated in Figure 4.

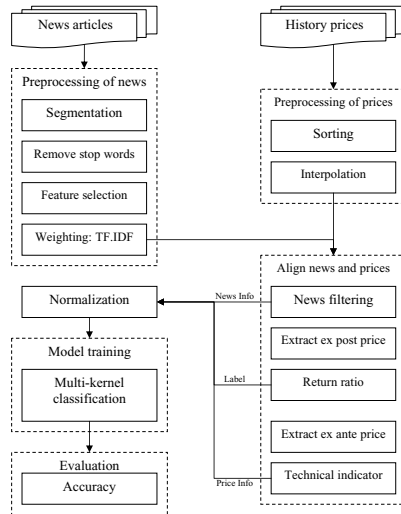


Fig. 4. Detail processing pipeline of our system

Section 3.1 describes brief information about the information sources fed into the system. Section 3.2 and Section 3.3 will talk about the processing of news

articles and history prices respectively. Section 3.4 presents how to align news and prices. Data normalization and model training are discussed in Section 3.5 and Section 3.6.

### 3.1 Information Sources

The system is designed in a way that enables to integrate two information sources, which comes from news articles and the ex ante prices. The input data should have following characteristics:

- **Time stamped.** Each news article is associated with a time stamp with proper precision indicating when the news was released. With this time stamp, the system could identify the order of news and find corresponding price information.
- **Tick based.** Tick based data means trading data is often recorded in a short interval.
- **Parallel.** Since system needs to tag news using price movement, news articles and history prices should be about the stories of the same time period.

### 3.2 Preprocessing of News Articles

News articles are regarded as raw materials that need to be preprocessed. The main steps are listed below:

1. **Chinese segmentation.** We segment the news articles by using an existing Chinese segmentation software<sup>3</sup>. Although the segmentation software could produce outputs with high quality, many words that are specific to finance domain cannot be segmented correctly. A finance dictionary is thus employed to refine the segmentation results.
2. **Word filtering.** This step actually does two things: 1) remove stop word; and 2) filter out other unimportant words (only leave representative words, such as nouns, verbs and adjectives).
3. **Feature selection.** Not all the words would be included into the final feature list. Feldman [22] (Chapter IV.3.1) selects about 10% of the words as features. Similarly, top 1000 words (out of 7052 words after filtering) with high  $\chi^2$  score are selected as features in the system.
4. **Weighting.** With the selected 1000 features, we calculate the widely used  $tf \cdot idf$  value for each word as its weight.

### 3.3 Preprocessing of History Prices

With the development of high frequency trading, tick data is popularly used so that results based on tick data are more convincing. With the following properties, tick data are distinguished from daily data:

---

<sup>3</sup> <http://ictclas.org/>

- **Big amount.** Tick data have much more records than daily data over the same time period.
- **Disorder.** Tick data is not recorded by the order of their time stamps, but by the time they arrive at the logging system.
- **Variant interval.** Time intervals between different records may not be the same. As transactions may happen in any seconds, a time interval between consecutive records is not always the same.

Raw tick price data is preprocessed through following steps:

1. **Sorting.** Since transactions do not arrive in the order of their time stamps, we must first sort the whole list of records by their time stamps.
2. **Interpolation.** Since time intervals between consecutive transactions are not the same. Over some time periods, there even does not exist any record, which leads to one problem: what price value should be filled in that time period. There are two ways to solve this problem: 1) linear time-weighted interpolation proposed by Dacorogna *et al.* [23]; and 2) nearest closing price. This method splits tick data in a minute basis and samples the closing price in each minute. If there is no record in a given minute, the closing price of last minute will be taken as the closing price of this minute. Although both methods make sense, we adopt the second method, which is simple and easy to implement.

### 3.4 Align News Articles and Market Prices

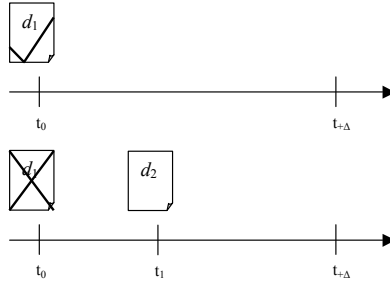
In order to train machine learning model with information from two sources, we need to prepare the raw data and change them as what the algorithm needs.

#### 3.4.1 News Filtering

(Not all the news articles are used because of two reasons: 1) *trading hour limitation.* Take Hong Kong Exchanges and Clearing Ltd (HKEx) for example, according to regulation of HKEx, 10:00am-12:30pm and 14:30pm-16:00pm are trading hours. Only news articles released during trading hours are considered to have impact on HKEx stock prices. Besides the trading hour limitations, Schumaker [21] suggests eliminate the opening 20 minutes in the morning and opening 20 minutes in the afternoon in order to absorb the impact of news that is released during the night and lunch break. 2) *news impact overlapping.* As illustrated in Figure 5, if the prediction length is  $\Delta$  and there are two news articles ( $d_1$  and  $d_2$ ) released within  $\Delta$ . In this scenario, it is hard to tell whether the change of price at time  $t_{+\Delta}$  is caused by either  $d_1$  or  $d_2$  or both. In our system,  $d_1$  will be eliminated.

#### 3.4.2 Extract and Process Ex Post Prices

Before feeding news article into machine learning model, each news should have a category tag. In high frequency trading, people would like to know what is the short-time impact of news on prices, which means people are much more caring about the price change shortly after the news released (named as *ex post*



**Fig. 5.** Example of news filtering

prices). Gidofalvi [24] shows that news impact has the biggest power 20 minutes after it’s released. Without the knowledge on how long the news impact lasts, we label the news by the change of price in future 5, 10, 15, 20, 25 and 30 minutes respectively, which can be regarded as an extension of work [21].

Corresponding to news time stamp  $t_0$ , current price value  $p_0$  in the sorted tick price series as well as the prices of future 5, 10, 15, 20, 25 and 30 minutes intervals denoted as  $p_{+5}$ ,  $p_{+10}$ ,  $p_{+15}$ ,  $p_{+20}$ ,  $p_{+25}$  and  $p_{+30}$ , respectively, can be found. if  $t_0 + \Delta$ , for example,  $t_0 + 20$ , exceeds the requirement of *trading hour limitation*, the news article will be eliminated. We convert ex post prices into the return rates by

$$R = \frac{p_i - p_0}{p_0}$$

We set a threshold of 0.3% (average transaction cost in the market), which means if  $R$  is greater than 0.3%, news is tagged as positive. In contrast, news is tagged as negative if  $R$  is less than  $-0.3\%$ .

### 3.4.3 Extract and Process Ex Ante Prices

Prices from 30 minutes to 1 minute before news is released, and sampled at 1 minute interval are extracted as ex ante prices in our experiment. However, if we naively take the 30 points as 30 features, the machine learning model will assume that the 30 features are independent from each other, which means the sequential dependency of the price serial  $p_{-30}, p_{-29}, \dots, p_{-1}$  is not preserved and the machine learning model will not be able to use the sequence information. Cao and Tay [25,26] convert the price series into RDP indicators. Following their method, we use the same formulae of RDPs which are listed in Table 1.

In addition to RDPs, we employ some other market indicators from stock technical analysis. The formulae of market indicators are listed in Table 2, where  $p_i$  is the price at minute  $i$ , and  $q$  refers to the order counted in minute.

After all, 30 ex ante price points are converted to 6 RDPs and 5 market indicators, all of which will be simply referred to as indicators in the following sections.

**Table 1.** The formulae of RDPs

RDP	Formula
RDP-5	$100 * (p_i - p_{i-5}) / p_{i-5}$
RDP-10	$100 * (p_i - p_{i-10}) / p_{i-10}$
RDP-15	$100 * (p_i - p_{i-15}) / p_{i-15}$
RDP-20	$100 * (p_i - p_{i-20}) / p_{i-20}$
RDP-25	$100 * (p_i - p_{i-25}) / p_{i-25}$
RDP-30	$100 * (p_i - p_{i-30}) / p_{i-30}$

**Table 2.** Indicator

Indicator	Formula	Description
RSI(q)	$100 * UpAvg / (UpAvg + DownAvg)$ $UpAvg = \sum_{p_i > (\sum_i p_i) / q} (p_i - (\sum_i p_i) / q)$ $DownAvg = \sum_{p_i < (\sum_i p_i) / q} (p_i - (\sum_i p_i) / q)$	Relative Strength Index
RSV(q)	$100 * (p_0 - \min_q(p_i)) / (\max_q(p_i) - \min_q(p_i))$	Raw Stochastic Value
R(q)	$100 * (\max_q(p_i) - p_0) / (\max_q(p_i) - \min_q(p_i))$	Williams Index
BIAS(q)	$100 * (p_0 - (\sum_i p_i) / q) / ((\sum_i p_i) / q)$	Bias
PSY(q)	$100 * (\sum 1\{p_i > p_{i-1}\}) / q$	Psychological Line

### 3.5 Normalization

After performing the previous steps, we have obtained: 1) group of news instances (denoted as  $N$ ); 2) group of indicator instances (denoted as  $I$ ); and 3) vector  $L$  containing labels. Each instance of  $N$  corresponds to one piece of news and each feature of instance corresponds to one selected word. Each instance in  $I$  also corresponds to one piece of news and each feature in  $I$  corresponds to one of the indicators. For features in  $N$  and features in  $I$  that only takes non-negative value, denoted as  $f_k$ , we use

$$norm(w_{ki}) = \frac{w_{ki} - \min(w_{k*})}{\max(w_{k*}) - \min(w_{k*})}$$

to normalize. The range of values after the normalization is  $[0, 1]$ . For features in  $I$  that could take both positive and negative values, denoted as  $f_m$ , we use

$$norm(w_{mi}) = \frac{w_{mi}}{\max(w_{m*})}$$

to normalize. The range of values after the normalization is  $[-1, 1]$ .

### 3.6 Model Training

We want to compare the ability of prediction between two information sources based model and single information source based model. SVM is selected to be the classifier. We implement four models that use information of news and ex ante prices in different ways. The details about the setup of those four models are described as follows:



1. **News article only.** This model takes labeled news instances as the input of SVM. It tests the prediction ability when there are only news articles. (Figure 6 (1))
2. **Ex ante prices only.** This model takes labeled price data as the input of SVM. It tests the prediction ability when there are only history prices. (Figure 6 (2))
3. **Naive combination of news article and ex ante prices.** This approach uses the simple combination of news articles and prices. Naive combination means combine the 1000 features from news and 11 features from indicators to form a 1011 feature vector. As instances of news and instances of indicators are one-one correspondence, the label for each instance is unchanged and the total number of instances remains the same. (Figure 6 (3))
4. **Multi-Kernel Learning (MKL).** MKL is employed to aggregate the information within news articles and ex ante prices of each news (SHOGUN [27], an implementation of MKL, is used in our experiment.). Unlike naive combination which trains SVM using

$$f(\vec{x}) = \text{sign}\left(\sum_{i=1}^m \alpha_i l_i \mathbf{k}_{naive}(\vec{x}_i, \vec{x}) + b\right)$$

where  $\vec{x}_i, i = 1, 2, \dots, m$  are labeled training samples of 1011 features, and  $l_i \in \{\pm 1\}$ , for the case of MKL, similarity is measured among the instances of news and instances of indicators respectively, and the two derived similarity matrices are taken as two sub-kernels of MKL (as shown in Figure 6 (4)) and weight  $\beta_{news}$  and  $\beta_{indicator}$  are learnt for sub-kernels,

$$\mathbf{k}(\vec{x}_i, \vec{x}_j) = \beta_{news} \mathbf{k}_{news}(\vec{x}_i^{(1)}, \vec{x}_j^{(1)}) + \beta_{indicator} \mathbf{k}_{indicator}(\vec{x}_i^{(2)}, \vec{x}_j^{(2)})$$

with  $\beta_{news}, \beta_{indicator} \geq 0$  and  $\beta_{news} + \beta_{indicator} = 1$ , where  $\vec{x}^{(1)}$  are news instances of 1000 features and  $\vec{x}^{(2)}$  are indicator instances of 11 features.

For training models 1, 2 and 3, we use grid search and 5-fold cross validation to find the best combination of model parameters. As for MKL, parameter selection is a little bit different. As the best parameter combination for sub-kernels of news and indicators has been found during the training of model 1 and 2, we just need to adopt the derived parameters and search the best parameters which are specific to MKL.

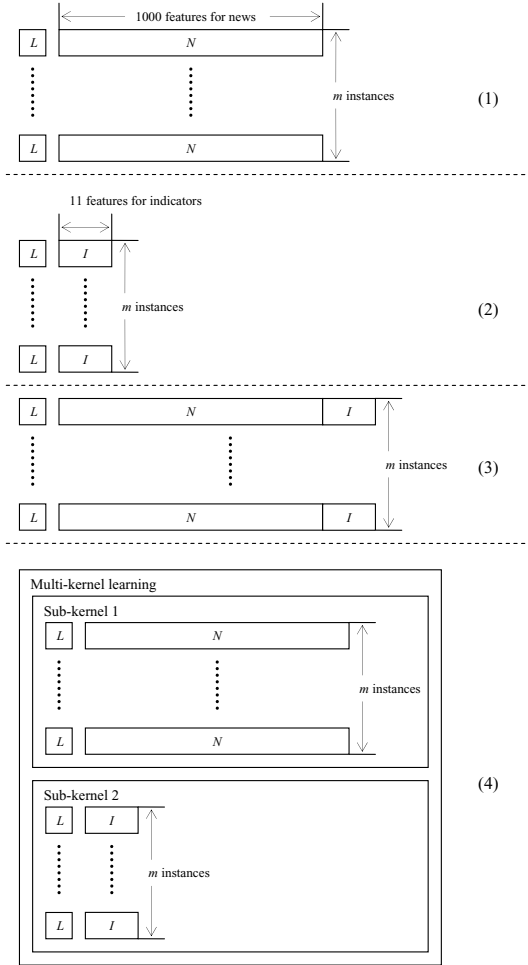
## 4 Experimental Results and Discussion

### 4.1 Data Sets

Parallel news articles and market prices serve as the experiment data sets.

- News articles. The news articles of year 2001 used in our experiment are bought from Caihua<sup>4</sup>. All the news articles are written in Traditional Chinese. Each piece of news is attached with a time stamp indicating when the news is released.

<sup>4</sup> <http://www.finet.hk/mainsite/index.htm>



**Fig. 6.** Model setup: (1) News article only; (2) Ex ante prices only; (3) Naive combination; (4) MKL

- Market prices. The market prices contain all the stocks’ prices of HKEx in year 2001.

Time stamps of news articles and prices are tick based.

HKEx has thousands of stocks and not all the stocks are playing actively in the market. We mainly focus on the constituents of Hang Seng Index<sup>5</sup> (HSI) which, according to the change log, includes 33 stocks in year 2001. However, the constituents of HSI changed twice in year 2001, which was on June 1st and July 31th. Due to the *tyranny of indexing* [28], price movement of newly added

<sup>5</sup> <http://www.hsi.com.hk/HSI-Net/>

constituent is not rational and usually will be mispriced during the first few months. We only select the constituents that had been constituents through the whole year. Thus, the number of stocks left becomes 23. The first 10-month data is used as the training/cross-validation set and the last 2-month data is used as the testing set.

## 4.2 Parameter Selection

During model training period, parameters are determined by two methods: grid search and 5-fold cross validation. Take model 1’s training for example, SVM parameters to be tuned are  $\gamma$  and  $C$ . For  $\gamma$ , algorithm searches from 0 to 10 with step size 0.2; For  $C$ , the step size is 1 and  $C$  searches from 1 to 20. Thus, there are totally  $50 \times 20 = 1000$  combinations of parameters (in other words, 1000 loops). In each loop, 5-fold cross validation is executed to validate the system’s performance, which equally splits the first 10-month data into 5 parts and use 4 of them to train the model and the left 1 part to validate. Among the 1000 combinations, the one with the best performance is preserved and used to configure the final model for testing.

For models 1, 2 and 3, the method of parameter selection is the same. For model 4, instead of changing  $\gamma$   $50 \times 50 = 2500$  times (50 for sub-kernel of news and 50 for sub-kernel of indicator), we just adopt the  $\gamma$ s which have already been selected in models 1 and 2’s training. MKL’s parameter  $C$  is selected by the same method as the other model.

## 4.3 Experimental Results

*accuracy*, which is adopted by many previous works [3,4,5,6], is used to evaluate the predication performance. The formula of accuracy is

$$accuracy = \frac{true\_positive + true\_negative}{true\_positive + false\_positive + true\_negative + false\_negative}$$

Cross validation results are listed in Table 3, and Table 4 lists the results of independent testing (numbers in bold font indicate the best results at that time point and the second best results are underlined). From the results, we can see that:

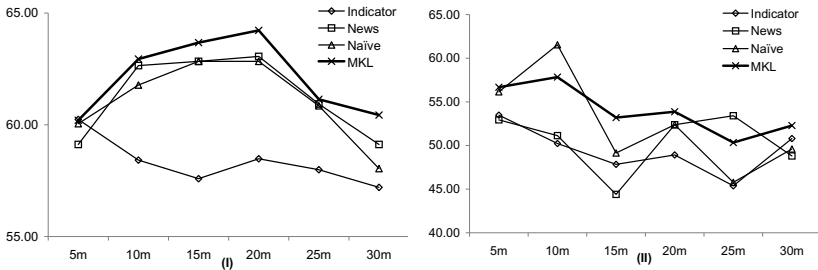
**Table 3.** Prediction accuracy of 5-fold cross validation (%)

Cross validation	5m	10m	15m	20m	25m	30m
Indicator	<b>60.24</b>	58.42	57.59	58.48	57.99	57.2
News	59.12	<u>62.65</u>	<u>62.84</u>	<u>63.06</u>	<u>60.93</u>	<u>59.12</u>
Naive combination	60.05	61.78	<u>62.84</u>	62.84	60.85	58.04
MKL	<u>60.20</u>	<b>62.94</b>	<b>63.68</b>	<b>64.23</b>	<b>61.14</b>	<b>60.44</b>

**Table 4.** Prediction accuracy of independent testing (%)

Independent testing	5m	10m	15m	20m	25m	30m
Indicator	53.48	50.23	47.84	48.92	45.38	<u>50.80</u>
News	52.94	51.13	44.40	<u>52.38</u>	<b>53.41</b>	48.80
Naive combination	<u>56.15</u>	<b>61.54</b>	49.14	<u>52.38</u>	45.78	49.60
MKL	<b>56.68</b>	<u>57.84</u>	<b>53.20</b>	<b>53.87</b>	<u>50.34</u>	<b>52.29</b>

1. MKL outperforms the other three models both in cross validation and independent testing, except for point 5m in cross validation and points 10m, 25m in testing. Although both Naive combination approach and MKL make use of market news and prices, naive combination does not outperform single information source based model as expected. The reason might be that simply combining the features of news and the features of indicator could lead to feature bias. As stated in Section 3.6, the number of news features has nearly 100 times than that of indicator. Features are thus greatly bias to the side of news. As can be observed in Figure 7 (I), the curve of *Naive* is quite close to *News*. On the other hand, due to a different learning approach, MKL balances the *predictability* of news and prices (Market news and stock prices have their own characteristics and the information hidden in either of them could be a complement to the other.). Comparing to cross validation, although MKL’s performance in independent testing decreases, it still can achieve 4 best-results and 2 second-best-results.



**Fig. 7.** Experimental results: (I) cross validation results, (II) independent testing results

2. From Figure 7 (I) and (II), it can be clearly observed that the slope of curve *Indicator* is almost negative, which means the predictability of prices decrease as time goes by. This observation is natural and consistent to the general knowledge that the impact of market information will be gradually absorbed by the market and the predictability will decrease as time goes by.

3. From Figure 7 (I), accuracy score at point 20m for curves *News*, *Naive* and *MKL*, all of which use the information of news articles, is better than

the other points, which means predictability of news articles reaches its peak at point 20m. This observation is consistent with the observation of Gidofalvi [24].

## 5 Conclusion and Future Work

In this paper, we build up a system which uses multi-kernel learning to integrate market news and stock prices to improve prediction accuracy for stock market. Experiment is conducted by using a whole year Hong Kong stock market tick data. Results have shown that multi-kernel based model could make better use of information in news articles and history prices than the model simply combines features of news articles and prices. It is also observed that multi-kernel model outperforms the models that just adopt one information source.

For future research on this topic, it is possible to further investigate this problem from two ways.

- Some news articles are usually not just talking about one specific stock but several stocks of the same industry. Instead of focusing on the constituents of HSI, industry section, which is at a higher level than individual stocks, could be a good research object.
- MKL in our system uses two information sources. More information sources could be found and added to this system. What kind of information sources could provide complementary information without redundancy is another question that is worth consideration.

**Acknowledgements.** The work in this paper is partially supported by the Research Grants Council of Hong Kong under Project No. RGC CityU 112909, the Natural Science Foundation of Hubei Province of China (No. 2010CDB08504), and Research Project of Shanghai Key Laboratory of Intelligent Information Processing under Grant No. I IPL-2010-007. Xiaotie Deng would like to acknowledge the support of a research grant from University of Liverpool.

## References

1. Fama, E.F.: The behavior of stock market prices. *Journal of business* 38(1) (1964)
2. Barberis, N., Thaler, R.: A survey of behavioral finance. *Handbook of the Economics of Finance* 1, 1053–1128 (2003)
3. Fung, G., Yu, J., Lam, W.: News sensitive stock trend prediction. *Advances in Knowledge Discovery and Data Mining*, 481–493 (2002)
4. Fung, G.P.C., Yu, J.X., Lu, H.: The predicting power of textual information on financial markets. *IEEE Intelligent Informatics Bulletin* 5(1), 1–10 (2005)
5. Wu, D., Fung, G., Yu, J., Liu, Z.: Integrating Multiple Data Sources for Stock Prediction. In: Bailey, J., Maier, D., Schewe, K.-D., Thalheim, B., Wang, X.S. (eds.) *WISE 2008*. LNCS, vol. 5175, pp. 77–89. Springer, Heidelberg (2008)

6. Wu, D., Fung, G.P.C., Yu, J.X., Pan, Q.: Stock prediction: an event-driven approach based on bursty keywords. *Frontiers of Computer Science in China* 3(2), 145–157 (2009)
7. Ederington, L.H., Lee, J.H.: How markets process information: News releases and volatility. *Journal of Finance* 48(4), 1161–1191 (1993)
8. Engle, R.F., Ng, V.K.: Measuring and testing the impact of news on volatility. *Journal of finance* 48(5), 1749–1778 (1993)
9. Seo, Y.W., Giampapa, J., Sycara, K.: *Financial news analysis for intelligent portfolio management*. Robotics Institute, Carnegie Mellon University (2004)
10. Schumaker, R.P., Chen, H.: Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Transactions on Information Systems (TOIS)* 27(2), 12 (2009)
11. Kohonen, T.: Self-organized formation of topologically correct feature maps. *Biological Cybernetics* 43(1), 59–69 (1982)
12. Kohonen, T., Somervuo, P.: Self-organizing maps of symbol strings. *Neurocomputing* 21(1-3), 19–30 (1998)
13. Ultsch, A.: Data mining and knowledge discovery with emergent self-organizing feature maps for multivariate time series. *Kohonen Maps* 46 (1999)
14. Fu, T., Chung, F.L., Ng, V., Luk, R.: Pattern discovery from stock time series using self-organizing maps. In: *Workshop Notes of KDD2001 Workshop on Temporal Data Mining*, pp. 26–29 (2001)
15. Smyth, P.J.: Hidden Markov models for fault detection in dynamic systems (November 7, 1995)
16. Pavlidis, T., Horowitz, S.L.: Segmentation of plane curves. *IEEE Transactions on Computers* 100(23), 860–870 (1974)
17. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: sentiment classification using machine learning techniques. In: *Proceedings of the ACL-2002 Conference on Empirical Methods in Natural Language Processing*, vol. 10, pp. 79–86 (2002)
18. Kim, S.M., Hovy, E.: Determining the sentiment of opinions. In: *Proceedings of COLING*, vol. 4, pp. 1367–1373 (2004)
19. Godbole, N., Srinivasaiiah, M., Skiena, S.: Large-scale sentiment analysis for news and blogs. In: *ICWSM 2007* (2007)
20. Pang, B., Lee, L.: Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* 2(1-2), 1–135 (2008)
21. Schumaker, R.P., Chen, H.: A quantitative stock prediction system based on financial news. *Information Processing & Management* 45(5), 571–583 (2009)
22. Feldman, R., Sanger, J.: *The text mining handbook* (2007)
23. Dacorogna, M.M.: *An introduction to high-frequency finance* (2001)
24. Gidófalvi, G., Elkan, C.: Using news articles to predict stock price movements. In: *Department of Computer Science and Engineering*. University of California, San Diego (2001)
25. Tay, F.E.H., Cao, L.: Application of support vector machines in financial time series forecasting. *Omega* 29(4), 309–317 (2001)
26. Cao, L.J., Tay, F.E.H.: Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks* 14(6), 1506–1518 (2004)
27. Sonnenburg, S., Rätsch, G., Henschel, S., Widmer, C., Behr, J., Zien, A., Bona, F., Binder, A., Gehl, C., Franc, V.: The SHOGUN machine learning toolbox. *The Journal of Machine Learning Research* 99, 1799–1802 (2010)
28. Ritter, J.R.: Behavioral finance. *Pacific-Basin Finance Journal* 11(4), 429–437 (2003)

# Querying Semantically Enriched Business Processes

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

**Abstract.** In this paper we present a logic-based approach for querying business process repositories. The proposed solution is based on a synergic use of an ontological framework (OPAL) aimed at capturing the semantics of a business scenario, and a business process modelling framework (BPAL) to represent the workflow logic. Both frameworks are grounded in logic programming and therefore it is possible to apply effective reasoning methods to query the knowledge base stemming from the fusion of the two. A software platform has been developed and the first tests are encouraging.

**Keywords:** Business Process, Semantic Annotation, Query Language.

## 1 Introduction

In recent years there has been an acceleration towards new forms of cooperation among enterprises, such as networked enterprises, where the resources and Business Processes (BPs) of the participating organizations are integrated to pursue shared objectives in a tightly coordinated fashion, operating as a unique (virtual) organization. In particular, building global BPs (i.e., cross-enterprise processes) by assembling existing local BPs found in different enterprises is not an easy operation, since the semantic interoperability problem arises both at a data level and at a process level. The local BPs are often built by using different tools, according to different business logics, and using different labels and terminology to denote activities and resources. To overcome this incompatibilities, the various participating enterprises need to agree on a common view of the business domain (e.g., represented by a reference ontology), and provide descriptions of the local BPs according to such an agreed common view.

Much work has been done<sup>1</sup> towards the enhancement of BP management systems [1] by means of well-established techniques from the area of the Semantic Web and, in particular, computational ontologies [2]. An enterprise ontology supports unambiguous definitions of the entities occurring in the domain, and eases the interoperability between software applications and the reuse/exchange of knowledge between human actors.

---

<sup>1</sup> See, e.g., the SUPER (<http://www.ip-super.org/>), COIN (<http://www.coin-ip.eu/>) and PLUG-IT (<http://plug-it.org/>) initiatives.

In this frame, we focus on the problem of querying repositories of semantically annotated BPs. The proposed solution is based on a synergic use of an ontological framework (OPAL [3]) aimed at capturing the semantics of a business scenario, and a business process modelling framework (BPAL [4]) to represent the workflow logic. Then, the semantic annotation of BPs w.r.t. ontologies allows us to query BPs in terms of the ontology vocabulary, easing the retrieval of local BP (or process fragments) to be reused in the composition of new BPs. Figure 1 depicts a birds-eye view of the querying approach, with the local BP repositories (LBPR<sub>x</sub>), the common set of ontologies and vocabularies (Reference Ontology) used for the semantic annotation ( $\Sigma_x$ ) of the BP repositories, and the query engine operating on the above structures.

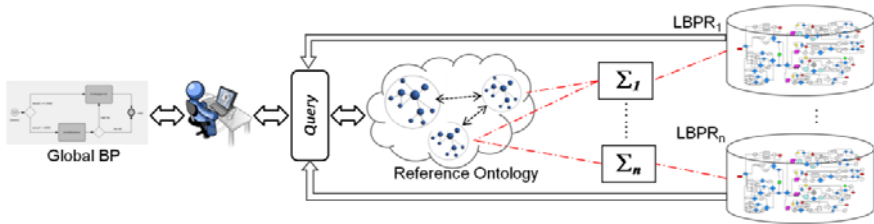


Fig. 1. Business Process Querying Approach

The proposed approach provides a uniform and formal representation framework, suited for automatic reasoning and equipped with a powerful inference mechanism supported by the solutions developed in the area of Logic Programming [5]. At the same time it has been conceived to be used in conjunction with the existing BP management tools as an ‘add-on’ to them, by supporting BPMN [6] and in particular its XPDL [7] linear form as a modeling notation and OWL [8], for the definition of the reference ontologies.

## 2 Knowledge Representation Framework

In this section we introduce the knowledge representation framework which is at the basis of the querying approach that will be proposed in Section 3. In this framework we are able to define an *Enterprise Knowledge Base* (EKB) as a collection of logical theories where: *i*) the representation of the *workflow graph* associated with each BP, together with its *behavioral semantics*, i.e., a formal description of its execution, is provided by a BPAL specification; *ii*) the representation of the *domain knowledge* regarding the business scenario is provided through an OPAL ontology.

### 2.1 Introducing BPAL

BPAL [4] is a logic-based language that provides a declarative modeling method capable of fully capturing the procedural knowledge in a business process. Hence it provides constructs to model activities, events, gateways and their sequencing. For branching flows, BPAL provides predicates representing *parallel* (AND), *exclusive*



(XOR), and *inclusive* (OR) *branching/merging* of the control flow. A BPAL BP Schema (BPS) describes a workflow graph through a set of *facts* (ground atoms) constructed from the BPAL alphabet. In Figure 2 an exemplary BPS modeled in BPMN is depicted, together with the corresponding BPAL translation.

In order to perform several reasoning tasks over BPAL BPSs, three core theories have been defined, namely the meta-model theory *M*, the trace theory *TR* and the dependency constraint theory *D*.

*M* formalizes a set of structural properties of a BPS, that at this level is regarded as a labeled graph, to define how the constructs provided by the BPAL language can be used to build a *well-formed* BPS. Two categories of properties should be verified by a well-formed BPS: *i) local* properties related to the elementary components of the workflow graph (for instance, every activity must have at most one ingoing and at most one outgoing sequence flow), and *ii) global* properties related to the overall structure of the process (for instance, in this paper we assume that processes are *structured*, i.e., each branch point is matched with a merge point of the same type, and such branch-merge pairs are also properly nested).

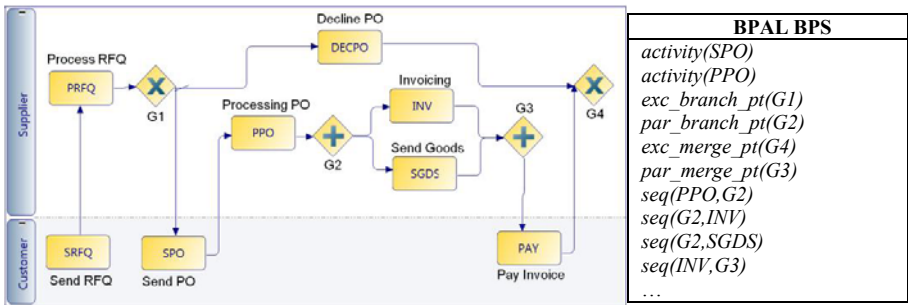


Fig. 2. BPMN eProcurement Process (left-side), partial BPAL translation (right-side)

*TR* provides a formalization of the trace semantics of a BP schema, where a *trace* models an execution (or instance, or enactment) of a BPS as a sequence of occurrences of activities called *steps*.

*D* is introduced for the purpose of efficiently verifying properties regarding the possible executions of a BPS. *D* defines properties in the form of constraints stating that the execution of an activity is dependent on the execution of another activity, e.g., two activities have to occur together (or in mutual exclusion) in the process (possibly, in a given order). Examples of such constraints are *i) precedence(a,b,p,s,e)*, i.e., in the sub-process of *p* starting with *s* and ending with *e*, if *b* is executed then *a* has been previously executed; *ii) response(a,b,p,s,e)*, i.e., in the sub-process of *p* starting with *s* and ending with *e*, if *a* is executed then *b* will be executed. In a structured BPS, like the ones considered in this paper, such constraints could be verified by an exhaustive exploration of the set of correct traces. However, this approach would be inefficient, especially when used for answering complex queries of the kind described in Section 3. Thus, we follow a different approach for defining the constraint patterns discussed in [9] by means of logic rules that infer the absence of a counterexample (e.g., in the

*response* case, a correct trace that does not lead, from a step of activity *a*, to a step of *b*). The set of these rules constitutes the theory *D*. This approach is indeed more efficient because, in order to construct a counterexample, we can avoid to actually construct all possible interleavings of the traces generated by the execution of parallel sub-processes and, in fact, we only need to perform suitable traversals of the workflow graph.

### 2.2 Semantic Annotation through a Business Reference Ontology

For the design of a *Business Reference Ontology (BRO)* to be used in the alignment of the terminology and conceptualizations used in different BP schemas, we consider as the reference framework the OPAL methodology [3]. **OPAL** organizes concepts through a number of meta-concepts aimed at supporting the domain expert in the conceptualization process, identifying active entities (*actors*), passive entities (*objects*), and transformations (*processes*). OPAL concepts may be defined in terms of concepts described in an ontology (or set of ontologies) describing a specific domain (or set of domains). Then the BRO is composed by an OPAL model linked to a set of domain ontologies, that can be already existing resources or artifacts developed on purpose.

The *Semantic Annotation*  $\Sigma$  defines a correspondence between elements of a BPS and concepts of a BRO, in order to describe the meaning of the former through a suitable conceptualization of the domain of interest provided by the latter in terms of related *actors*, *objects*, and *processes*.  $\Sigma$  is specified by the relation  $\sigma$ , which is defined by a set of assertions of the form  $\sigma(El, C)$ , where *El* is an element of a BPS and *C* is an OPAL concept.

Technically, the language adopted for the definition of a BRO is a fragment of OWL, falling within the OWL-RL profile. OWL-RL, is an OWL subset designed for practical implementations using rule-based techniques. In the *EKB*, ontologies are encoded using the triple notation by means of the predicate  $t(s,p,o)$ , representing a generalized RDF triple (with subject *s*, predicate *p*, and object *o*). For the semantics of an OWL-RL ontology we refer to the axiomatization (OWL 2 RL/RDF rules) described in [8].

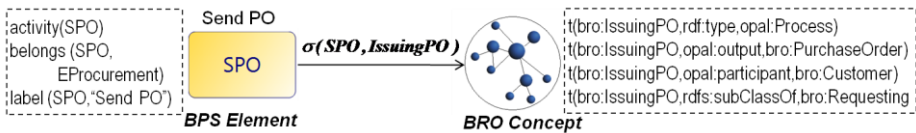


Fig. 3. Semantic Enrichment of Process Schemas

Figure 3 reports an example of semantic annotation related to the eProcurement process of Figure 2, where a basic definition in terms of *inputs*, *outputs* and related *actors* is provided for *IssuingPO* (we assume the usual prefixes *rdfs* and *owl* for the RDFS/OWL vocabulary, plus *opal* for the introduced vocabulary and *bro* for the specific example).

### 3 Querying an Enterprise Knowledge Base

An *EKB* is formalized by a First Order Logic theory, defined by putting together the theories introduced in the previous section:

$$EKB = BRO \cup OWL\_RL \cup \Sigma \cup M \cup B \cup TR \cup D$$

where: *i*)  $BRO \cup OWL\_RL \cup \Sigma$  represents the *domain knowledge*, i.e., **BRO** is an OPAL Business Reference Ontology, encoded as a set of triples of the form  $t(s,p,o)$ ; **OWL\_RL** is the OWL 2 RL/RDF rule set, included into the *EKB* to support reasoning over the *BRO*; and  $\Sigma$  is a semantic annotation, including a set of assertions of the form  $\sigma(El,C)$ ; *ii*)  $M \cup B$  represents the *structural knowledge* about the business processes, i.e., **M** is the meta-model theory and **B** is a *repository* consisting of a set of BP schemas defined in BPAL; *iii*)  $TR \cup D$  is a formalization of the *behavioral semantics* of the BP schemas, i.e., **TR** is the trace theory and **D** is the theory defining the dependency constraints.

A relevant property of the *EKB* is that it has a straightforward translation to a logic program [5], which can be effectively used for reasoning within a Prolog environment. This translation allows us to deal within a uniform framework with several kinds of reasoning tasks and combinations thereof. Every component of the *EKB* defines a set of predicates that can be used for querying the knowledge base. The reference ontology **BRO** and the semantic annotation  $\Sigma$  allow us to express queries in terms of the ontology vocabulary. The predicates defined by the meta-model theory **M** and by the BP schemas in **B** allow us to query the schema level of a BP, verifying properties regarding the flow elements occurring in it (*activities, events, gateways*) and their relationships (*sequence flows*). Finally **TR** and **D**, allow us to express queries about the behavior of a BP schema at execution time, i.e., verify properties regarding the execution semantics of a BP schema.

In order to provide the user with a simple and expressive query language that does not require to understand the technicalities of the logic engine, we propose *QuBPAL*, a query language based on the SELECT-FROM-WHERE paradigm (see [10] for more details) that can be translated to logic programs (where nested and disjunctive queries are translated to multiple rules) and evaluated by using the XSB engine (<http://xsb.sourceforge.net>). More specifically, *QuBPAL* queries which do not involve predicates defined in **TR**, i.e., queries that do not explicitly manipulate traces, are translated to logic programs belonging to the fragment of Datalog with stratified negation. For this class of programs the tabling mechanism of XSB guarantees an efficient, sound and complete top-down evaluation.

As an example, below we report a *QuBPAL* query and its corresponding Datalog translation. We prefix variables names by a question mark (e.g.,  $?x$ ) and we use the notation  $?x::Conc$  to indicate the semantic typing of a variable, i.e., as a shortcut for  $\sigma(x,y) \wedge t(y,rdfs:subClassOf,Conc)$ , in order to easily navigate the ontology taxonomy.

```

SELECT <?p,?s,?e>
WHERE activity(?s::bro:Requesting) AND belongs(?b::bro:FinancialTransaction,?p,?s,?e) AND
precedence(?a::bro:Invoicing,?b,?p,?s,?e)

```

q(P,S,E):- t(C\_1,rdfs:subClassOf,bro:Requesting),t(C\_2,rdfs:subClassOf,bro:FinancialTransaction),  
 t(C\_3,rdfs:subClassOf,bro:Invoicing),σ(S,C\_1),σ(B,C\_2),σ(A,C\_3),belongs(S,P),belongs(E,P),  
 belongs(A,P,S,E),belongs(B,P,S,E), wf\_subproc(P,S,E),precedence(A,B,P,S,E).

This query returns every well-formed process fragment (i.e., structured block) that starts with a *requesting* activity and that contains a *financial transaction* preceded (in every possible run) by an *invoicing*. The *SELECT* statement defines the output of the query evaluation, which in this case is a process fragment identified by the triple  $\langle ?p, ?s, ?e \rangle$ , where  $?p$  is a BP identifier,  $?s$  is the starting element, and  $?e$  is the ending element. The query may include a *FROM* statement (absent in the above example), indicating the process(es) from which data is to be retrieved (possibly the whole repository). In the *WHERE* statement it can be specified an expression which restricts the data returned by the query, built from the set of predicates defined in the *EKB*, the = predicate and the connectives AND, OR, NOT with the standard logic semantics. If we consider the process fragment of Section 2.1, the answer to the above query contains the sub-process starting with SPO and ending with PAY.

This query shows the interplay of the different components of the *EKB*: the notions of well-formed process fragment (*wf\_subproc*) and containment (*belongs*) are formalized in the BPAL meta-model theory, *precedence* is a dependency constraint regarding the behavioral semantics of the BPS,  $\sigma$  and  $t$  are defined in terms of the semantic description of the domain specified in the BRO.

## 4 Implementation

A prototype of the proposed framework has been implemented as a Java application, interfaced with the XSB logic programming engine through the Interprolog library (<http://www.declarativa.com/interprolog>). The BPAL reasoner is depicted in Figure 4. On the left part of this figure, enclosed in a dotted rectangle, we have grouped the components involved in the *setup phase*, when the *EKB* is built.

The process repository *B* is populated by process schemas modeled by business experts using a BPMS capable of exporting XPDL, that is translated into BPAL by means of the module *XPDL2BPAL*. The business reference ontology *BRO* is imported from an OWL-RL ontology by the module *OWL2LP* that translates the *BRO* into a set of ground facts in the triple notation. The reasoning over the ontology is supported by the rule-set *OWLRL*, obtained by a translation of the OWL 2 RL/RDF rules. The semantic annotation  $\Sigma$  is encoded as an OWL file too, and it is similarly imported into the *EKB*. The parsing of OWL files is based on the Jena2 toolkit (<http://jena.sourceforge.net/>). Finally the *EKB* is completed by the logic programs encoding the meta-model theory *M*, the trace theory *TR* and the dependency constraints *D*. Having populated the *EKB*, the reasoning tasks are performed by querying the knowledge base through *QuBPAL* queries that are translated into Datalog by the module *QBPAL2LP* and evaluated by the XSB engine. The computed results can be exported through the *XpdlWriter* module as an XPDL file, for its

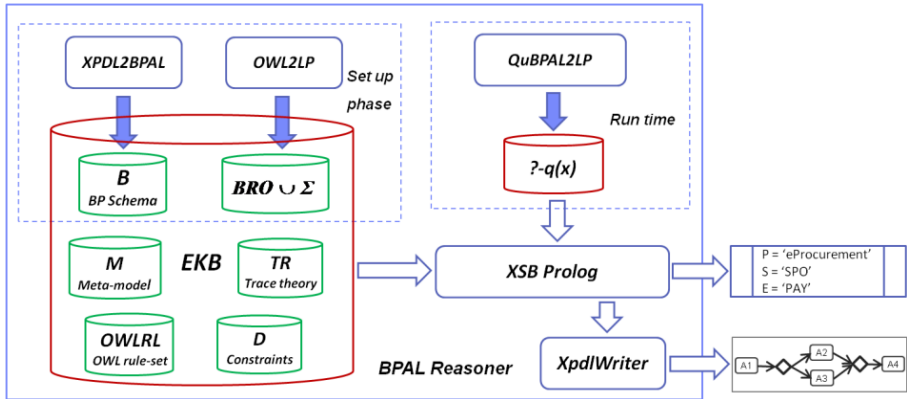


Fig. 4. Architecture of the BPAL Reasoner

visualization in a BPMS and its further reuse. These components are enclosed in a dotted rectangle on the right part of Figure 4.

We conducted in [10] a preliminary evaluation of the system performance on a desktop machine (Intel Core2 E4500 CPU (2x2.20 GH), 2GB of RAM), to show the feasibility of the approach. In particular, the rule-based implementation of the OWL reasoner and the effective goal-oriented evaluation mechanism of the Prolog engine shown good response time and significant scalability. The results are summarized in Table 1. Timings are expressed in seconds and represent the average value over 10 runs. We generated artificial XPDL files, describing three BP repositories, **T1-T3** of different size and structure. In the first part of Table 1 we report, for each repository, the number of BPs, the total size, i.e. the total number of flow elements, the total number of gateways and the size of the smallest and biggest BP. As Business Reference Ontology we created an eProcurement ontology (about 400 named concepts described by about 2500 triples), by including part of the OWL translation of the SUMO ontology (<http://www.ontologyportal.org/translations/SUMO.owl>). In particular, we used the *Process* hierarchy introduced in SUMO as root for the activity taxonomy (about 250 concepts) adopted for the random annotation of the generated BPs. First, we tested the set up phase (middle part of Table 1), by importing into the platform each repository from an XPDL file, the ontology and the semantic annotation from OWL. Then, we performed three queries **Q1-Q3** against each repository. Q1 is analogous to the one shown in Section 3. Q2 *retrieves every opal:Object that is related to a concept used for the annotation of an activity lying on a path from an activity annotated with A to an activity annotated with B*. Q3 *retrieves every sub-process that is executed as an alternative to one where an activity annotated with C is eventually executed*. We report for each run (bottom part of Table 1) the number of results obtained and the total time spent for the evaluation, including the QuBPAL query translation (*QuBPAL2LP*), the communication overhead between Java and XSB and the export of the results as a new XPDL file (*XpdWriter*).

**Table 1.** Evaluation Results

Test Data Sets						
	<i>Nr. of BPS</i>	<i>Tot. Size</i>	<i>Nr. of Gateways</i>		<i>Min BPS Size</i>	<i>Max BPS Size</i>
<b>T1</b>	50	11757	4114		172	308
<b>T2</b>	100	18888	6442		157	237
<b>T3</b>	200	25229	8556		104	164
Set Up Phase Evaluation						
	BP Repository Import		BRO Import		Σ Import	
	<i>XPDL2BPAL</i>	<i>XSB Compile</i>	<i>OWL2LP</i>	<i>XSB Compile</i>	<i>OWL2LP</i>	<i>XSB Compile</i>
<b>T1</b>	3.6	7.4	1	0.7	1.8	1.2
<b>T2</b>	7.8	11.2	1	0.7	2.5	1.7
<b>T3</b>	15.3	18	1	0.7	3.3	2.5
Run Time Phase Evaluation						
	Q1		Q2		Q3	
	<i>Nr. of Res.</i>	<i>Time</i>	<i>Nr. of Res.</i>	<i>Time</i>	<i>Nr. of Res.</i>	<i>Time</i>
<b>T1</b>	11	2.5	133	4.8	47	10.2
<b>T2</b>	15	5.3	125	11.3	66	14.7
<b>T3</b>	9	8	109	17.2	44	16.9

## 5 Related Work and Conclusions

In this paper we presented a framework conceived to complement existing BPMS by providing advanced querying services. The proposed solution is based on a synergic use of ontologies to capture the semantics of a business scenario, and a business process modelling framework, to represent the underlying application logic. Both frameworks are seamlessly connected thanks to their grounding in logic programming and therefore it is possible to apply effective reasoning methods to query the knowledge base encompassing the two.

A first body of related works proposes the extension to business process management of techniques developed in the context of the semantic web. Relevant work in this direction has been done within the SUPER project, where several foundational ontologies to model functional, organizational, informational and behavioral perspectives have been developed. In [11] a querying framework based on such ontologies is presented. In [12] SPARQL queries, formulated through a visual language, are evaluated against business processes represented through a BPMN meta-model ontology annotated with respect to domain ontologies. Other approaches based on meta-model ontologies have been discussed, e.g., [13]. Unlike the aforementioned works, where the behavioral aspects are hidden or abstracted away, dependency constraints defined in terms of the execution semantics can be used in a QuBPAL query. Hence, the *EKB* provides a homogeneous framework where one can evaluate complex queries that combine properties related to the ontological description, the workflow structure, and the behavioral semantics of the modeled processes.

Other approaches for BP querying are based on graph matching, through visual languages [14,15] grounded in graph grammars. Such approaches allow the user to query the graph representation of a process workflow in an intuitive way, but they need to be combined with external tools to reason about properties of the behavioral semantics (e.g., [14] implements translations to finite state models to be verified by using model checking techniques). Such approaches strongly differs from ours on scope and purpose, since their focus is on verifying structural features of process

schemas, and the semantics of the business domain is not considered. Our framework not only provides a method based on Datalog for querying the structure of the workflow graph, but due to the logic-based representation it also integrates additional reasoning services. In particular, a very relevant advantage of QuBPAL is the possibility of formulating queries involving the knowledge represented in domain models formally encoded by means of ontologies. Indeed, QuBPAL queries can be posed in terms of the ontology vocabulary, which offers a “global view” of the processes annotated with it, hence *i*) decoupling queries from specific processes, *ii*) overcoming semantic heterogeneities deriving, e.g., from different terminologies, *iii*) posing queries at different generalization levels, taking advantage of the semantic relations defined in the ontology, such as *subsumption*.

Future works are intended to increase the expressivity of the approach, by supporting a larger number of workflow patterns [1], and to perform the optimization of the query evaluation process, that can be strongly improved by exploiting query rewriting techniques.

## References

1. ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M., Russell, N.: Modern Business Process Automation: YAWL and its Support Environment. Springer, Heidelberg (2010)
2. Hepp, M., et al.: Semantic business process management: A vision towards using semantic web services for business process management. In: Proc. ICEBE (2005)
3. De Nicola, A., Missikoff, M., Navigli, R.: A software engineering approach to ontology building. *Information Systems* 34(2), 258–275 (2009)
4. De Nicola, A., Missikoff, M., Proietti, M., Smith, F.: An Open Platform for Business Process Modeling and Verification. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6261, pp. 76–90. Springer, Heidelberg (2010)
5. Lloyd, J.W.: Foundations of Logic Programming, 2nd edn. Springer, Berlin (1987)
6. OMG: Business Process Model and Notation, <http://www.omg.org/spec/BPMN/2.0>
7. XPD L 2.1 Complete Specification, <http://www.wfmc.org/xpdl.html>
8. OWL 2: Profiles, <http://www.w3.org/TR/owl2-profiles>
9. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proc. ICSE 1999, pp. 411–420 (1999)
10. Missikoff, M., Proietti, M., Smith, F.: Querying semantically annotated business processes. In: IASI-CNR, R, pp. 10–22 (2010)
11. Markovic, I.: Advanced Querying and Reasoning on Business Process Models. In: Proc. BIS 2008. LNBIP, pp. 189–200. Springer, Heidelberg (2008)
12. Di Francescomarino, C., Tonella, P.: Crosscutting concern documentation by visual query of business processes. In: Ardagna, D., Mecella, M., Yang, J. (eds.) Business Process Management Workshops. Lecture Notes in Business Information Processing, vol. 17, pp. 18–31. Springer, Heidelberg (2009)
13. Haller, A., Gaaloul, W., Marmolowski, M.: Towards an XPD L Compliant Process Ontology. In: SERVICES I 2008, pp. 83–86 (2008)
14. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)
15. Beerli, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying business processes with BP-QL. *Information Systems* 33(6), 477–507 (2008)

# Introducing Affective Agents in Recommendation Systems Based on Relational Data Clustering

João C. Xavier-Junior<sup>1</sup>, Alberto Signoretti<sup>2</sup> Anne M.P. Canuto<sup>3</sup>,  
Andre M. Campos<sup>3</sup>, Luiz M.G. Gonçalves<sup>1</sup>, and Sergio V. Fialho<sup>1</sup>

<sup>1</sup> Computing and Automation Engineering Department, UFRN, Natal, Brazil  
jcxavier01@gmail.com , {lmarcos,fialho}@dca.ufrn.br

<sup>2</sup> Computer Science Department, UERN, Natal, Brazil  
albertosignoretti@uern.br

<sup>3</sup> Informatics and Applied Mathematics Department, UFRN, Natal, Brazil  
{anne, andre}@dimap.ufrn.br

**Abstract.** This paper proposes the use of a multi-agent system (MAS) with affective agents in a recommendation system based on relational data clustering. This MAS works as a mediator between the user and the data stored in the system. In the proposed system, after logging in, each user will have an affective agent, called Interface agent, for interaction purposes. This agent models the user's data requests according to the user's profile and its affective status, sending it to the Recommender agent, which recommends a set of map points to be visualized. The system analyzes the user's feedback in order to verify whether the recommended information was satisfactory. This feedback is analyzed through the monitoring of the interaction interface..

**Keywords:** Affective agents, Recommendation Systems, Data clustering.

## 1 Introduction

Recommendation systems are becoming popular in applications and as a research field due to recent advances on Internet technologies [7]. Currently, a reasonable number of recommendation systems use data mining techniques to make recommendations after learning information from previous actions and attributes of users. In this sense, these systems are often based on the development of user profiles that can be persistent (history data), ephemeral (actions during the current session), or both. Clustering is one of the data mining techniques used in recommendation systems and they can be used to identify groups of users who appear to have similar preferences. In this case, the recommendations can be based on previous actions of users with similar preferences.

Emotions are an important part of the human decision making process. Therefore, computers that can interpret these emotions or affective states of the user may be more effective in providing assistance. This proposal uses the concept



of affective agents in both ways, either regarding human assistance, in our case a guidance, as well as regarding the improvement of the efficacy of the agent's reasoning process by defining an attention focus.

The main motivation of introducing affective agents in recommendation systems based on relational data clustering was originated in the positive results obtained from two existing projects developed in our research group. One of them uses an attention focus conducted by the agent's affective status to guide the perception process of goal oriented agents [13]. The other one uses the relational structure of the database and the searches log file carried out by the users in order to create data clusters [5]. The aim of this work is to join the positive points of these two previous works to improve the effectiveness of a recommendation system.

In this proposed system, the user can enjoy the best possible experience in obtaining the information he seeks. The environment is a multi-agent system that uses a number of affective agents. They use three types of information to provide an accurate guidance for data search: the information stored in data clusters, the searcher's log files and the attention focus based on the user's preferences.

## 2 Related Works

Recommendation systems can use different techniques in their decision-making process, such as: rule-based filtering, content-filtering or collaborative filtering. Some recent works in collaborative filtering have used data mining techniques to build recommendation models from large data sets. One of the data mining techniques most used in Collaborative filtering is data clustering, as found in [1] [12]. In [12], for instance, the authors used the k-means clustering algorithm, which is the most popular algorithm, where the user specifies the number of clusters (k). In addition, in [1], a clustering approach based on a semi-supervised learning process was proposed to construct a recommendation system for movies. In the mentioned work, a constrained k-means was used to create a highly accurate recommending system.

On the other hand, computational models representing emotion, personality and humor have been widely used to simulate human behaviour. Several different architectures have already been proposed with this aim, like ALMA (A Layred Model of Afect) [2], BASIC (Believable Adaptable Socially Intelligent Character for Social Presence) [11] among others. These agents can show surprise or fear and a set of other emotions, including a mood driven behaviour initialized by the agent's personality.

The present work uses emotions and personality based models in the agent architecture. In this respect, this work is similar to the aforementioned works. Nonetheless, this work does not attempt to represent human behaviour as realistic as possible. Our goal is to use the human behaviour model as an inspiration for providing an efficient focusing process for the agent decision-making process.

### 3 System Overview

The recommendation mechanism was implemented as part of Geographical Information System named NatalGIS. This is a multi-agent system which provides geographic information for a group of researchers, environmental managers and tourists. This system is responsible for the environmental management of an area of coral reefs located in the state of Rio Grande do Norte, Brazil [3].

The original system architecture is composed by three types of agents (Controller, Interface and Recommender). In NatalGIS, each user will interact with the system through an Interface agent (IA), which captures the user's profile (explicitly through the user's interaction or being implicitly inferred) and monitors the user's feelings and intentions through the interface events. After collecting this information, a recommendation request is prepared and sent to the Recommender agent (RA), which accesses the clusters database (clusters DB view repository) in order to mine the necessary information for building the data recommendation requested by the Interface agent.

The process of requesting and receiving this recommendation is called a search cycle. In addition, this recommendation is a list of geospatial map points, which might be interesting for the user. Finally, the Controller agent (CA) is responsible for activating the Interface agents for each user's login request and for maintaining an overall control of the system. The Interface agent (IA) is an affective agent whose decision-making process will take into consideration the affective state and attention focus of each user.

#### 3.1 Interface Agent

The Interface agents are software facilitators (wizards) that run in the background, analyzing the users actions in order to assist them or perform actions in their place. Each user logged in the system has a particular Interface agent.

As already mentioned, in the NatalGIS system, the Interface agent (IA) is an affective agent which is responsible for tracking down the intentions and feelings of the user in order to create a search process which is related to the needs of the user. The tracking process is implemented through the analysis of the events triggered by the user (event trigger analyser module) in conjunction with previously gathered information about the user (the user profile database). This database contains important information about the user, such as the user's type and preferences, as well as an initial focus of attention.

Initially, the user profile database contains only information provided by the user. In addition, the clustering and recommendation processes are basically built using the type of user. When the user gains access to the system, his/her profile is updated and those processes will also use access information of the user. In other words, the initial recommendation process is made using generic information about the users (for example, the user type) and the more a user accesses the systems, the more detailed and accurate the recommendation process becomes.

The evaluation of the triggered events can cause reactions that change the state of the affective module. These changes are used to define the operation to

be performed on the attention focus and to adapt the agent's behavior, aiming to fulfil the needs of the user.

In order to guide the user, the IA uses its affective module and the interaction with the user to establish the best attention focus. Then, a request of service is sent to the Recommender agent (RA), which will be responsible for creating a ranking list of map points (recommendation object - RO) based on the information sent by the IA (affective state and the user profile).

Finally, after receiving the map points list the IA presents the most important points, according to the ranking list of size  $L$ . The value of the  $L$  parameter is a function of the agent's affective state [13]. These most important points are then presented to the user and the form of presentation is based on the preferences of the user. After this last step, a new search cycle begins and the IA initiates the evaluation of the events that were triggered by the user.

The other two agents can be described as follows.

- Recommender agent (RA): It is a service provider for the Interface agents. It is important to emphasize that each Interface agent will interact with one Recommender agent. However, each Recommender agent will provide service to, at most,  $N$  Interface agents simultaneously. In this sense, the size of the pool is dynamic and controlled by the Controller agent (CA) which defines this size based on the quantity of activated IAs. This agent is responsible for applying a ranking function in order to generate an ordered list of map points based on the users' relevance criteria defined by the IA. The ranking function uses the attention focus, user profile information, the access history of the user and the history of access of other users with similar profiles as arguments. After completing the service task, the RA sends the ordered map points list to the requesting IA.
- Controller agent (CA): it is responsible for managing all the connections requested by the users. For each requested connection, an Interface agent (IA) is activated and a Recommender agent (RA) is associated to this IA. For the RAs, the pool is initially defined with a proportion of at most five IAs for each RA ( $N = 5$ ). This proportion is dynamically adjusted depending on the time response of the pool of RAs (the value of  $N$  can increase or decrease). When the session is terminated, the CA is responsible for deactivating the corresponding IA and the association recommender/interface is finished. This agent is responsible for clustering the relational data related to the users when accessing the NatalGIS system. In addition, the overall clustering process is performed periodically (daily, weekly or monthly). Then, the list of suggestions is constantly updated, based on the users actions.

## 4 Implementational Aspects

### 4.1 The Affective Module

The proposed affective component uses three kinds of components: Emotions as a short term affective element which decays according to the personality profile.

Moods as a medium term affective element which lasts longer than emotions and is not associated with a specific event. Finally, Personality as a long term affective element that reflects individual characteristics and influences the environmental perception, behavior and action [6]. The selection of the models for implementation was carried out considering recognized computational implementations. As a result, the following models were selected: the BigFive model for personality [8], the PAD model for humor (mood) [9] and the OCC appraisal model for emotions [10].

According to BigFive or OCEAN model, it is possible characterize define a person according to five personality traits named as Openness, Conscientiousness, Extroversion, Agreeableness, and Neuroticism. According to The PAD model, mood can be expressed in terms of Pleasure, Arousal and Dominance and represented in a 3D space [9]. Finally, the OCC model [10] is used to implement the emotions elicitation by a cognitive model based essentially on the concepts of appraisal and intensity of environmental events.

## 4.2 Attention Focus

The attention Focus works as a virtual membrane in order to produce a filtered subset of geographical information for the ranking map points process [13]. This process of calculating the attention focus is based on the affective module of the architecture. This link with the affective module is useful for reducing or increasing the amount of geographical information according to the IA's emotional state, as well as for defining what is more or less important for the IA in a particular state. The agent's attention focus is structured using spatial and temporal focus [13].

**Spatial Focus.** The spatial focus is responsible for establishing the level of importance (*LoI*) of the geographical information available in the data base. It is characterized according to a set of attributes called *aspects* (the secondary tables of the relational database). The interest related to each aspect of the database is represented by values defined by the IA in conjunction with the user. These values are mapped in a range  $[0, 1] \in \mathbb{R}$ . They are updated during the use of the system following the feedback sent by the user in relation to the recommended information. Based on the *LoI*, the RA creates a priority order over the map points (ROs) available in the data base.

**Temporal Focus.** Once the map points are defined and prioritized by the RA, it is important to reduce this set to a limited number of elements. Thus, the temporal focus is responsible for setting the amount of map points that will be available for the IA. This is done by cutting the ordered list according to a specific *L* parameter.

As already mentioned, *L* is not a fixed value and it varies according to the agent's affective state and, as a result, assumes different values when, for instance, the agent is relaxed or stressed. This approach is used by Janis and Mann [4] when they describe the relationship between the effectiveness of individuals and their state of stress through an inverted "U" curve. The assumptions used in this

work are: 1) the quantity of map points considered for the user presentation is directly related to the effectiveness of the user interpretation and decision, and 2) the agent's emotional state is directly related to the agent's stress level. As a result, the  $L$  function was empirically defined as a pseudo Gaussian distribution. This derivation is implemented using an average value among the distance of the point representing the current agent's state of humor in PAD-3D space [9], and the positions representing the extreme relaxed mood and the extreme anxious mood.

### 4.3 The Relational Clustering Module

This module uses a Data Mining technique (Clustering) to identify different groups of users in NatalGIS according to their history data (access logs). The clustering processing is made by the Controller agent and it uses the history of the users' accesses to group the users. In order to generate different clusters, we used the Hierarchical Agglomerative algorithm (average link method) and also modelled the relational database (access logs) in a hierarchical structure, aiming to improve the effectiveness of the clustering algorithm [5].

The result of the clustering procedure is then stored in a database called Clusters Database. Aiming to ease the access of the RAs to these data, we generated a view which consists of the clusters created in the clustering process. As mentioned in a previous section, the CA is responsible for periodically running this process, according to the system's monitoring aspects (number of new users' accesses and value of clustering measures).

### 4.4 The Creation of the Map Points List

The creation of the map points list is based on a process of association user-cluster(s). This process is made in two different ways, for first time users and for users with previous access to the system. In the former case, only the user type is taken into consideration in the association process. In this sense, the systems will make its recommendation based on previous accesses of the users of the same type of the first time user. For users with previous accesses, the association process takes into consideration the previous accesses to define the cluster(s) to which the user belongs.

Once the association user-cluster is defined, the list of map points is created. As already mentioned, the  $L$  value is defined by the temporal attention focus. The level of importance (LoI) is a parameter used to rank all the aspects in the spatial attention focus. For the highest priority aspect, look at the associated cluster(s) to find instances (map points) with the same aspect. The criterion used to select the instances is the distance to the centroid of the cluster(s). Selected instances are put in the map list and the amount of instances in the map list is defined by the level of importance of the corresponding aspect.

## 5 The Proposed Prototype

Aiming to evaluate the NatalGIS multi-agent architecture and the clustering module, a prototype has been developed. This prototype has been used by a small number of researchers, managers and tourists.

Regarding affective agents, this prototype uses a fixed temporal and dynamic spatial focus. The latter is based on elicited emotions from triggered events by the user. In this sense, we evaluate the best set of interface events for correct IA's emotion elicitation. Moreover, the handling of the prototype by different users helps to consolidate the set of implemented emotions. Although the OCC model classifies 22 types of emotions, we implemented only 4 types in our prototype: joy, distress, disappointment and satisfaction. For the next phase of development, the affective architecture will include the PAD and OCEAN models. This enhancement will allow the temporal focus to acquire a dynamic behavior.

Regarding the clustering module, the usage of the prototype will be helpful to evaluate the accuracy and the relevance of the recommended information. This evaluation can indicate whether the clustering parameters need adjustments.

The Interface agent (IA) was not implemented with a visual representation. The main idea is to create a user's empathy with the system instead of with the visual representation of the agent.

## 6 Final Remarks

In this paper, we presented a manner of combining the use of affective agents with data clustering as a way of providing more accurate mechanisms for recommending geographic information to the users of the NatalGIS system.

A prototype was developed and tested by a small group of alpha-users. Their feedback has been used to guide the development of the whole system. More accurate tests must be carried out in order to evaluate the relevance of specific components of the proposal. For instance, the influence of the modelled elements on the level of user satisfaction concerning the received recommendations. Nevertheless, it is worth to emphasize that we have been producing interesting results in both areas of this paper (affective agents and data clustering).

Finally, the fact of using an affective attention focus in a recommendation system based on data clustering is a new approach and seems to decisively contribute for recommending accurate and relevant information. Moreover, this approach allows a dynamic system customization during running time.

**Acknowledgements.** This work has the financial support of CAPES and CNPq (Brazilian Research Councils), under processes numbers BEX 2481/09-0, 550810/2007-2 (140013/2008-3), 140239/2011-1 and 479629/2008-0.

## References

1. Christakou, C., Lefakis, L., Vrettos, S., Stafylopatis, A.: A movie recommender system based on semi-supervised clustering. In: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC 2006), vol. 2, pp. 897–903. IEEE Computer Society, Washington (2005), <http://portal.acm.org/citation.cfm?id=1134824.1135435>

2. Gebhard, P.: Alma - a layered model of affect. In: Proceedings of The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 29–36. ACM, Utrecht (2005)
3. Cabral, J., Goç Alves, L., Xavier-Junior, J.: Web gis by ajax for analysis and control of environmental dat. In: Proceedings of 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 25–32. Plzen, Czech Republic (2009)
4. Janis, I.L., Mann, L.: Decision making: a psychological analysis of conflict, choice, and commitment. Free Press, New York (1977)
5. Xavier-Jr, J.C., Freitas, A.A., Canuto, A.: Web log data clustering for a multi-agent recommendation system. In: IEEE Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC 2010), Qingdao, China, pp. 471–476 (July 2010)
6. Kasap, Z., Moussa, M.B., Chaudhuri, P., Thalmann, N.M.: Making them remember: Emotional virtual characters with memory. *IEEE Computer Graphics and Applications* 29(2), 20–29 (2009)
7. Schmidt-Thieme, L., Friedrich, A.F., Guest, G.: introduction: Recommender system. *IEEE Intelligent Systems* 22(3), 18–21 (2007)
8. McRae, R.R., Costa, P.T.: The five-factor model of personality: Theoretical perspectives, chap. Toward a new generation of personality theories: Theoretical contexts for the five-factor model, pp. 51–87. The Guilford Press, New York (1996)
9. Mehrabian, A.: Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology* 14(4), 261–292 (1996)
10. Ortony, A., Clore, G., Collins, A.: *The Cognitive Structure of Emotions*. Cambridge University Press, New York (1998)
11. Romano, D.M., Sheppard, G., Hall, J., Miller, A., Ma, Z.: Basic: A believable adaptable socially intelligent character for social presence. In: Proceedings of The 8th Annual International Workshop on Presence (PRESENCE 2005). Springer, London (2005)
12. Romero, C., Ventura, S., Delgado, J., De Bra, P.: Personalized links recommendation based on data mining in adaptive educational hypermedia systems. In: Duval, E., Klamma, R., Wolpers, M. (eds.) EC-TEL 2007. LNCS, vol. 4753, pp. 292–306. Springer, Heidelberg (2007)
13. Signoretti, A., Feitosa, A., Campos, A.M., Canuto, A.M., Fialho, S.V.: Increasing the efficiency of npcs using a focus of attention based on emotions and personality. In: SBC - Proceedings of SBGames 2010, Florianopolis - Brazil (November 2010)

# Converting Conversation Protocols Using an XML Based Differential Behavioral Model

Claas Busemann<sup>1</sup> and Daniela Nicklas<sup>2</sup>

<sup>1</sup> OFFIS - Institute for Information Technology

<sup>2</sup> Carl von Ossietzky Universität Oldenburg

Oldenburg, Germany

{busemann,nicklas}@offis.de

**Abstract.** Conversation protocols are used to communicate between information systems, with services, with sensors, or with human beings. As many of these protocols share similar application purposes, the protocols also seem to share similar basic functionality. Using the Extensible Markup Language (XML) as a unified syntax for data transmission might be a step in the right direction. Beyond that, mapping techniques like the Extensible Stylesheet Language Transformation (XSLT) or XQuery can be used to achieve compatibility between different protocols by converting the messages of a protocol into a new representation. However, these approaches come to an end as soon as the communication behavior of a protocol changes. In this paper, we introduce a method that allows the modeling of conversation protocol changes which also includes changes of the communication behavior. The model is based on XQueries, which are used for the data transformation, and adds a layer on top of it. Our case study and evaluation shows that a high level of compatibility between protocol versions and different protocols can be achieved when using the described approach.

**Keywords:** XML, Protocols, Conversion, Model.

## 1 Introduction

Due to the increasing number of information systems which share similar application purposes, the number of conversation protocols that are used by these systems also increase. Conversation protocols can range from simple textual representations, like SSI [9] or NMEA [18] to complex binary protocols which are often used in industrial fieldbus applications (e.g., EtherCAT [6]). To achieve interoperability between information systems and to integrate smart objects, the developers usually program adapters or wrappers, which is a complex, time-consuming, and expensive task [14]. The cost for smart object integration into existing systems can reach 35 to 50 percent of all system development [5]. Web-based conversation protocols often use XML to encode their messages. Because of the easy handling and supporting standards, many XML based protocols have been developed for different application domains over the last several years [22]. Hence, as there is a high number of XML protocols which often exist in different



versions, these protocols often have to be converted from one representation to another to achieve compatibility between different systems. For data conversion between these protocols, more general approaches like XSLT [13] or XQuery [2] exist. These techniques allow the developer to specify queries or mapping rules that transform messages to another representation. However, when dealing with conversation protocols, these techniques come to an end, as they do not consider the communication behavior. If, for example, an additional message is added, removed, has to be merged from other messages, or has to be split into several other messages, there is no way to define this in the transformation script.

We solve this problem by adding an additional layer to the the standard data conversion. This layer allows the modeling of communication behavior changes from one protocol to another. Therefore we can achieve a high level of compatibility between extremely different XML protocols without implementing additional code. This is also proven by our evaluation. Our main contribution is a model that allows the simple description of communication behavior changes.

The rest of the paper is organized as follows: Section 2 shows the related work. The models which have been defined to describe the behavior changes can be found in Section 3 while Section 4 includes a case study. An evaluation of this study can be found in Section 5. We conclude in Section 6.

## 2 Related Work

A number of authors have proposed models for specifying schematic correspondences between schema based documents or databases like [19, 23]. Mapping concepts which define how to map one schema to another have been proposed by [16, 11]. However, none of these papers focus on behavioral changes that are necessary if the mapping is done between two communication protocols. XSLT [13] and XQuery [2] are some of the most common methods to convert XML documents [21] from one representation to another or even to other data formats [4]. Several papers described how to optimize these techniques to improve the results (e.g. [12, 10]). Due to the high comfort of these techniques they are also used to convert XML based protocols as mentioned in [8, 17]. However, these protocol conversions only focus on the transformation of the transmitted data and usually do not convert the communication behavior. If the behavior is considered, the conversion is done by implementing specific code for each use case instead of using a model like described in this paper.

The basic conversion of protocols has been proposed in the mid-nineties by authors like [15, 20]. These works mainly focus on the conversion from one network structure to another including functions like address conversion and stream handling. The work does not include the model based description and manipulation of communication nor the generation of request and response messages out of the collected data. Glombitza et al. provide an open framework for protocol conversion [11]. The framework is able to convert protocols such as HTTP, SMTP, and LTP among each other. It is also able to convert the payload data of uncompressed SOAP and compressed SOAP. However, the conversion is mainly

done using static rules and is not based on a model like described in this paper. There is also no function to generate additional responses.

### 3 Models

The description of communication behavior changes is based on two separate models. The “Message Model”, which is used to identify a message of a protocol, and the “Differential Protocol Model”, which is used to describe the changes from one protocol to another. Both models are defined using an XML schema and are described in detail in the following subsections.

#### 3.1 Message Model

The Message Model is needed to identify messages in the protocol. Identifiers are then used in the Differential Protocol Model to define behavioral changes. We use XML standard technology for this task. Therefore the model defines one or more elements that have to be part of the message. The model also specifies a name for each message. Technically this is realized by defining one or more XPath expressions for each message. Every XPath expression also includes a minimum number and maximum number of nodes that have to be returned by that expression. This allows an easy identification of different messages, even if the basic structure is similar, as XPath can also be used to check the content of a node.

#### 3.2 Differential Protocol Model

The Differential Protocol Model is used to describe the changes between two protocols. It is separated into two parts: The data transformations and the communication behavior conversion. The data transformation is defined using XQuery expressions. Therefore the model simply specifies an XQuery and the messages on which the expressions should be executed. Note that this kind of data transformation is state of the art and does not differ from methods used in other projects. To specify the communication behavior conversion, we introduce a set of basic commands. These commands have been identified by analyzing the behavioral changes between several protocols and protocol versions. However, while these commands seem to work well for the protocols discussed in this paper, other protocols might need additional ones. This will be evaluated in our future work. The commands are specified as follows:

**Message Deflected:** The message exist in the target protocol but has to be converted. It specifies the original and the target protocol message name.

**Message Removed:** This command reflects that a message does not exist in the target protocol. Usually, the message would be dropped by a converter. However, the command also defines “message fall back names” for this command. A fall back name still reflects that the message does not exist in the target protocol, but the message can be converted to a message that is known by the target protocol.

**Message Merged Passive:** The message can be converted to a message of the target protocol, but additional data is needed to build the new message. The command also means that no additional action has to be performed to gather the needed data. As the command also specifies the messages that are needed to build the new message, a converter can use this information to transform the messages into the new message.

**Message Merged Active:** Similar to the passive merge command, this command means that additional messages are needed to build the target message. However, in this case it is necessary to send one or more messages to a specified service to gather the needed data. Therefore the command specifies one or more XQueries which are used to generate request messages out of the collected messages, the destinations for each of the request messages, and The name of the new message.

**Message Separated:** The received message has to be separated into at least two new messages. Therefore the command specifies the names of the new messages. The separation of the messages is done by the XQueries which have been defined for the data transformation.

**Response Added:** The sender or any other destination awaits a response message which will not be sent by the target protocol stack. This could for example be an acknowledgment, that has been removed in a new protocol version. The command specifies all information that is needed to build the response message. Therefore it includes an XQuery that is used to generate the response. However, this only works if all information that needed to build the response is either included in the original message or in the XQuery.

As all of these commands can be combined with each other, a high level of compatibility between different protocol versions and even protocols can be achieved.

## 4 Case Study

The case study is done by converting the SCAI protocol of the SCAMPI middleware [7] into the protocols of the SWE framework [3] and backwards. Both protocols are conversation protocols for open sensor platforms and share a similar application purpose, which is administrating sensors and transmitting sensor data. However, the communication behavior of most parts of these protocols completely differs from each other. The messages of the protocols can be broken down to four basic functions: request a measurement, receive a measurement, create a new sensor, and request a new sensor. Each one of these commands is usually followed by a response. There are some additional commands in both protocols which cannot be converted as they do not have a representation in the other protocol. These commands are ignored for the evaluation. In the following subsections the four convertible commands and the differences in their communication behavior are explained in detail.

**Request a Measurement:** This command is used to request a measurement from a sensor or a service. In both protocols a request is sent to the service which is answered with a measurement. As the communication behavior is similar and

the data in both messages is the same, no communication behavior conversion is needed.

**Receive Sensor Data:** This command is used to transmit sensor data to a service and seem to be quite similar. However, there is one difference: the SCAI protocol does not send an acknowledgment back when a measurement is received. If the SCAI protocol is converted to the SWE protocol, the acknowledgment can simply be removed (Message Removed). If the SWE protocol is converted to the SCAI protocol the acknowledgment has to be generated by the converter as the SWE protocols demands it (Response Added). Fig. 1A shows the converted communication behavior in both directions.

**Create a New Sensor:** This command is used to register a new sensor at a service. Therefore a description of the sensor is transmitted. The communication behavior is extremely different. The SCAI protocol sends three different messages to create a sensor. The SWE protocol sends the same data in one message. All

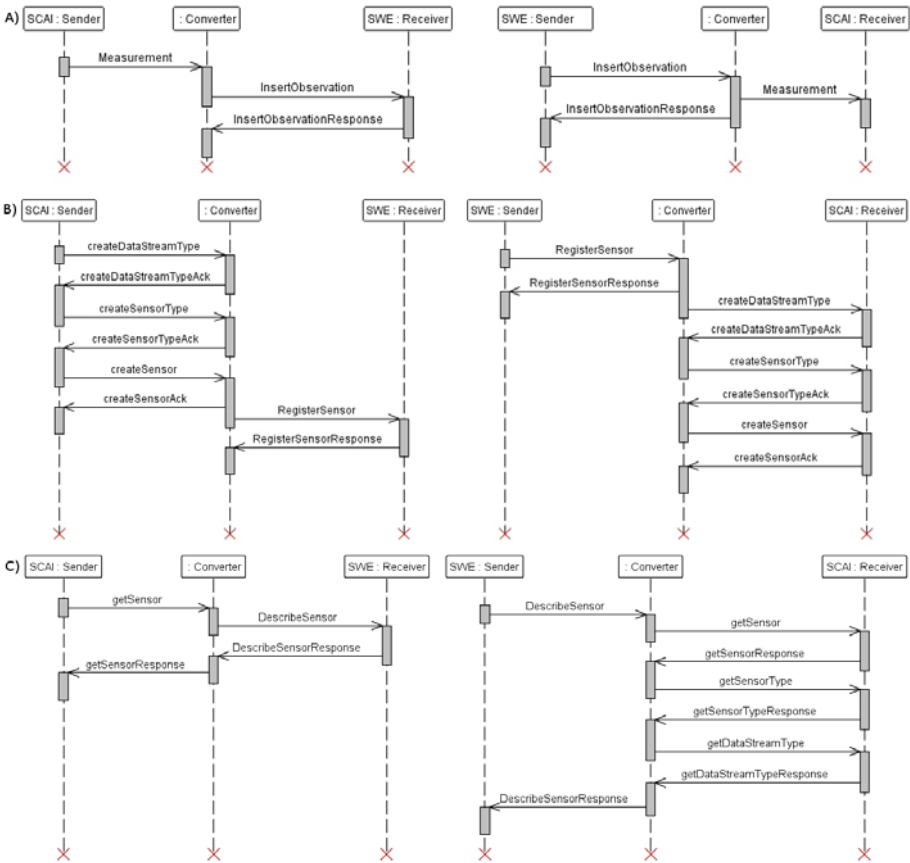


Fig. 1. Converted Communication Behavior of Protocols Functions

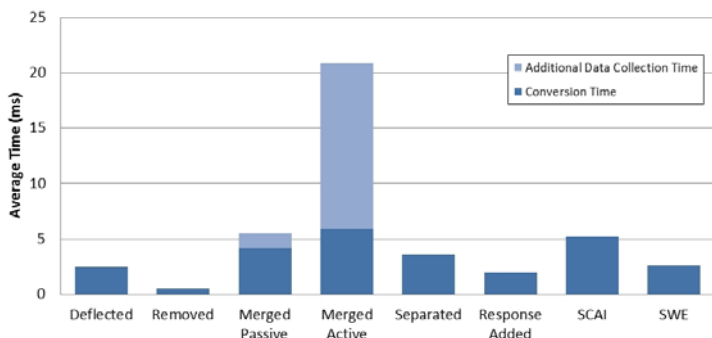
messages are respond with an acknowledgment. The acknowledgments do not contain the same data and can not be converted. To convert the SCAI protocol to the SWE protocol the three request have to be collected . As soon as all messages are collected they can be combined (Message Merged Passive) and the SWE request can be produced. As the SWE acknowledgment can not be converted, this message is simply removed (Message Removed). Therefore a new SCAI acknowledgment message is created by the converter (Response Added). To convert the SWE protocol to the SCAI protocol the request message has to be separated (Message Separated). The responses can not be converted and are removed (Message Removed). The converter however produces a response out of the initial request as the SWE protocols demands an acknowledgment (Response Added). Fig. 11B shows the converted communication behavior in both directions.

**Request Information About a Sensor:** This command is used to get information about a sensor from a service. Therefore a request is sent to the service which sends back a response that includes the sensor description. The communication behavior seems to be quite similar. This is the case if the SCAI protocol is converted to the SWE protocol. However, if the SWE protocol is converted to the SCAI protocol this does not work, as the SCAI response does not contain the information that is needed to build the SWE response. The converter has to collect additional data form the SCAI service. Therefore it generates a request message out of the message and sends it to the SCAI service. The response of this request is again used to generate a new request which causes a new response. After that all responses are used to generate the response to the SWE request (Message Merged Active). Fig. 11C shows the converted communication behavior in both directions.

## 5 Evaluation

In this section, the conversations from the previous section are evaluated by measuring the average processing time for each message. Based on these measurements, the average processing time for each command described in Section 3 has been identified. Fig. 2 shows the results of this evaluation. The measurements include the identification of the protocol, the message, the behavior changes, the transformation to the target protocol and a consistency check.

Most operations could be done with an average processing time between 2 and 5 ms. However, merging messages seems to take longer then the other commands. This is mainly caused by the fact that messages have to be collected before the output message can be generated. When a message is merged active the processing time takes about 21 ms. However, this strictly depends on the number of messages that have to be gathered and the communication delay of the services that response these messages. In our case, two additional messages have been collected which took about 15 ms. We also measured the average time that it takes to convert a SCAI message to a SWE message and backwards. This test assumes that every message of each protocol appears equally often. Converting SCAI to SWE messages takes about 19 ms. The conversion from SCAI to SWE



**Fig. 2.** Average time for message conversion

takes 16 ms. Due to the fact that the converter has to store messages if they have to be merged passive, we also analyzed the memory behavior. Using the standard configuration of the java virtual machine the converter can store about 450 messages until it runs out of memory. However, this depends on the size of the messages and the configuration of the virtual machine.

This evaluation shows that the conversion of protocols using the described model is possible while still having acceptable processing times. Active merging a message seems to be the most expensive command, however this strictly depends on the service that is used to gather the missing data.

## 6 Conclusion and Future Work

In this paper we introduced a method that allows the extension of standard XML data transformation techniques with functions that can be used to convert the communication behavior of one XML protocol to the behavior of another. This is realized by defining two models to describe protocol differences and by implementing a converter which interprets the models and thereby can convert the communication behavior of the protocols. The evaluation proves that a higher level of compatibility can be achieved when using the differential behavior model by converting the SCAI protocol the SWE protocols and backwards. In the future, we are going to extend the model with new functions which will be based on the analysis of other protocols. Also the generation of requests and responses will be extended, so that it will be able to generate messages out of data that may have been collected in other sessions.

## References

1. An, Y., Hu, X., Song, I.-Y.: Round-trip engineering for maintaining conceptual-relational mappings. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 296–311. Springer, Heidelberg (2008)
2. Boag, S., Chamberlin, D., Fernández, M.F., Florescu, D., Robie, J., Simon, J.: XQuery 1.0: An XML Query Language. Tech. rep., W3C (2007)

3. Botts, M., Percivall, G., Reed, C., Davidson, J.: OGC® sensor web enablement: Overview and high level architecture. In: Nittel, S., Labrinidis, A., Stefanidis, A. (eds.) GSN 2006. LNCS, vol. 4540, pp. 175–190. Springer, Heidelberg (2008)
4. Breitling, F.: A standard transformation from xml to rdf via xslt. CoRR abs/0906.2291 (2009)
5. Brodie, M.L.: Integration in A Service-Oriented World: The Big Picture. In: I-ESA (2006)
6. Buettner, H., Janssen, D., Rostan, M.: EtherCAT - the Ethernet fieldbus, PC Control Magazine 3: 1419. Tech. rep. (2003)
7. Busemann, C., Kuka, C., Westermann, U., Boll, S., Nicklas, D.: Scampi - sensor configuration and aggregation middleware for multi platform interchange. GI Jahrestagung, 2084–2097 (2009)
8. Chu, X., Buyya, R.: Service oriented sensor web. In: Mahalik, N.P. (ed.) Sensor Networks and Configuration, pp. 51–74. Springer, Heidelberg (2007)
9. European Union Framework Programmes on Research: Simple Sensor Interface Protocol v1.2. Tech. rep (2006)
10. García-Sánchez, P., Laredo, J.L.J., Sevilla, J.P., Castillo, P.A., Guervós, J.J.M.: Improved evolutionary generation of xslt stylesheets. CoRR abs/0803.1926 (2008)
11. Glombitza, N., Mietz, R., Romer, K., Fischer, S., Pfisterer, D.: Self-description and protocol conversion for a web of things. In: International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, pp. 229–236 (2010)
12. Groppe, S., Groppe, J., Böttcher, S., Wycisk, T., Gruenwald, L.: Optimizing the execution of xslt stylesheets for querying transformed xml data. Knowl. Inf. Syst. 18(3), 331–391 (2009)
13. Kay, M.: XSL Transformations (XSLT) Version 2.0. Tech. rep., W3C (2007)
14. Lempert, S., Pflaum, A.: Towards a Reference Architecture for an Integration Platform for Diverse Smart Object Technologies. In: MMS (2011)
15. Liu, M.T.: Network interconnection and protocol conversion. Advances in Computers 42, 119–239 (1996)
16. Mao, L., Belhajjame, K., Paton, N.W., Fernandes, A.A.A.: Defining and using schematic correspondences for automatically generating schema mappings. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 79–93. Springer, Heidelberg (2009)
17. McCann, D., Roantree, M.: A query service for raw sensor data. In: Barnaghi, P., Moessner, K., Presser, M., Meissner, S. (eds.) EuroSSC 2009. LNCS, vol. 5741, pp. 38–50. Springer, Heidelberg (2009)
18. National Marine Electronics Association: NMEA 0183 Standard. Tech. rep. (2010)
19. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. 10(4), 334–350 (2001)
20. Tao, Z.P., Bochman, G.v., Dssouli, R.: An efficient method for protocol conversion. In: International Conference on Computer Communications and Networks, p. 0040 (1995)
21. Wang, X., Cao, C.: Mining association rules from complex and irregular xml documents using xslt and xquery. In: ALPIT, pp. 314–319 (2008)
22. XML Protocol Working Group: XML Protocol Comparisons. Tech. rep., W3C (2000)
23. Yan, L.-L., Miller, R.J., Haas, L.M., Fagin, R.: Data-driven understanding and refinement of schema mappings. In: SIGMOD Conference, pp. 485–496 (2001)

# Facilitating Casual Users in Interacting with Linked Data through Domain Expertise

Cormac Hampson and Owen Conlan

Knowledge & Data Engineering Group,  
Trinity College Dublin, Ireland  
{Cormac.Hampson,Owen.Conlan}@tcd.ie

**Abstract.** Linked Data use has expanded rapidly in recent years; however there is still a lack of support for casual users to create complex queries over this Web of Data. Until this occurs, the real benefits of having such rich metadata available will not be realised by the general public. This paper introduces an approach to supporting casual users discover relevant information across multiple Linked Data repositories, by enabling them to leverage and tailor semantic attributes. Semantic attributes are semantically meaningful terms that encapsulate expert rules encoded in multiple formats, including SPARQL. Semantic attributes are created in SABer (Semantic Attribute Builder), which is usable by non-technical domain experts. This opens the approach to almost any domain. A detailed evaluation of SABer is described within this paper, as is a case study that shows how casual users can use semantic attributes to explore multiple structured data sources in the music domain.

**Keywords:** Domain Experts, Casual Users, Semantic Attributes, Linked Data, Data Exploration.

## 1 Introduction

In recent years, casual computer users have found themselves accessing diverse structured and semi-structured data on an increasingly regular basis. A casual computer user can be seen as one with Internet browsing ability, but without programming skills and data modeling expertise [1]. With the advent of the Linking Open Data community project<sup>1</sup> and the proliferation of web services and mash-ups, this trend of casual users interacting more with distributed data sources is likely to continue. However, while one may intuitively expect the additional structure in the data to have been exploited to provide sophisticated query capabilities, this has largely not proven to be the case [2]. Many applications using structured data provide access to their underlying data store via query languages; however these are suitable primarily for developers with a knowledge of the language rather than regular end-users wishing to ask very specific questions through a usable human interface [2].

Imagine a casual computer user trying to locate “all nostalgia music artists currently touring the USA”. The information necessary to answer this quite subjective

---

<sup>1</sup> <http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>



query is likely to be stored over multiple sources, thus techniques should support casual users to find it. One such way to help casual users would be to give them access to Subject Matter Expertise (SME) to support complex query formation. Such human domain expertise<sup>2</sup> is widely used in expert systems, which are developed to help users find reliable information in narrow areas such as medicine or accounting [3]. There are many examples of how casual users could benefit from being supported by such experts to explore Linked Data collections. For instance the increasing availability of government health, transport and education data can open up the opportunity to discover things like bicycle accident black spots and correlations between certain environmental factors and high levels of a disease [4]. Thus, supporting casual users with SME can help them find useful information.

Though SME can be an excellent way of supporting casual users to locate useful information, the encoding of this expertise into an Expert System's knowledge base and the creation of the corresponding rules in its inference engine is not trivial. Hence, domain experts who do not have skills in these areas are likely to require the additional support of a knowledge engineer. Furthermore, even when this expertise is encoded successfully, it often resides in bespoke standalone systems, which leaves little scope for the expertise to be reused in different applications. This is especially true when the user interface, domain expertise, and the knowledge base are tightly coupled together. Thus, a generic platform where non-technical experts could encode their experience and knowledge of a domain in a reusable model would be a welcome solution. Such an approach may not capture as much detail as a bespoke system supported by a knowledge engineer, but the simplicity of its design and its generic applicability would make widespread implementation much easier.

This paper addresses the compelling need for casual users to be supported in exploring information domains encoded as Linked Data. It introduces an approach to accomplish this that allows these users to leverage domain expertise to create complex queries across a domain. By appealing to domain experts with limited computer skills, it makes the adoption of this approach in different domains much more likely. As the dependence on digital data escalates, and the increasing use of such repositories by casual users occurs, the need for such user-friendly approaches will be greatly increased. This paper addresses this need directly and is structured as follows: section two discusses some related work; section three gives a brief overview of SARA (Semantic Attribute Reconciliation Architecture) and section four introduces SABer (Semantic Attribute Builder); section five discusses a case study in the music domain; section six details an evaluation of SABer; and section seven summarises the paper.

## 2 Related Work

A number of generic Linked Data browsers have been developed, such as Tabulator [5] and Disco<sup>3</sup>, in order to support casual users browse RDF data on the web. These

---

<sup>2</sup> In terms of this paper an "expert" is defined as anyone with an opinion on a specific domain and the ability to express it; however developers should be free to limit their choice of experts to a specific individual or select group.

<sup>3</sup> <http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/>

systems display all the information that they can find about specific dereferenceable URIs, and enable the user to browse from resource to resource. However, though aimed at supporting casual users, Linked Data browsers have yet to gain wide usage outside the Semantic Web community. Sparallax<sup>4</sup> was developed in order to provide a set-based faceted browsing interface for SPARQL endpoints. Facets and collections are automatically generated as the user browses the data sets, however users of Sparallax are limited to searching one data source at a time, and it is not possible to reconcile information from different repositories. Unfortunately there is no end-user evaluation of the system published, so it is unclear yet if casual users fully accept the set based paradigm and can see its benefits.

Explorator [6] is a domain independent tool for exploring the Semantic Web, which aims at enabling users with minimal knowledge of RDF models to explore an RDF database, without a priori knowledge of the data domain. The developers of Explorator feel that current tools which allow the user to manipulate raw RDF data do not provide a user friendly way to ask questions, and that the user only has a limited way to rearrange, group or filter the data, and process it further. Similar to Sparallax, Explorator uses a set-based paradigm, where after an initial key word search (to match resources in the RDF) the resulting set of data can be further manipulated to either remove uninteresting elements or to add additional elements of interest. Initial small scale experiments have been conducted [7] with users who knew some basic concepts of the Semantic Web and RDF. However, even using these participants who are more experienced than casual web users, the participants struggled to perform tasks using Explorator. The developers plan additional larger-scale experiments to compare different user interface alternatives and interaction paradigms, in order to better support both novice and expert users in exploring the Semantic Web. This is important as the developers admit that Explorator is currently better suited to advanced users who have solid knowledge about RDF rather than casual users.

A common way of supporting data exploration by casual end users is by enabling them to leverage Subject Matter Expertise (SME) encoded by domain experts. This practice is widespread in domain specific applications, because generic search technology starts to degrade when addressing the needs of users in particular fields such as healthcare and finance [3]. Unfortunately the complexity of encoding SME into such a system (be it rules for an inference engine or comprehensive development of an ontology), is such that a non-technical domain expert is likely to require the support of a knowledge engineer. This in turn increases the time and costs involved in such an exercise and the SME encoded often resides in a standalone system with little scope for expertise to be reused in different applications. There is a tradeoff between the benefits of offering manually generated expertise to users and the efficiency of providing automatically generated supports (e.g. dynamically created facets). Hence, the development of domain independent tools which are quick and easy to use, but do not overly compromise in terms of the features they deliver (e.g. expressiveness of queries supported) should be greatly encouraged.

Konduit VQB [8] is a visual query builder that aims to assist users in building queries and running them over RDF data. Specifically the tool is aimed for users with little or no knowledge about SPARQL, as well as users more familiar with Semantic

---

<sup>4</sup> <http://sparallax.deri.ie>

Web technologies. Though Konduit VQB is not strictly an SME encoding tool; it could be used by non-technical experts to encode SME in SPARQL, with this SME then leveraged by users wishing to explore an RDF repository. The developers feel that the schema-based SELECT query builder in the tool is the most user-friendly approach to building SPARQL queries, and recommend it for those users having the least knowledge of semantic technologies. According to the developers, it is simple, intuitive and satisfies the large number of occasions when the user wants to search for something based on certain properties. This specific approach does not support the full range of expressivity that SPARQL offers. However the developers feel that the extra complexity involved in forming more complex queries (e.g. using the CONSTRUCT query builder they offer) is likely to be inaccessible to users with no knowledge of RDF and SPARQL.

SPARQLViz [9] contains a Graphical Query Composer that allows users to generate syntactically correct SPARQL queries through a wizard interface. It is particularly useful for novice users who have little or no experience with SPARQL, as the user is able to compose a valid query simply by using familiar user interface widgets in a wizard-like manner [9]. According to the developer the two best features of the approach offered by SPARQLViz are that generated queries are always valid, saving a lot of debugging time, and an in-depth knowledge of the SPARQL query language is no longer needed to be able to generate them. Unfortunately there were no experiments conducted regarding the second assertion, so it is not possible to determine what level of SPARQL expertise (if any) SPARQLViz requires of its users in order to generate queries.

In terms of using a visual query builder to encode SME, a form or wizard based approach appears to offer the most potential for non-technical domain experts due to their relative simplicity. Specifically, the more intuitive schema-based approach for generating statements was mentioned as the most suited to non-technical users. An example of a commercial application that uses such a rule building approach is iTunes<sup>5</sup>, which has a *smart playlist*<sup>6</sup> builder. The *smart playlist* builder is aimed at casual end users and allows them to generate XQueries in a schema-based approach. These XQueries generate specific music playlists that allows users to better manage and sort their music collection. At smartplaylists.com<sup>7</sup>, users can share the rules they have encoded in iTunes, so that others can benefit from their expertise. This supports the notion that SME encoded in this way can be a valuable way of supporting other users to explore and manage their day to day data. The smart playlist feature in iTunes has even been used to organise pdfs about radiology [10] which further supports the contention that a schema-based approach to rule generation, coupled with a form/wizard interface is accessible to a non-technical user of computers. There are other approaches to visually building queries such as the use of graphs which are employed by NITELIGHT [11] and iSPARQL<sup>8</sup>. However, in order to use these tools the user must have a full comprehension of the underlying RDF schema and the query language syntax, which implies a high cognitive load for newcomers and less

---

<sup>5</sup> <http://www.itunes.com>

<sup>6</sup> <http://support.apple.com/kb/HT1801>

<sup>7</sup> <http://www.smartplaylists.com/>

<sup>8</sup> <http://demo.openlinksw.com/isparql/>

experienced users [6]. Indeed the developers of NITELIGHT admit that the close correspondence between the graphical notations and query language constructs makes the tool largely unsuitable for users who have no previous experience with SPARQL. Hence, the form/wizard interface appears to be more suited to non-technical domain experts, and is the approach employed by SABer. Section 6 of this paper describes a user experiment with non-technical domain experts, which validates this approach.

### 3 SARA (Semantic Attribute Reconciliation Architecture)

This paper describes an approach for exploring Linked Data by enabling users to form queries from Subject Matter Expertise (SME). This approach is realised in a middleware system called SARA (Semantic Attribute Reconciliation Architecture) [12,13]. SARA is a domain independent framework that that supports casual users (using an application connecting to its API) leverage SME in order to query different information sources (including Linked Data) in a consolidated fashion. SARA considers three groups of users: end-users (who access SARA through custom third-party apps that talk to the API); domain experts (who are not necessarily technical and can use SABer to create semantic attributes for use in SARA); knowledge engineers (who register a data source for use in SARA/SABer by creating a Source Model).

In SARA, superclasses are key entities from the domain chosen by an expert (the term is used here in a sense unrelated to OWL or RDFS superclasses). In essence, any queries made by client applications to the SARA are looking to return instances of one of these superclasses. For instance in the music domain you could select superclasses such as *Artist*, *Song*, *Album*, *Venue* etc. There is no limit to the number of superclasses you select, and there is no need to define any relationships or properties for them which can be an arduous task when creating domain ontologies. Furthermore, new superclasses can be added to SARA at any time so you are not limited to your initial selection.

SARA currently supports data sources in XML, RDF or those accessible through Web APIs. In order to add a Linked Data source to SARA a reusable Source Model must be created in XML for each SPARQL endpoint or Linked Data Repository. Figure 1 shows an example of a Source Model for an RDF data source. It contains the name of the data source, the address of the RDF database or SPARQL endpoint, any namespace prefixes that the predicates use, and any superclasses that this source contains. The SPARQL code corresponding to each domain superclass is the code that will return instances of this superclass within the data source. In its simplest form this code is just a single SPARQL triple in the form of `?result ?predicate_name ?id`, with `?predicate_name` the only code changing from one superclass to another. Thus in Figure 1 `?result foaf:name ?id` (line 7) returns instances of the *Music\_Artist* superclass and `?result mysp:country ?id` (line 11) returns instances of the *Country* superclass.

As can be seen in Figure 1, the predicate `mysp:country` has the alias *Country that MySpace artist is from* (line 22) so that it is clearer to domain experts what this predicate actually represents. This predicate has the subject *Music\_Artist* as this is the domain superclass that has `mysp:country` as a property in this particular data source. Likewise, the predicate `mysp:country` has a corresponding object of a *Country*

superclass (line 24), as these are the type of instances that this predicate returns from this data source. This process of associating predicates with superclasses allows multiple data sources with different schemas to co-exist in SARA, without having to go through the time consuming and problematic process of being homogenised to a canonical model. In the case of the *myp:totalfriends* predicate, its subject is also *Music\_Artist* (line 16), with its object being a specific value (the number of friends an artist has on the MySpace website) rather than another domain superclass. Thus “Value” is inputted instead of a superclass name (line 17). The final part of the model shown in Figure 1 describes the transform information necessary to convert instances of one superclass to another. In this instance it depicts a single SPARQL triple that transforms *Music\_Artist* instances into *Countries* (line 30). In other cases a transform may contain several triples.

```

1. <Name>MySpace SPARQL Endpoint</Name>
2. <Location>http://virtuoso.dbtune.org/sparql</Location>
3. <Graph>&lt;http://dbtune.org/myspace/&gt;</Graph>
4. <Prefix>foaf:&lt;http://xmlns.com/foaf/0.1/&gt;</Prefix>
5. <Superclass>
6.   <Name>Music_Artist</Name>
7.   <Code>?result foaf:name ?id.</Code>
8. </Superclass>
9. <Superclass>
10.  <Name>Country</Name>
11.  <Code>?result mysp:country ?id.</Code>
12. </Superclass>
13. <Predicate>
14.   <Name>mysp:totalFriends</Name>
15.   <Alias>Total friends on MySpace is</Alias>
16.   <Subject>Music_Artist<Subject>
17.   <Object>Value</Object>
18.   <Units>N/A</Units>
19. </Predicate>
20. <Predicate>
21.   <Name>mysp:country</Name>
22.   <Alias>Country that MySpace artist is from</Alias>
23.   <Subject>Music_Artist<Subject>
24.   <Object>Country</Object>
25.   <Units>N/A</Units>
26. </Predicate>
27. <Transform>
28.   <Subject>Music_Artist</Subject>
29.   <Object>Country</Object>
30.   <Join>?Music_Artist mysp:country ?id.</Join>
31. </Transform>

```

**Fig. 1.** Sample Source Model for an RDF source

Once a Source Model is created for a specific repository it can be reused in any other SARA installation. This means that collections of different Linked Data repositories can be assembled very quickly in SARA if their Source Models have

already been generated. Any predicates registered in a Source Model (which capture details about the technical access to a source) can be used to generate semantic attributes (which capture domain knowledge and may be created by non-technical domain experts). Semantic attributes are defined as discrete units of domain expertise that can be combined together and tailored to support user exploration of an information domain. Within SARA, semantic attributes encapsulate query fragments that can be combined together to form complex queries. They typically act as abstractions and simplifications from the raw data, which are intended to make it more accessible for the ordinary, non-expert user. For instance, semantic attributes can encompass subjective characteristics such as *nearness*, *popularity* and *expensiveness*, as well there more objective values such as *distance in miles*, *number of records sold* and *price*.

Tailoring a semantic attribute enables users to specify, if they wish to, what their interpretation is of a *high quality* audio file or a *popular* song etc. This is achieved by allowing variables in a semantic attribute's query fragment to be populated by user inputted parameters. Each semantic attribute can also include default values defined by the domain expert that allow informed queries to be run quickly without tailoring. A semantic attribute may contain just a single predicate or else combine multiple predicates into a single semantic attribute, e.g. combining the predicates *bitrate*, *sample rate* and *file type* into a single semantic attribute *audio file quality*. Furthermore, all semantic attributes can also be sub-categorised into a number of separate ranges or parameters e.g. the semantic attribute *Price* could be divided into {Expensive - Average - Cheap}, and *Weight* into {Under Weight - Normal Weight - Over Weight - Obese}. This categorisation allows non-experts to access information without detailed knowledge of the domain. All semantic attributes within SARA are represented in XML as Semantic Attribute Models.

SARA has already been successfully applied to a number of domains including music, films, digital humanities and publications. This has been helped by the fact that SARA supports the reusability of semantic attributes and source models in different installations. However, in order to make SARA's widespread deployment more likely, it was necessary to support non-technical domain experts to encode SME as semantic attributes. This led to the development of the SABer (Semantic Attribute Builder) authoring tool, which can be used by non-technical experts without the support of a knowledge engineer.

## 4 SABer (Semantic Attribute Builder)

The Semantic Attribute Builder (SABer) was developed in Adobe Flex to work in tandem with SARA, and focuses on allowing non-technical users to encode their expertise in SPARQL, XQuery or as native API calls, and encapsulating this SME as semantic attributes. It achieves this by automating as many processes as possible, ensuring that rules generated are syntactically correct, and by not requiring the domain expert to understand the underlying query languages. This paper will limit discussion to the creation of SPARQL based Semantic Attributes that are compatible with Linked Data repositories. It must be stressed that the novelty of SABer is not specifically in its GUI itself, but rather that in conjunction with SARA it enables non-technical domain experts to quickly generate SME in a wide range of domains.

Creating a semantic attribute using SABer is a two step process with each step having a dedicated page in the application. The first process in step one is to name the semantic attribute being created. It is important to choose a descriptive name that conveys its meaning clearly, as this is the name that end users will see in the client application. The next task to complete on this page is to select the predicates you want to create your semantic attribute rules from. Once a domain expert is satisfied with the predicates they have chosen they must select from a drop down menu the type of semantic attribute they want to create. They have a choice of three; expert, template or hybrid. An expert semantic attribute only contains the expert's default rule(s) which can't be tailored, a template semantic attribute contains no expert default rule(s) and must be tailored by the end user, and a hybrid semantic attribute contains expert default rules as well as corresponding template rules which can be tailored. When the user is satisfied with his choices he can click to move onto the next stage.

Depending on whether the domain expert has selected an expert, template or hybrid rule the next page displayed will vary slightly. However, regardless of the type of semantic attribute being created, it is at this stage that the domain expert creates the rule or rules for their semantic attribute. The first thing a domain expert must do to generate their SPARQL query is to select the domain superclass that they want to return. To choose a superclass, the domain expert must simply select it from a dropdown menu. This generates the first part of each rule and is printed onscreen as "Return any <Superclasses> where" as depicted in Figure 2, where the user has selected the superclass *MusicArtist*.

Underneath this line the expert is presented with three dropdown menus, a text field and a button all in a row. The first dropdown menu contains all the superclasses that the RDF source has associated with. Depending on what superclass the expert chooses, the predicates (or more precisely the alias of the predicates) in the adjacent dropdown menu will change accordingly. This second dropdown menu contains all the predicates that the domain expert has chosen in the first step, but restricted to those predicates that are associated with the superclass chosen in the first dropdown menu. Thus Figure 1 shows that when the domain expert chooses the superclass *MusicArtist* from the first dropdown menu, they are presented in the second dropdown menu with the elements *Country from is*, *Total Friends on MySpace is*, and *Total page views on MySpace*. However if the domain expert had chosen *Song* as the superclass in the first dropdown menu of that line, then the second dropdown menu would have been populated with *Track Duration*, *Composer*, *Genre* etc. If there are any units associated with the predicate chosen then they are displayed at the end of the line to make clear what range of values is appropriate to input.

RETURN ALL: MusicArtist (5) WHERE:

MusicArtist Country from is = [ ] +

- Country from is
- Total friends on MySpace is
- Total page views on MySpace is

GET RESULTS

Fig. 2. Sample two Lines of Expert Rule for RDF Based Semantic Attribute

The domain expert would then select the predicate they were interested in and then move on to the third dropdown menu. This dropdown menu contains the available list of operators that the end user can select from. Currently these include; *Greater than*, *Less than*, *Equals*, *Not Equals to*, *Greater than or Equals to*, and *Less than or Equals to*. The domain expert simply selects which operator they want from the drop down menu. The operators other than *Equals* all result in a FILTER statement being added to the SPARQL rule that is in the process of generation.

All the domain expert has to do to finish this line of the rule is to input a value into the adjacent textbox. Thus in Figure 3, on the second line the expert chose the metadata *Total Friends on My Space*, the operator *<*, and inputted the value *50,000*. This essentially equates to the WHERE part of a SELECT SPARQL statement with the rest of the query automatically generated from the information defined in the Source Model. If the domain expert wants to add more lines to this rule all they have to do is click on the “+” button at the end of the line. This adds another identical line underneath the first, except that it has an additional *and/or* dropdown menu at the start of the line and an additional “-” button at the end.

The *and/or* dropdown menu allows the user to specify if the *MusicArtist* should satisfy both of the rule lines or either of them. If the user selects *or* from this dropdown menu, it results in an OPTIONAL statement being added to the SPARQL rule that is being generated. In Figure 2 the expert has used *and* so only wants *Music Artists* that satisfy both rule lines e.g. *Music Artists whose Total Friends on MySpace is less than 50,000 AND greater than 20,000*. The additional “-” button at the end of the line allows for a rule line to be deleted easily. Each parameter can contain as many rule lines as the domain expert wants, with Figure 2 showing the completed four line rule for *Averagely Popular Irish Artists on MySpace*. At any time in the process the domain experts can select the “Get Results” button to see what instances are currently in the data source that satisfy their rule.

RETURN ALL: MusicArtist (S) WHERE:

MusicArtist Total friends on MySpace is < 50000 +

and MusicArtist Total friends on MySpace is > 20000 + -

and MusicArtist Country from is = Ireland + -

**Fig. 3.** Sample four Lines of Expert Rule for RDF Based Semantic Attribute

The process for creating a template semantic attribute based on RDF data is almost identical to the process just described for creating an expert semantic attribute. The only difference is highlighted in Figure 4. Instead of having a blank text field in which domain experts can input a specific value, they instead are presented with another dropdown menu with two options “*Some Text*” and “*Some Number*”. This allows domain experts to create rules such as *Return all Artists where Country From = “Some Text”* or *Return all Songs where chart position < “Some Number”*. By generating these kinds of rules it enables end users to tailor a rule more specifically to what they want.



RETURN ALL: MusicArtist (S) WHERE:

MusicArtist Total friends on MySpace is > Some Number +

and MusicArtist Total friends on MySpace is < Some Number + -

and MusicArtist Country from is = Some Text + -

Fig. 4. Sample four Lines of Template Rule for RDF Based Semantic Attribute

The process for a domain expert creating hybrid semantic attributes for RDF sources is identical to creating an expert semantic attribute. The only difference is that when a hybrid semantic attribute is submitted, SABer automatically generates an associated template rule for each of the expert rules to support tailoring. Once any semantic attribute gets submitted, the values and rules inputted into SABer get concatenated with a template to form an XML Semantic Attribute Model. This model then gets saved to SARA and made available to client applications interested in that domain.

## 5 Case Study

This section describes a case study of a SARA installation that connected to five separate music data sources in three different formats. The aim of this case study was to show how SARA technically supported queries to reconcile information from these separate sources. The data sources used in this case study were:

1. An XML iTunes library with over 30,000 songs stored in an eXist database
2. The US Singles charts from 1950-2008 stored as XML in an eXist database
3. The freebase.com music SPARQL endpoint<sup>9</sup>
4. The MySpace.com SPARQL endpoint<sup>10</sup>
5. Last.fm web services<sup>11</sup>

The eXist databases and remote SPARQL endpoints could be directly accessed by queries encapsulated in the semantic attributes. However in the case of web services with a native API, such as the Last.fm service used in this case study, a Java wrapper was needed to proxy queries and results. Each of these five sources had Source Models registered to SARA which in turn got visualised in SABer. The domain superclasses chosen were *Artist*, *Song*, *Album* and *Country*.

SABer was then used to create semantic attributes which were stored in SARA's Semantic Attribute Library. As will be shown in the section 6, SABer can support non-technical domain experts to generate such semantic attributes. For this case study twenty-five semantic attributes for the domain were created including:

- Artists currently touring specific countries
- Top MySpace artists from specific countries
- Popular Jazz artists in the US Charts in the 1980s
- Similar artists to a specified artist

<sup>9</sup> <http://lod.openlinksw.com/sparql>

<sup>10</sup> <http://virtuoso.dbtune.org/sparql>

<sup>11</sup> <http://www.last.fm/api>

Once the semantic attributes were made available in SARA it was possible for queries to be sent to it from a client application via its API. For instance, queries combining multiple semantic attributes that reference different sources could be sent to SARA such as:

- Return all Artists from the iTunes collection (iTunes XML database), that have Concerts Scheduled in the USA (Last.fm web service), despite their most recent top 10 Album in the USA being more than ten years ago (US charts database).
- Return all Countries (MySpace SPARQL endpoint) that had popular Artists in the USA during the 1990s (US charts database)
- Return all Songs by The Beatles (freebase SPARQL endpoint) that are in top 10 popular Beatles songs on Last.fm (Last.fm web service) despite not charting in the top 10 in Americas (US charts database).

Many of these queries allowed specifics to be tailored by the end user, so that they could easily specify different bands other than *The Beatles*, tailor the definition of *popular*, or change the range of time. Once the results from the individual data sources were sent back to SARA they were reconciled into a final result set. This was made possible as the different sources contained instance level identifiers to help disambiguation (dereferenceable URIs used in Linked Data are an example of such instance level identifiers). The final result set was then sent as XML for rendering in the client application.

This case study has shown how SARA supports semantic attributes created in SABer to be utilised by a client application, gives consolidated access to multiple sources of different types, and enables instance level integration of results from different data sources. Furthermore, extra superclasses, data sources and semantic attributes could be appended to the system seamlessly if required, and the models generated for this case study could be plugged into different installations that required access to music information.

## 6 SABer Evaluation

For SARA to be applicable to many domains it had to be shown that non-technical domain experts could successfully generate semantic attributes in SABer. As stated previously, for the purposes of this paper ‘non-technical’ people refers to computer literate participants with basic skills such as operating Internet browsers, but with no computer programming experience. SABer has previously been used by domain experts with computer programming experience to create useful semantic attributes that were deployed within applications. Hence, this experiment was devised to measure the usability of SABer and its effectiveness in supporting non-technical domain experts to generate semantic attributes. Furthermore, it helped explore whether SABer (and by extension the Semantic Attribute Model) can sufficiently abstract the user away from the differences in the various semantic attributes and their underlying data types.

Two groups were assembled of twelve participants each (one group technical the other non-technical), with each person engaging in the experiment separately and in

isolation from other participants. An entire session including the demonstration, performance of tasks and filling in of questionnaires typically took around forty minutes to complete. The first step of the experiment was for each participant to have the creation of three different semantic attributes in the music domain demonstrated to them. This demonstration was done by the evaluator and consisted of him inputting the semantic attribute details from a task sheet into SABer. Demonstrating these three tasks involved an identical process to what the participants would be undertaking in the experiment. SABer supports semantic attributes of three different types (expert, template and hybrid) and of three data types (XML, RDF and data accessible through a Web API). Each of the three tasks demonstrated to the participants involved a combination of a different semantic attribute type with one of the different data types.

Once these three tasks were demonstrated, the participants were given nine different semantic attributes of varying complexity to create in SABer. These tasks were presented to the participants in a random order as users tend to get quicker with later tasks when they are more familiar with the application interface. By presenting the tasks in a random order it meant that the average time taken to create semantic attributes would not be longer due to appearing near the start of the list, and likewise not be shorter due to being near the end of the list. The semantic attributes users were asked to create are listed as follows:

1. Quality Of Audio Files in my iTunes Collection
2. Top Singles in US in 1990s
3. Artists of a Genre who had US single that Reached a Specific Position
4. Countries Paul McCartney has Concerts Scheduled in
5. Artists Scheduled to Play Iceland
6. Popular Beatles Songs According to Last.fm
7. Songs On A Specific Album By Specific Artist
8. Countries with Very Popular Artists on MySpace
9. Irish Artists Popularity on MySpace

These semantic attributes spanned the same five sources outlined in the case study section.

Because the participants were given the rules to encode into semantic attributes (e.g. highly Irish popular artists on MySpace are those with *Total friends on MySpace > 50,000 and Country from is = Ireland*) it was not necessary for them to be “expert” in the domain per se. This was justified as the usefulness of the semantic attributes being created was not being evaluated, but rather the ease in which coherent rules could be constructed by non-technical users. During the course of the experiment the length of time it took each semantic attribute to be created was recorded, the accuracy of the semantic attribute noted (whether it exactly matched the semantic attribute given to them on paper) and any questions or problems that they asked recorded. Users were also given the opportunity to create semantic attributes of their own after creating the nine semantic attributes set for them.

Once finished using SABer, each user completed a SUS (Standard Usability Scale) test [14] to measure the systems usability and also filled in a short post questionnaire. The SUS test provided an indicator as to the usability of SABer and the post questionnaire gave space for participants to elaborate on any usability issues or functionality they would like to see. The post questionnaire asked participants to

specify if they found inputting rules for any of expert, hybrid or template semantic attributes a considerably more difficult challenge, and likewise if they found inputting rules for any specific query type significantly more challenging. This would allow it to be accessed if part of the SABer application needed to be adjusted to make inputting semantic attributes of certain types more intuitive. Moreover it would help determine if the Semantic Attribute Model was sufficiently generic and abstract to allow users to ignore the underlying idiosyncrasies of querying different data types.

Once the experiment was completed the results from the twelve technical users were used to provide a baseline performance in terms of speed and accuracy in creating the semantic attributes. Technical domain experts have already used SABer to generate useful and deployable semantic attributes, thus by comparing the time it takes for non-technical users to create accurate semantic attributes it would give a good indicator as to the usability and utility of the tool. Hence, if the group of non-technical users performed, on average, at a level of accuracy and efficiency near their more technical counterparts, it could be reasonably concluded that SABer was suitable for non-technical domain experts to use. Moreover, by comparing the average SUS score for technical, non-technical and overall groups, a good indication of the tool's usability would be garnered.

With regards to the speed in creating semantic attributes, on average the non-technical users were only 8.4% slower than their technical counterparts. In terms of the accuracy of the semantic attributes created, the difference in performance was even smaller between groups than in the speed comparison. Technical users on average got 8.5 out of 9 accurate with a Standard Deviation (SD) of 0.67., with non-technical users getting 8.2 out of 9 accurate (SD of 0.84). It must also be noted that all inaccuracies by participants in both groups can be classified as slips where wrong figures or spellings were inputted by the users. Slips are defined by attentional failures where the action was unintended [15]. These kinds of errors can be easily corrected, and there were no cases of a user fundamentally not being able to create a semantic attribute or giving up half way through. Furthermore, all users were able to create their own semantic attributes after completing the nine semantic attributes that were set for them. Many of these semantic attributes were of comparable complexity to those set for them during the experiment. In terms of usability, on average the technical users gave SABer a SUS score of 83.3% (SD of 9.4), and the non-technical users 74.4% (SD of 10.7) The average SUS score for all 24 users was 78.85% (SD of 10.05). Systems that score above 72.5% on the SUS scale can be classified as having good usability [16], so it can be concluded that SABer is considered a usable tool by both non-technical and technical users.

The post questionnaire that participants filled in asked them to specify if they found inputting rules for any of expert, hybrid or template semantic attributes a considerably more difficult challenge. 22 of the 24 users found no significant extra difficulty in creating semantic attributes of different types (hybrid, template or expert). None of the participants felt that creating template semantic attributes was more difficult than any of the others. This meant that all participants were comfortable with selecting "some text" or "some number" while creating rules instead of inputting specific values. This was important to validate as non-technical users would typically not be as familiar with the concept of a variable as technical users would, and this concept had to be presented to them in an intuitive fashion. The

post questionnaire also asked if participants found inputting rules in a specific query language significantly more challenging. 22 out of the 24 users found no significant difference in difficulty in creating semantic attributes using XQuery, SPARQL or API calls. This was important as it showed that users were largely indifferent to the underlying technologies they were working with, and meant that the semantic attribute sufficiently abstracted them away from the underlying technical complexity of each query language.

Usability is defined by the International Organization for Standardization (ISO) as the effectiveness, efficiency and satisfaction with which a specified set of users can achieve a specified set of tasks in a particular environment [17]. Participants in both the technical and non-technical groups were shown to perform the tasks effectively and efficiently. Moreover, there was no significant difference in speed, accuracy or perceived usability of SABer between the two groups. Users also found no significant difference in creating semantic attributes of different types or with different queries thus showing that SABer sufficiently abstracted users away from the underlying data sources. It can thus be concluded from this section that SABer is a tool with good usability and that domain experts without a computer science background could use it to create semantic attributes with a minimal amount of training. The significance of this is that it should be possible for experts in metadata rich domains to capture SME as semantic attributes if given a sufficient choice of elements. Hence, the tool is very applicable to Linked Data sources.

## 7 Summary

This paper described an approach to support casual users discover relevant information across multiple Linked Data repositories, by enabling them to leverage and tailor semantic attributes. Currently the rich metadata exposed through Linked Data is only minimally used by casual users, hence it is important to make this information more accessible to these users. SARA and its authoring tool SABer are designed to support interaction by casual users with Linked Data, and both systems were detailed within this paper. It was also shown how Linked Data repositories can be registered with SARA through a reusable Source Model, which means that once created, the same model can be reused in any installation that wants to access that specific data source.

The paper also detailed how SABer supports non-technical domain experts to encode SME as queries in multiple languages (including SPARQL) that are then encapsulated as semantic attributes. A successful evaluation of SABer was discussed in detail. Any semantic attributes generated in SABer can be reused in any installation of SARA that access those same sources, which makes it easy for different client applications to quickly benefit from the SME encoded by domain experts. This reusability of SME and its non-reliance on knowledge engineers make the approach supported by SARA and SABer suitable for almost any domain with rich metadata. A case study of SARA's use in the music domain was also described, which showed how users could form complex queries over multiple structured and semi-structured data sources (including Linked Data accessed by SPARQL endpoints). This case study showed SARA's potential to be used to help casual users navigate the rapidly expanding Web of Data.

**Acknowledgments.** This research has been supported by The Irish Research Council for Science, Engineering and Technology: funded by the National Development Plan.

## References

1. Huynh, D., Miller, R., Karger, D.: Potluck: Data mash-up tool for casual users. *Web Semantics: Science, Services and Agents on the World Wide Web* 6, 274–282 (2008)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. *International Journal on Semantic Web and Information Systems* 5, 1–22 (2009)
3. Vaughan-Nichols, S.: Researchers Make Web Searches More Intelligent. *Computer* 39, 16–18 (2006)
4. Hall, W.: What web science could mean for businesses. *uterweekly.com* (2010), <http://www.computerweekly.com/Articles/2010/04/12/240862/interview-wendy-hall-on-what-web-science-could-mean-for.htm%20>
5. Berners-Lee, T., et al.: Tabulator: Exploring and analyzing linked data on the semantic web. In: *Proceedings of the 3rd International Semantic Web User Interaction Workshop* (2006)
6. De Araújo, S.F.C., Schwabe, D.: Explorator: a tool for exploring RDF data through direct manipulation. In: *LDOW 2009: Linked Data on the Web* (2009)
7. De Araújo, S.F.C., Schwabe, D., Barbosa, S.D.J.: Experimenting with Explorator: a Direct Manipulation Generic RDF Browser and Querying Tool. In: *Visual Interfaces to the Social and the Semantic Web*, pp. 1–9 (2009)
8. Ambrus, O., Moeller, K., Handschuh, S.: Konduit VQB: a Visual Query Builder for SPARQL on the Social Semantic Desktop. In: *Workshop on Visual Interfaces to the Social and Semantic Web* (2010)
9. Borsje, J., Embregts, H.: Graphical Query Composition and Natural Language Processing in an RDF Visualization Interface. B.Sc. Erasmus University, Rotterdam (2006)
10. Qian, L.J., Zhou, M., Xu, J.R.: An easy and effective approach to manage radiologic portable document format (PDF) files using iTunes. *AJR. American Journal of Roentgenology* 191, 290–291 (2008)
11. Russell, A., Smart, P., Braines, D., Shadbolt, N.R.: NITELIGHT: A Graphical Tool for Semantic Query Construction. *Semantic Web User*, 1–10 (2008)
12. Hampson, C., Conlan, O.: Supporting Personalized Information Exploration through Subjective Expert-created Semantic Attributes. In: *IEEE International Conference on Semantic Computing. ICSC 2009, Berkeley*, pp. 384–389 (2009)
13. Hampson, C., Conlan, O.: Leveraging Domain Expertise to Support Complex, Personalized and Semantically Meaningful Queries Across Separate Data Sources. In: *IEEE International Conference on Semantic Computing. ICSC 2010, Pittsburgh*, pp. 305–308 (2010)
14. Brooke, J.: SUS-A quick and dirty usability scale. *Usability Evaluation in Industry*, 189–194 (1996)
15. Reason, J.: *Human error*. Cambridge University Press, Cambridge (1990)
16. Bangor, A., Kortum, P., Miller, P.: Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies* 4, 114–123 (2009)
17. Jokela, T., Iivari, N., Matero, J., Karukka, M.: The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11. In: *Proceedings of the Latin American conference on Human-Computer Interaction*, pp. 53–60 (2003)

# Using Expert-Derived Aesthetic Attributes to Help Users in Exploring Image Databases

Cormac Hampson, Meltem Gürel, and Owen Conlan

Knowledge & Data Engineering Group,  
Trinity College Dublin, Ireland  
{cormac.hampson,gurelm,owen.conlan}@tcd.ie

**Abstract.** Image repositories often contain a large amount of metadata about their content. However many resources, such as photographs, have inherent aesthetic qualities that can be difficult to describe in a semantically consistent and usable manner, yet would be highly valuable for users in exploring large image repositories, such as Flickr. Automatically augmenting existing metadata with expert perspectives has the potential to give users a consistent aesthetic vocabulary to search and explore such repositories. SARA (Semantic Attribute Reconciliation Architecture) is a system that supports users to leverage domain expertise while searching for items in a metadata-rich domain. X2Photo is a tool built on SARA's functionality to enable image searching based on a picture's aesthetic characteristics and user-generated tags. This paper describes X2Photo in detail, the approach to augmenting visual media with expertise, and the evaluation results which reveal how semantically described aesthetics can support complementary search axes for image retrieval.

**Keywords:** Image Retrieval, Subjectivity, Expertise, Crowd Sourcing, Semantic Attributes, User exploration.

## 1 Introduction

Visual media stored in image repositories typically contain a lot of descriptive and technical metadata, be it user-generated tags, or data extracted from low level features or from content analysis. However many resources, such as photographs, have inherent aesthetic qualities that can be hard to describe in a semantically consistent and usable manner. This can make it difficult for users to explore image repositories for relevant photographs from an aesthetic perspective. In this paper we consider aesthetic attributes as means of describing emotions associated with specific images e.g. sadness, excitement, joy etc. Typically users are reduced to keyword searching over an image's associated tags, which tend to focus more on the content of the image (what it is portraying) rather than its aesthetic. Moreover, due to its subjective nature, any tags describing an images aesthetic are likely not to be semantically consistent from one tagger to the next, adding to the difficulty of locating appropriate images. For example people are quite good at identifying and tagging elements of a photograph, i.e. it contains a dog on a beach, but not in capturing its aesthetic i.e. it is tranquil and cool.

Advances in digital photography technology and related internet storage services have given users unprecedented ability to capture photographs and make them available to a large audience. Many popular image search engines, such as Flickr<sup>1</sup>, utilise the metatags associated with a photograph. Unfortunately, they may be inaccurate and misleading at times [1]. While these tags can be useful in defining the “content” of a photograph, their single dimensional nature can limit more refined searching. Thus, consistently exposing the aesthetics of an image as a criterion that may be searched upon presents a significant challenge.

Introducing the use of expert knowledge into the exploration of photograph collections can enable end-users to discover more accurate results, and can help guide them through the process by allowing them to leverage domain specific identifiers. This may not overcome the subjectivity of the terminology, but will at least provide a consistent reference point for the domain. Furthermore, the knowledge of domain experts can lead the end-users to find photographs through the expert’s vocabulary which may not have been obvious to the end-user when initially defining the photograph they were seeking. This is because domain experts have clear perspectives when it comes to defining the key characteristics of a domain. For instance, there is an expert domain dedicated to the subject of wine. When analyzing a particular type of wine, different perceptions such as colour /clarity, bouquet and taste are used by experts of this domain to express their sensations through descriptive words or phrases. The wine-tasting terminology comprises of high-level subjective terms which are derived from low-level characteristics of a wine. For example, the term “bitter” typically refers to the tannin content of a wine, and the term “oily” is used for the combination of high glycerine and slightly low acid content [2].

This expert terminology creates a semantic space for wine tasting. The words used can differ among experts but are based on the same characteristics. Applying similar techniques to the photography domain can help define a more automated clear-cut aesthetic search environment. Even if you may not totally agree with the expert’s terminology in such a subjective area as aesthetics, you will have a consistent view on the domain and will know what type of images to expect in the results. Moreover if users can personalise and tailor an experts domain view to their own, it can give a user even more control and flexibility when searching for photographs.

This paper examines to what extent the combination of tags and subjective expertise, can support end users in exploring visual media? Here we refer to tags as annotations, such as those that Flickr users have assigned to their digital photographs. The expert knowledge is automatically derived from non-textual low-level data contained within a digital photograph; specifically the hue, saturation and lightness of its dominant colours. It is proposed that when combined, these two features should enable exploration from a content perspective (achieved via the selection of tags) and from an aesthetic perspective (derived from the expert knowledge). To investigate this, an application called X2Photo that works in tandem with the SARA [3] (Semantic Attribute Reconciliation Architecture) middleware system has been built and is described within this paper. The remainder of this paper is organised as follows: Section two highlights some related work and problems in the field; Section three discusses the design and implementation of X2Photo, as well as how SARA’s

---

<sup>1</sup> <http://www.flickr.com>



authoring tool SABer (Semantic Attribute Builder) was used by an expert in the photographic domain to describe aesthetics. Section four describes the evaluation of X2Photo and section five summarises the research discussed in this paper.

## 2 Related Work

Most of the current research in image retrieval is concerned with bridging the semantic gap. In essence, this is the gap between the low-level physical features of the image and the high level perception of what the image portrays. As Hare states [4], the representations one can compute from raw image data cannot be readily transformed to high-level representations of the semantics that the images convey. It is these semantics in which users typically prefer to articulate their queries.

Research activity in visual image retrieval increased following the adoption of Content-Based Image Retrieval (CBIR). CBIR is the method of retrieving images on the basis of automatically-derived features such as colour, texture and shape. These systems try to retrieve images that are similar to a specification or pattern (e.g. shape sketch, example image) a user defines. The automatic retrieval process within these systems suggest an advantage compared to keyword based search systems as there is no possibility of the necessary metadata not being present. However, the limitations of current content-based retrieval approaches and their incompatibility with searchers' queries are often pointed out [4, 5]. The major obstacle in CBIR approaches is the gap between visual feature representations and semantic concepts of images. In general, the problem with these algorithms is their dependency on visual similarity in judging semantic similarity [6]. Especially for photographs, it is very difficult to devise effective features that reflect their aesthetic characteristic. As semantic similarity is a highly subjective measure, it is not reasonable to rely on such algorithms, especially when the semantic space comprises of aesthetic values.

Image retrieval based on keyword features [7, 8] was mainly developed by the database management and information retrieval community. The typical query scenario in such image retrieval systems is Query By Keyword (QBK). In this process the semantics of images are represented by keywords, with query results being acceptable if the keyword annotations are accurate and complete. However, as the size of the image database gets larger, manual annotation cannot be regarded as a viable procedure to continue. Popular image search engines such as Google<sup>2</sup>, Yahoo!<sup>3</sup> and Bing<sup>4</sup> try to overcome this issue by extracting the keyword features surrounding an image on the Web. Although this method can find numerous results, the returned images are not entirely accurate since there is no guarantee that surrounding textual information relates directly to the image. Likewise, when trying to attach semantics to visual content, you have the problem of dealing with homonymy, where a single tag may have various meanings. Hence engines that retrieve images indexed through such methods can only be accurate within a certain limit [9].

Flickr is an online community platform that enables its users to upload, store and organise digital photos. Features that Flickr uses to strengthen its metadata are to

---

<sup>2</sup> <http://www.google.com/imghp>

<sup>3</sup> <http://images.search.yahoo.com>

<sup>4</sup> <http://www.bing.com/images>

allow users to group their photos into sets, and their sets into collections. Moreover users of Flickr can create and add photographs to special interest groups on any possible topic, improving the relevance of a photo's metadata. Within Flickr, users and their contacts form the backbone of photograph propagation. Research indicates that social browsing, i.e. finding photographs by browsing through the photograph streams of contacts, is one of the primary methods by which users find new images on Flickr [10]. This suggests that in such an environment users are likely to "follow" other users and that photograph enthusiasts welcome the idea of expert guided browsing.

Looking at the current approaches, it is apparent that bridging the semantic gap is still an open issue. Indexing based on surrounding textual information is highly unreliable, and textual annotations depend on the knowledge and expressiveness of individuals, which causes ambiguity. Retrieving images through this textual data often results in inaccurate and irrelevant clusters of images. Furthermore, current technologies that are based on low-level visual information do not allow users to search for images by higher-level semantics. The need to provide initial query images or to find images based on unintuitive low-level characteristics explains why these approaches haven't yet found a noticeable place in the commercial world. Regarding photography appreciation, both of these approaches, though acceptable for defining content, are inefficient in reflecting the aesthetic characteristics of images. Sinha and Jain [19] point out that content only is not enough in inferring the semantics of photographs and suggest fusing content and context to extract semantics; referred to as a contextual analysis. Enser [20] also suggest that it is necessary to utilise both the concept and the content of a photograph to improve the efficiency of image retrieval techniques, stating that hybrid image retrieval systems should be welcomed.

Being able to retrieve images from image repositories using high level semantics defined by experts may help these systems to realise their potential more. Likewise, when introducing subjective qualities such as aesthetics as search criteria, it would be useful to have systems that can support personalisation within the process. By combining this with a user interface that supports end-users to manipulate photograph collections in a personalisable and compelling way, the system would empower users in exploring and accessing large image repositories.

### **3 X2Photo**

The previous section highlighted how image retrieval techniques that combine textual annotations or keyword search with low-level characteristics are being considered in order to bridge the semantic gap. X2Photo is an application designed to help tackle this problem and is described in detail in this section. SARA [3] (Semantic Attribute Reconciliation Architecture), the middleware that X2Photo is built on, is described briefly next.

#### **3.1 SARA**

SARA is a domain independent framework that allows for low level metadata to be aggregated into semantically meaningful characteristics that ordinary users can

understand. These characteristics (called semantic attributes) are defined by experts, and then leveraged by end users to help their exploration of a domain. The semantic attributes can be objective or subjective in nature, and SARA can support the tailoring of these characteristics to an end users perspective or context. By adding these semantically meaningful concepts to a space that didn't have them before, it supports end-users (via an appropriate client application) to employ expert knowledge to create high-level, semantically meaningful queries over multiple sources from a domain. Essentially, these semantic attributes can be seen as generalised rules for the domain and SARA acts as a semantic mediator between end-users and the raw data sources they seek to explore. Importantly, these semantic attributes can be generated by non-technical domain experts without the help of a knowledge engineer by using SARA's authoring tool SABer (Semantic Attribute Builder). This means that semantic attributes can be generated by experts from almost any domain.

SARA has already been successfully applied to a number of domains including music, films, digital humanities and publications. Hence, its support for subjective semantic attributes, based on aggregated low level data, meant it was an ideal system to help the exploration of image repositories from an aesthetic perspective. The main aim of X2Photo is to use the functionality offered by SARA to help users browse large image repositories with reference to the aesthetics of the photographs, as well as their content. Specifically it supports the retrieval of Flickr photographs using domain expertise in aesthetics, as well as user generated tags. In order for X2Photo to use SARA it needed an expert vocabulary to describe the aesthetics of digital photography. This vocabulary would then be leveraged by end-users to give them a consistent approach to browsing for images. The next section describes why a vocabulary based on colour psychology and colour theory was chosen for X2Photo.

### 3.2 Colour Theory and Colour Psychology as an Aesthetic Vocabulary

Within CBIR (Content Based Image Retrieval) approaches, colour has been seen as a key feature to characterise the content of digital content collections [11-13]. Common colour features include, colour-covariance matrices, colour histograms, colour moments, and colour coherence vectors. Even though these colour features are efficient in describing colours, they are not directly related to high-level semantics. Hence, one way to derive human perception through colours is to investigate the psychology of colour in art [14, 15]. Artists use colour to explore visual perception and to represent or evoke emotions. The psychological effects of colour, hue, saturation, and brightness have been studied to reveal having various effects on the viewer [16, 17].

Complementary to colour psychology is colour theory, which is a language that conceptually and perceptually describes the essentials of colour and their interactions [18]. Unlike colour psychology, colour theory doesn't describe responses that are unique to cultures or certain periods, but rather focuses on universal psychological responses to colour. An example would be the warmth or coolness of a colour, i.e. the temperature. Colours such as blue and green are cool colours and can be thought of as having calming effects. However, this effect can transform as the colour's luminosity changes, i.e. a bright open sky may be exciting. Likewise, cool colours on one end of the scale can be seen as cold, impersonal, and gloomy but on the other comforting and

nurturing. In photography, colour theory is utilised to understand how certain colours and their combinations create different moods in photographs. For instance, some colour combinations such as complementary colours appear striking and vibrant when in close proximity. Furthermore, as they get closer to the same saturation and lightness, the vibrant look will strengthen. On the other hand, colours that are close in the spectrum will usually appear more peaceful and calm.

X2Photo required an expert vocabulary based on raw low-level data of digital photographs. By extending the language that colour theory provides with subjective concepts from colour psychology, a consistent vocabulary was developed that helped the exploration of photographs from an aesthetic perspective. The user could adapt to the expert's perspective or find it open to questioning. However the important factor here was not the vocabulary, but rather providing a base that a subjective concept could be built upon and if necessary personalised.

### 3.3 Design

Based on the information in the previous section it was decided that for representing colour in digital images, the HSL (Hue, Saturations, Lightness) colour model would be the most efficient regarding this research's aims. HSL colour space describes perceptual colour relationships more accurately than RGB and is far more intuitive. Fortunately, this kind of metadata is commonly found in digital images or can be easily extracted from the photograph. HSL colour space is also more closely related to human visual perception. Another point deducted from colour theory was that the human eye is more sensitive to hue than saturation and lightness. Therefore hue should be processed with a finer quantisation.

The expert vocabulary created for X2Photo consisted of nine semantic attributes, each with three or four parameters that were encoded by the domain expert in SABer (see Table. 1). *Temperature* was one such semantic attribute for the photography domain created, and this was quantised into a number of different parameters ranging from *Warm* to *Cold*. Thus *Cool* was a single parameter of the semantic attribute *Temperature*. When *Temperature* was defined, the hue of the colour was taken into consideration as follows. Colour theory defines colours such as blue and green as cool colours with red and orange defined as warm. Hue is represented as an angle of the colour circle. So if it is divided into twelve equal intervals, on each 30° angle, the following colours result; red, red-yellow (orange), yellow, yellow-green, green, green-cyan, cyan, cyan-blue, blue, blue-magenta, magenta, and magenta-red. Red, yellow, green, cyan, blue, and magenta are regarded as the key colours with each having intermediate colours in between. Thus, the classification for *Temperature* into four parameters was constructed as follows, where H, S and L represent hue, saturation and lightness respectively:

$$\begin{aligned} \text{WARM} &= \{(0 \leq H < 75 \text{ and } 15 \leq L \leq 90) \text{ or } (H \geq 300 \text{ and } 65 \leq L \leq 90)\} \text{ and } (S \geq 25) \\ \text{SUBTLE} &= (75 \leq H < 120) \text{ and } (15 \leq L \leq 90) \text{ and } (S \geq 25) \\ \text{COOL} &= (120 \leq H < 210) \text{ and } (15 \leq L \leq 90) \text{ and } (S \geq 25) \\ \text{COLD} &= (210 \leq H < 300) \text{ and } (15 \leq L \leq 90) \text{ and } (S \geq 25) \end{aligned}$$

If a photograph's colour space satisfies the third equation, it is considered as a *Cool* photograph. Table 1 lists the nine semantic attributes created for X2Photo, each with

three or four different parameters. These subjective semantic attributes were created in a similar way to the *Temperature* example described above, and were also joined by one objective semantic attribute named *has tag called* that allowed users to specify tags that the end images should have. All the semantic attributes created were listed in the X2Photo interface so that they could be joined together into a complex query by end users e.g. *Return all images that are Calm, Cool and Misty that has a tag called Boat or Fisherman.*

**Table 1.** The nine semantic attributes and their parameters created for X2Photo

<b>1. Power</b>	<b>2. Passion</b>	<b>3. Energy</b>	<b>4. Joy</b>	<b>5. Ease</b>
Vigourous Powerful Robust Strong	Passionate Desirous Romantic Sensitive	Explosive Exciting Energetic Lively	Frantic Ecstatic Jolly Cheerful	Easeful Content Mellow
<b>6. Light</b>	<b>7. Blue</b>	<b>8. Temperature</b>	<b>9. Purity</b>	
Luminous Misty Deep	Tranquil Calm Soothing	Warm Subtle Cool Cold	Intricate Bold Innocent Pure	

### 3.4 Implementation

The scope of this research was to develop a prototype system using a local database of images, but implement it using technologies that could be easily adapted for an online environment. With this in mind, the Flickr photograph collection was chosen in order to build a local cache of images. This section describes the implementation of X2Photo.

#### 3.4.1 Data Collection

This research's experimental approach called for sets of arbitrary photographs to be collected with no distinct styles. Therefore, photographs needed to be cached from a large number of users. The ideal way to realise this requirement was to query Flickr for a list of public photos. Using this approach, more than 12,000 random photographs from Flickr were cached in small, medium, and large sizes. The Flickr API was also used to retrieve the metadata related to the cached photographs. Once the necessary data for each photograph was parsed and stored, a tags repository was created based on unique tags within the collection. A list of tags related to the given tag, based on clustered usage analysis within Flickr, was also stored. At this stage it was realised that the metadata from Flickr alone was not extensive enough to capture an image's aesthetics. Only objective concepts such as when and where the photograph was taken and whether it was an indoor or outdoor image could be derived.

To create an aesthetic vocabulary that supported colour theory required more metadata about the images to be obtained. This meant that each digital photograph's pixel values were analysed in order to get its dominant tones and colours. Red, green and blue values for each pixel were then extracted and within each block these values

were rounded to their contextual RGB values to avoid almost duplicate colours. A photograph’s hue had to be processed with a finer quantisation as human perception is more sensitive to hue than saturation and lightness. Once an image was processed all its data was combined into a uniform model. This schema was then registered with SARA so that semantic attributes could be formed from them, and so that this media repository could be linked to other ones if so desired.

**3.4.2 User Interface and Architecture**

Figure 1 shows the front-end of the system in which the three main areas of the interface can be seen. The main part of the screen is called the Discovery Space and contains the result set of photographs from a user query. The wall of photographs can be dragged by user and individual images selected to see what tags and semantic attributes are associated with it.

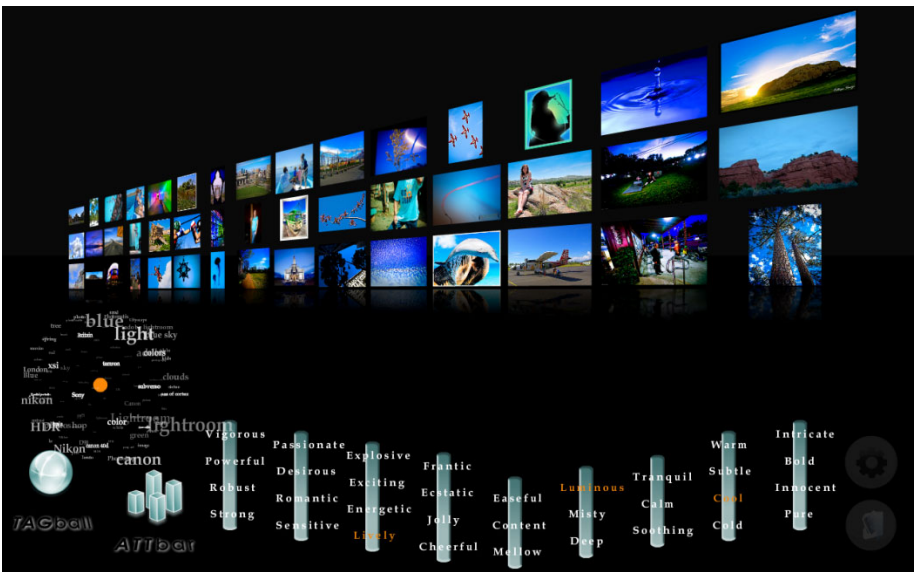


Fig. 1. The X2Photo Interface

The bottom of the screen is dominated by the AttBar which represents each of the nine semantic attributes as a vertical bar. Each bar contains the different parameters relating to each semantic attribute, with Figure 1 showing the results from a query containing the aesthetics *Lively*, *Luminous* and *Cool* from the AttBar. The user can select a parameter from each bar to add a query. Using SARA’s facility for tailoring of domain expertise, it would also be possible for end-users not happy with the results they were getting to alter the rules associated with each semantic attribute, so that a different range of images were returned for that particular parameter.

The single dimensional nature of tags has already been discussed, highlighting that they are limited in communicating the aesthetic values of photographs, but are instead more useful in defining their content. Thus, in order to help the user find an image

with specific content, the system had to show any tags associated with the result collection of photographs, as well as those from each individual photograph. By integrating this with support for aesthetic exploration of images, it gives users a more flexible way of finding relevant photographs. The number of tags typically exceeds a number that could be clearly displayed with a simple tag clouds, hence a TagBall was used instead to allow large numbers of tags to be displayed while not cluttering the UI. In the bottom left hand corner of Figure 1 is the TagBall which displays all the Flickr tags related to the entire result set or individual photograph. The user just has to select any of these to refine their searches.

When users click on an image, the interface zooms into the photograph, and they can click the flip button on the top left corner of the photograph to see its details. If a user clicks the down arrow icon, the photograph is brought into focus, with the AttBar displaying its associated semantic attributes, and the TagBall displaying the relevant tags. A user can store the photograph to the Favourites area by clicking the star button. Figure 2 shows the overall architecture of the application. A user forms queries by selecting semantic attributes in the X2Photo GUI, with all queries sent to SARA via its parameter based API. Because the metadata relating to the images are stored as XML, XQueries encapsulated within each semantic attribute are sent to the database storing the metadata and the relevant image identifiers returned for each semantic attribute. These separate result sets (if the query contains more than one semantic attribute) are then reconciled into a consolidated set which is returned to X2Photo in XML. The corresponding images from the photograph cache are then rendered to the user in X2Photo's GUI.

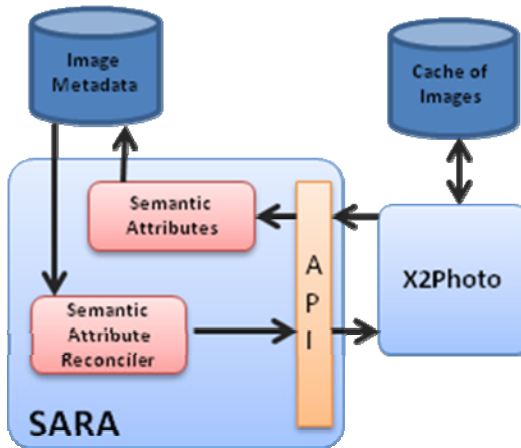


Fig. 2. X2Photo Architecture

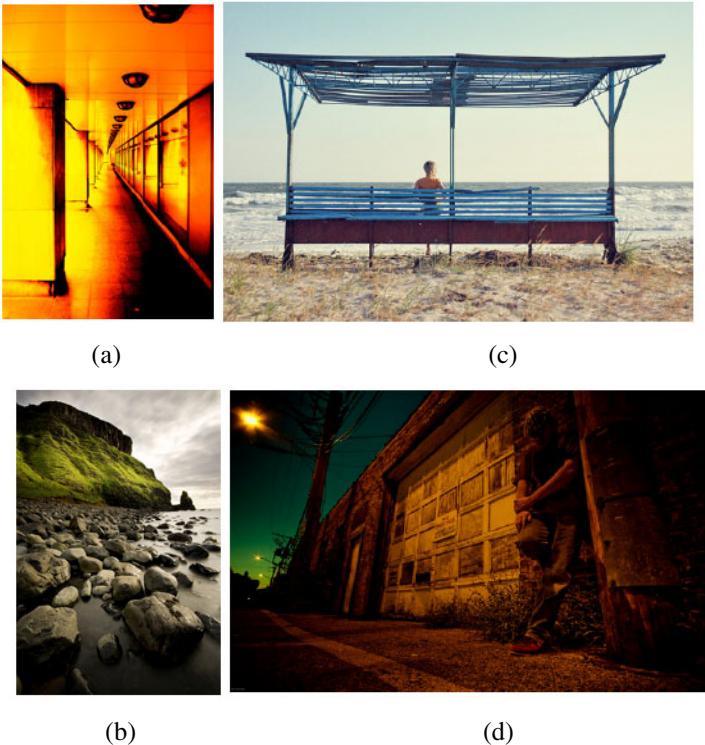
## 4 Evaluation

X2Photo was evaluated to test the usability, functionality and the overall appeal of the system. Furthermore, the potential benefits of injecting subjective expert knowledge

based on the manipulation of non-textual, low-level data, in comparison to conventional methods (like image retrieval via tags only) was also examined. With this aim in mind the following approach was pursued: Four photographs (see Figure 3) not present in the 12,000 random photographs collection were selected. These photographs were then prescribed different semantic attributes by the domain expert, who was an experienced photographer. The nine users were first shown all four photographs, and were then asked to freely describe them in their own words. Then they were given an overview of the tool and were asked to do the following tasks:

- For photograph 1, find similar photographs via X2Photo
- For each photograph found, add it to the Favourites.
- Repeat this task for all four photographs.
- Once complete, go to Flickr and for photograph 1; again try to find similar images either with the words originally used to describe the photographs or with different ones.
- Repeat this task for all four photographs.

After finishing these tasks they were given a survey to fill out, to complete the user-test.



**Fig. 3.** The four initial photographs shown to users



#### 4.1 Describing the Images

The majority of users evaluating X2Photo were technically proficient with computers, and four considered themselves to be amateur photographers. How the different users described the four photographs had some noteworthy aspects, such as those users who were interested in photography tending to use more technical phrases. For instance, some wrote terms such as “over-exposed” when describing the photograph *a*, mentioned the angle at which the photograph *b* might have been shot at, and questioned whether this photograph was altered in an image editing program to obtain its deep contrast. These users tended not to describe the content of the photograph as much as the users with little photography experience. Some users preferred to describe the photographs with more personal expressions such as “lonely” and “tempting” when referring to photograph *c*. Photograph *d*, as expected, was interpreted differently by almost all the users. While some tried to figure out what the man in the picture might be doing, some chose to describe him, resulting in many different impressions such as “gritty”, “relaxed” or “run-down”.

Almost all the users first chose expressions like “warm”, “cold”, “airy”, “gloomy” and “energetic”, some of which directly coincided with the actual attributes determined by the domain expert. They then proceeded to describe the actual content. Two of the nine participants were more objective in their descriptions and chose to name the elements they saw in the photographs with words like “corridor”, “bench”, “rocks”, “back alley”, etc. However, the vast majority of users combined their perceptions with the content: “...a cool calm picture but alive...there’s a woman sitting on a bench... feels breezy but soft... waves look relaxing”.

#### 4.2 Finding Images in X2Photo

Just like their preferences in describing the photographs, the users’ approach to finding similar photographs in X2Photo were particularly different. Four users never actually enabled the TagBall. Coincidentally their descriptions of the photographs were heavily consisted of expressions like “moody”, “dark”, “calm”, etc. They directly chose similar words present within the AttBar and then carried out their searches. After receiving their initial results two of these users were surprised to see how the tool interpreted their descriptions. They did not agree with the expert and started experimenting with the AttBar rather than continuing with their searches. After observing some consecutive result sets and bringing some photographs into focus, they stated that they grasped the association the expert was making, and modified their searches accordingly. The other two users who didn’t use the tag ball performed 2-3 consecutive searches which were refined each time, to find a similar photograph. Observing the similar photographs that users returned, it was interesting to see what the users based their similarity criteria on. While some photographs have a similar feel to them regarding the concept or the context, some are similar in content as well. Figure 4 shows examples of similar pictures (of images *b* and *c* in Figure 3) found by users using the TagBall and AttBar in X2Photo.

### 4.3 Finding Images in Flickr

When the users tried to find similar photographs in Flickr, their approaches were again different. For example, one user used “fiery clinical harsh” to search for the photograph a, which were the expressions he had used when describing the photographs originally. In contrast, another abandoned their expressive vocabulary used originally to describe photograph b (because he was very familiar with searching on Flickr) and chose to use the search phrase “Scotland cliff coast”. Some users were very articulate in their searches and submitted phrases such as “city lights low angle journalistic lonely” to find similar photographs. With three user’s searches within Flickr, a slight change in their vocabulary could be seen. For instance, a user who had previously described the photograph c mainly based on content; “beach, person sitting on the bench, greyish” found a similar image within X2Photo that the expert thought to be “romantic”, “soothing” and “innocent”. Within Flickr, the user thus carried out his first search with the terms “romantic sea scenery”. Users familiar with Flickr also used the advanced search available and refined their queries, but again tended to use content-based terms to carry out their searches. In the end all the users were able to find at least one similar image, which was not surprising considering the amount of photographs Flickr has. However it was noteworthy how all the users had to resort to content-based terms (identical in many cases). This showed how such systems can limit the ways individuals search for photographs.

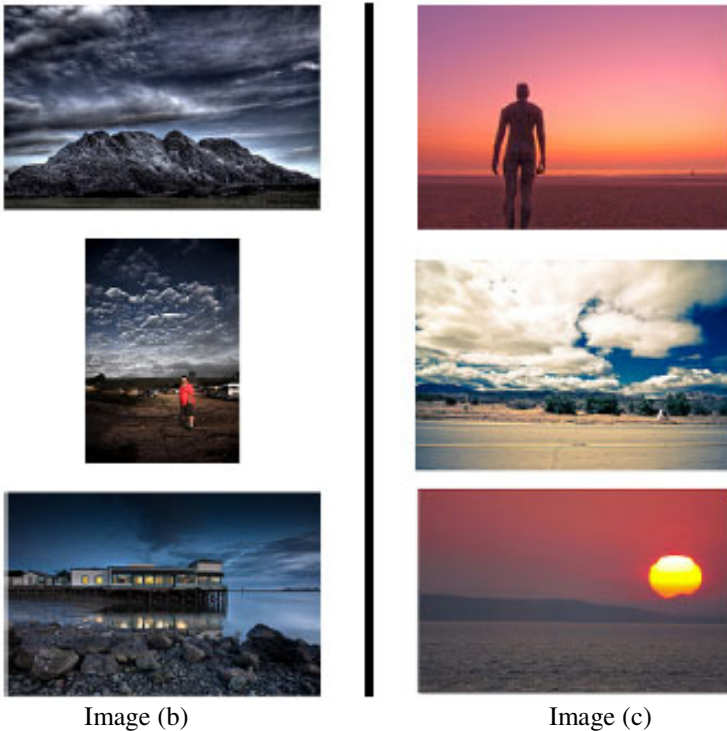


Fig. 4. Examples of similar pictures found by users using the TagBall and AttBar in X2Photo

#### 4.4 User Survey

A user survey was also conducted once the given tasks were completed. The questionnaire intended to evaluate each feature's functionality as well as aesthetic qualities, and also the overall system quality regarding various aspects. The general response to the usability and appeal of the Discovery Space was very positive, agreeing that the continuous flow enabled them to browse the photographs thoroughly, and that the interaction with the space was appealing. Below is a summary of the survey results:

- 8/9 users considered the overall UI to be very good
- 8/9 strongly agreed that the system was attractive
- 8/9 users found the zoom effect in the interface to be very good or good
- 9/9 users thought the AttBar was very good or good, that the concept was comprehensible and the classification of the attributes were clear.
- 8/9 users found the ability to refine a search with a focus image to be very useful or useful. By seeing what semantic attributes and tags were associated with the focus image it allowed them to use these as a springboard for their browsing.

#### 4.5 Analysis

The user experiment and the survey that followed suggested that when describing photographs, whether interested in photography or not, people like to communicate “how” a photograph is as well as “what” it portrays. This finding indicates a need for a wider vocabulary to be available to users in order to retrieve accurate and relevant photographs from any collection. Traditional tag-based systems tend to be dominated by content-based terms, thus ignoring the artistic quality which is a key factor that evokes appreciative emotions. Hence these systems often reduce photographs to a list of mainly content-based words. As most people have become accustomed to this approach, in such an environment they tend to ignore other ways in which they would approach a photograph, and are therefore relegated to search for the tagged simplification of a photograph, rather than the actual photograph itself.

Based on the photographs found by the users when using the more natural expressions via the X2Photo system, it indicated that this approach could grant users the additional useful axes to when searching for photographs. Thus, injecting expert knowledge, based on the manipulation of raw low-level data, into a conventional system only supporting tag-based search, allows users to more freely express both the photograph and the picture it is conveying. Even though a specific expert vocabulary may not be suitable or correct for each individual, users can adapt to the expert's view or better yet choose to subscribe altogether to a different expert expanding the semantic space. SARA also provides the functionality for end users to tailor an expert's semantic attribute to better fit their own vocabulary.

X2Photo received overall positive feedback; the users clearly understood the idea and the overall concept. They suggested that users should be able to subscribe to different experts and that there should be a more comprehensive range of semantic attributes. This all indicates that the users understood the aim of the tool and how it could be extended further. All users agreed that they could see a real-life application

of the tool if some further improvements were made and they could see themselves utilising such a tool in their everyday lives. In order to offer users an alternative way of finding a photograph, a system has to have a rich vocabulary. Hence, by increasing the range of low level features, it would enable experts to create more refined semantic attributes, resulting in a more useful system for end users.

## 5 Summary

This paper investigated the possible benefits of augmenting the conventional tag-based query techniques used in many image databases, with subjective expert knowledge built on raw low-level data. Based on this notion, X2Photo was developed which aimed to empower users in retrieving photographs from collections, using not only objective tags from Flickr, but also subjective expertise based on a photograph's colour space. Semantic attributes were encoded into the system via SABer, which provided end users with semantically meaningful access points into the domain, which were not previously available.

The user test and the results of its accompanying survey, highlight how people like to communicate the aesthetics of a photograph as well as what it portrays. However, when utilising a conventional tag-based system, they tend to ignore the aesthetics and emotions conveyed in the images, as the tag-based systems tend to be overloaded towards content based tags. Hence this can lead to limited searching via tagged simplifications of a photograph, ignoring the aesthetics of the photograph. The types of photographs found by the users, using the more natural expressions offered by the system, indicate that this approach can be used to grant users more versatility when searching for photographs. Hence, injecting expert knowledge into a conventional system that only offers tag-based searching, would allow users to freely express both the aesthetic of the photograph they want, as well as the picture it conveys. This offers users an alternative pathway to access large photograph collections.

All the users agreed that the system was a powerful tool for exploring photographs and when asked if they could see a real-world application stemming from X2Photo, all users concurred, as long as further improvements were made. Some engineering decisions need to be reconsidered in order to offer a more robust system and an alternative approach to manipulating the tags would be beneficial. The number of semantic attributes could also be increased, and the extraction of the underlying features improved with more sophisticated image analysis techniques. This would enable experts to create more refined semantic attributes. Moreover as a specific expert vocabulary may not be suitable or correct for each individual, users should be able to subscribe to different experts to consider different perspectives.

Leading image search engine such as Google Images have recently provided a few colours to be selected in order to have results with similar colour spaces. Considering the huge volume of images they index, and this new functionality they offer, it can be suggested that the methodology proposed in this paper could be integrated seamlessly into online image searching. This new functionality would allow users to pose verbal queries rather than selecting some basic colours to match. This approach could be also applied to other media such as video and audio, with SARA supporting experts in those fields to classify characteristics that end users could leverage in their searches.

Likewise, multiple experts from the same domain can be supported by SARA, with the end user able to select characteristics created by different experts and tailoring them to their own needs if necessary. Finally, because SARA provides a consolidated interface to multiple sources from a domain, it can support applications that give users powerful searching and browsing operations over many separate image repositories.

**Acknowledgments.** This research has been supported by The Irish Research Council for Science, Engineering and Technology: funded by the National Development Plan.

## References

1. Cui, J., Wen, F., Tang, X.: Real time google and live image search re-ranking. In: Proceeding of the 16th ACM International Conference on Multimedia, pp. 729–732 (2008)
2. Jackson, R.S.: Wine tasting: a professional handbook. Elsevier, Amsterdam (2002)
3. Hampson, C., Conlan, O.: Leveraging Domain Expertise to Support Complex, Personalized and Semantically Meaningful Queries Across Separate Data Sources. In: Proceeding of the Fourth IEEE International Conference on Semantic Computing (ICSC 2010), Pittsburgh, USA, pp. 305–308 (2010)
4. Hare, J.S., Lewis, P.H., Enser, P.G.B., Sandom, C.J.: Mind the gap: Another look at the problem of the semantic gap in image retrieval. In: Multimedia Content Analysis, Management, and Retrieval 2006, vol. 6073, pp. 75–86 (2006)
5. Enser, P.G.B., Sandom, C.J., Lewis, P.H.: Surveying the reality of semantic image retrieval. In: Bres, S., Laurini, R. (eds.) VISUAL 2005. LNCS, vol. 3736, pp. 177–188. Springer, Heidelberg (2006)
6. Datta, R., Li, J., Wang, J.Z.: Content-based image retrieval: approaches and trends of the new age. In: Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval, pp. 253–262. ACM, New York (2005)
7. Tamura, H., Yokoya, N.: Image database systems: A survey. *Pattern Recognition* 17, 29–43 (1984)
8. Shen, H.T., Ooi, B.C., Tan, K.L.: Giving meanings to WWW images. In: Proceedings of the 8th ACM International Conference on Multimedia, pp. 39–47. ACM, New York (2000)
9. Cai, D., He, X., Li, Z., Ma, W.Y., Wen, J.R.: Hierarchical clustering of WWW image search results using visual, textual and link information. In: Proceedings of the 12th Annual ACM International Conference on Multimedia, pp. 952–959. ACM, New York (2004)
10. Marlow, C., Naaman, M., Boyd, D., Davis, M.: Position paper, tagging, taxonomy, flickr, article, to read. In: Collaborative Web Tagging Workshop, Edinburgh, Scotland (2006)
11. Gong, Y.: Advancing content-based image retrieval by exploiting image color and region features. In: *Multimedia Systems*, vol. 7, pp. 449–457. ACM, New York (1999)
12. Yu, H., Li, M., Zhang, H.J., Feng, J.: Color texture moments for content-based image retrieval. In: Proceedings of the International Conference on Image Processing, pp. 929–932 (2002)
13. Shih, J.L., Chen, L.H.: Color image retrieval based on primitives of color moments. In: Chang, S.-K., Chen, Z., Lee, S.-Y. (eds.) VISUAL 2002. LNCS, vol. 2314, pp. 88–94. Springer, Heidelberg (2002)
14. Davis, S.: Color perception: Philosophical, Psychological, Artistic, and Computational Perspectives. Oxford University Press, Oxford (2000)

15. Gage, J.: *Color and Meaning: Art, Science, and Symbolism*. University of California Press, Berkeley (1999)
16. Fehrman, K., Fehrman, C.F.: *Color: The Secret Influence*. Prentice-Hall, Englewood Cliffs (2000)
17. Valdez, P., Mehrabian, A.: Effects of color on emotions. *Journal of Experimental Psychology* 123, 394–408 (1994)
18. Parramon, J.: *Color Theory*. Watson-Guption Publications, New York (1989)
19. Sinha, P., Jain, R.: *Semantics In Digital Photos A Contextual Analysis*. In: *Proceeding of the Second IEEE International Conference on Semantic Computing*, pp. 58–65 (2008)
20. Enser, P.: Visual image retrieval: seeking the alliance of concept-based and content-based paradigms. *Journal of Information Science* 26, 199–210 (2000)

# SkyMap: A Trie-Based Index Structure for High-Performance Skyline Query Processing

Joachim Selke and Wolf-Tilo Balke

Institut für Informationssysteme  
Technische Universität Braunschweig  
Braunschweig, Germany  
{selke,balke}@ifis.cs.tu-bs.de

**Abstract.** Skyline queries have become commonplace in many applications. The main problem is to efficiently find the set of Pareto-optimal choices from a large amount of database items. Several algorithms and indexing techniques have been proposed recently, but until now no indexing technique was able to address all problems for skyline queries in realistic applications: fast access, superior scalability even for higher dimensions, and low costs for maintenance in face of data updates. In this paper we design and evaluate a trie-based indexing technique that solves the major efficiency bottlenecks of skyline queries. It scales gracefully even for high dimensional queries, is largely independent of the underlying data distributions, and allows for efficient updates. Our experiments on real and synthetic datasets show a performance increase of up to two orders of magnitude compared to previous indexing techniques.

## 1 Introduction

Skyline queries, introduced in [2], quickly found its way into a widespread range of data management applications. Soon after the first skyline algorithms have been presented, emerging fields like e-shopping or location-based services used the economic intuitiveness of Pareto-optimal result sets, e.g., for deriving all reasonable product alternatives [15][8]. The basic idea is to filter out all those items (database tuples) dominated in terms of usefulness by other items, i.e., there is no utility function declaring a dominated item as best choice. For example, when trying to find an inexpensive hotel close to the beach, overpriced inland hotels can safely be excluded.

In a brief period of time, several important problem settings building on the original skyline paradigm have been identified and individual solutions have already been proposed, e.g., [3], [14], or [11]. However, all these algorithms rely on specialized techniques, whereas the widespread applicability of skyline queries also raises the question of how these queries can be answered efficiently in general-purpose database systems. This especially sparked interest in the use of indexing techniques to boost skyline query performance (e.g., [11], [6], [16], or [9]). Indexes are indeed the key for broad database support for efficient skyline computation. In fact, [9] shows that a wide variety of special skyline queries ( $k$ -dominant skylines, skybands, subspace skylines, etc.) can be supported using a single index structure. But although indexes can dramatically speed up

retrieval, they of course also introduce maintenance costs and tend to quickly degenerate on higher dimensional data. *The task thus is to design a robust index structure that (1) enables high-performance skyline processing, (2) is easy to maintain, and (3) gracefully scales with data dimensionality.*

In this paper we propose SkyMap, an innovative trie-based index structure for skyline query support. In a nutshell, SkyMap adopts a recursive grid-like space partitioning approach, which facilitates efficient navigation. In contrast to traditional data-driven space partitioning with the inherent danger of degeneration over time, SkyMap takes the problem domain into account. Since skyline computation builds on ranks rather than absolute scores, any data set can be transformed spreading data evenly over each dimension. Thus, an efficient data independent space partitioning scheme without degeneration problems can be used, and the trie generally stays balanced. In particular, we will show that for each data set there exists only a single unique SkyMap making the resulting trie structure independent of the order in which data is inserted or deleted.

Moreover, data is only contained in the trie's leaf nodes, which fosters efficient maintenance operations. Especially, expensive rebalancing operations are avoided. Navigation within the SkyMap index is particularly efficient by relying on inexpensive bitwise operations only. The index also closely controls each node's fan-out by limiting the split factors, to optimize the trie's depth-to-width ratio. Thus, the index's degradation into a linear list is effectively prevented.

We extensively evaluated SkyMap on different data sets and compared its performance to the current state-of-the-art indexing schemes: approaches based on quadtrees (OSP-SPF [16] and BSkyTree [7]) and on UB-trees (ZB-tree [9]). Although these techniques show their strength in different aspects (higher pruning power vs. improved maintenance), SkyMap outperforms them on all counts. We show that our trie-based indexing scheme gains an order of magnitude in query performance across the board, for skyline maintenance even up to two orders of magnitude.

## 2 Related Work

The investigation of skyline processing started out with list-based approaches like the well-known BNL algorithm [2] and at first focused on algorithmic aspects and sophisticated heuristics like presorting [4] or limiting [1]. However, the necessity for efficient indexing schemes was soon recognized. Indexes allow for efficient pruning, hence large parts of the database can immediately be excluded from costly dominance tests. The first approaches featured efficient bitmap indexes [5] and R-trees with nearest neighbor search [6] or branch-and-bound pruning [11]. The major problem with these approaches was that the underlying data structures did not lend themselves readily for adoption to skylining problems. Since overlapping regions in multi-dimensional index structures proved to pose severe performance penalties, recent approaches focus on strictly disjoint space partitionings or efficient one-dimensional indexing using space-filling curves.

The ZB-tree [9] is based on the UB-tree, where each data point is mapped to its Z-address, which in turn provides a one-dimensional key for B-tree indexing. The benefits of this approach for skyline processing are twofold. On one hand, through the Z-addresses the B-tree imposes a presorting on the data, which can be exploited for dominance tests: No database item can dominate any item having a lower Z-address.



On the other hand, the regions covered by each tree node can easily be estimated (upper bound) without keeping track of minimum bounding rectangles. A major drawback of the ZB-tree approach is that regions may overlap, which hampers effective pruning. Moreover, the maintenance of B-trees is rather expensive in case of frequent updates, in particular due to rebalancing operations caused by node underflows. Still, the actual degeneration of the index structure over time is not a problem.

OSPSPF [16] and the B-SkyTree [7] focus on the problem of overlapping regions and improve performance by disjoint space partitioning. Both approaches rely on quadrees and basically are alternative implementations of the same idea. Each tree node stores a single data item, which is used to split the underlying space with respect to each dimension. The actual performance of this data-driven partitioning scheme is strongly dependent on finding optimal splitting points. While a careful initial bulkloading will lead to a perfectly balanced tree structure, frequent updates cause the index' performance to deteriorate quickly. Moreover, since all nodes store actual data, deletions either ruin cache efficiency when performing lazy deletions or force expensive reorganization operations.

In summary, we are forced to conclude that there is no single index structure offering superior pruning power, easy maintenance, and graceful scaling.

### 3 Preliminaries

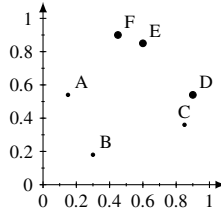
Henceforth, we consider a set of  $n$  database items, where each item's utility can be evaluated with respect to scorings over  $d$  criteria. Without loss of generality we assume that each score lies within the interval  $[0, 1)$  and that higher values are better. Then, the database can be represented as a set  $A \subset [0, 1)^d$  of size  $n$ .

Skylines captures the intuitive idea of Pareto optimality. Formally, given two points  $x = (x_1, \dots, x_d)$  and  $y = (y_1, \dots, y_d)$ , the point  $x$  is said to *dominate*  $y$  (denoted by  $x \succ y$ ) if and only if  $x \neq y$  and  $x_i \geq y_i$ , for each dimension  $i$ . Furthermore,  $x$  and  $y$  are said to be *incomparable* (denoted by  $x \parallel y$ ) if and only if neither  $x = y$ , nor  $x \succ y$ , nor  $x \prec y$ . We write  $x \succ_i y$  if  $x_i > y_i$ . The skyline of a data set consists of exactly those items that cannot be ruled out by means of Pareto dominance. Formally, the *skyline* of a point set  $A$  is the set  $\mathcal{S}(A) = \{x \in A \mid \text{there is no } y \in A \text{ such that } y \succ x\}$ . Characteristic properties of a data set  $A$  are its dimensionality  $d$ , its cardinality  $n$ , and its skyline size  $s := |\mathcal{S}(A)|$ .

As a running example, consider the data set depicted in Fig. 1. Each point corresponds to a database item, scored with respect to two different criteria. We see that items A, B and C can safely be removed from further consideration since each is dominated by some other item (A is dominated by F, E, and D; B is dominated by F, E, and D; and C is dominated by D). The remaining items D, E, and F form the skyline.

### 4 The SkyMap Approach

The design goals for a skyline-centered indexing technique can be divided into two categories: boosting performance and minimizing maintenance overhead. For *boosting performance*, the index structure should be well-balanced, avoid overlaps between data regions, and control the nodes' fan-out degree to prevent degradation into a linear list. For *minimizing the maintenance overhead*, all data should be collected in leaf nodes,



**Fig. 1.** Our running example

whereas all internal nodes should be purely navigational. Expensive reorganizations of the index structure must be avoided.

Although these goals sound contradictory, we show that it is indeed possible to implement them within a single data structure. Looking at the overlap problem, it quickly becomes clear that disjoint space-partitioning is needed. Combining this partitioning with our requirement of leaf-only data storage while minimizing reorganizations definitely calls for a data-independent partitioning scheme. Of course, such a data-independent partitioning would hurt the index structure’s balancedness whenever the data distribution is skewed in any dimension. But can we guarantee evenly spread data for skyline computations? Interestingly, we can.

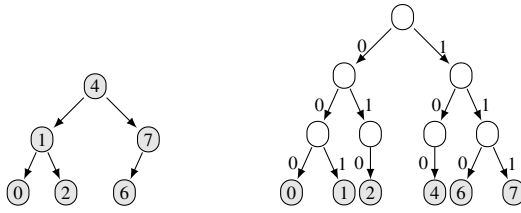
This is because the notion of Pareto dominance, and thus the base of skyline computation, does only rely on the actual score values in so far as they introduce a dominance ranking between database items. That means: As long as the relative ranking with respect to each dimension stays intact, we may arbitrarily modify all absolute score values. In particular, we may apply a transformation to each individual dimension that spreads database items evenly. This class of mathematical transformations is known as *copulas* [10]. Copulas are computationally inexpensive and are typically applied for correlation analysis. The following (intuitive and easy-to-prove) lemma shows that computing the skyline of a copula-transformed data set will indeed yield the data set’s original skyline.

**Lemma 1.** *Let  $A$  be a  $d$ -dimensional data set,  $f_1, \dots, f_d : [0, 1] \rightarrow [0, 1]$  strictly monotonic increasing functions,  $f(x) := (f_1(x_1), \dots, f_d(x_d))$ , and  $f(A) := \{f(x) \mid x \in A\}$ . Then,  $x \in \mathcal{S}(A)$  if and only if  $f(x) \in \mathcal{S}(f(A))$ , for any  $x$ .*

#### 4.1 Tries

To satisfy our design goals, we chose to base the SkyMap index on tries. The term *trie* refers to all tree data structures that perform a data-independent disjoint space-partitioning and use (typically binary) strings for navigation [13].

To give an example, Fig. 2 illustrates the difference between a binary search tree (BST) and a 3-digit binary trie (3BT), when it comes to storing a given set of numbers. While there is a one-to-one mapping between nodes and data items in the BST, the 3BT differentiates between internal nodes, which are solely used for navigational purposes, and leaf nodes, which store the actual data. Moreover, the BST is constructed only by performing ordinal comparisons between numbers, whereas the 3BT exploits the



**Fig. 2.** A binary search tree and a 3-digit binary trie storing the set  $\{0, 1, 2, 4, 6, 7\}$

numbers’ binary representation. Finally, there are many different BSTs storing the data set shown in Fig. 2 whereas the 3BT is unique.

Quadtrees are the multidimensional extension of BSTs, while our SkyMap is a multidimensional extension of binary tries, which additionally provides efficient algorithms for dominance checks. The regular and predictable structure of tries provides high memory locality and enables efficient updates of the stored data set—two key ingredients of SkyMap. It also heavily exploits the binary representation of data points. Henceforth, we indicate score values written in binary by the prefix  $\mathcal{B}$ , e.g.,  $0.75 = \frac{1}{2} + \frac{1}{4} = \mathcal{B}0.11$ .

For easy presentation and without loss of generality, we assume in the following that all of  $A$ ’s points are distinct.

### 4.2 Z-Addresses, Z-Regions, and Z-Subregions

In Section 3 we assumed without loss of generality that all data points are located in  $[0, 1)^d$ . Therefore, the  $i$ -th coordinate value of the point  $x = (x_1, \dots, x_d)$  can be represented as  $x_i = \mathcal{B}0.\alpha_{i,1}\alpha_{i,2}\dots$ , where  $\alpha_{i,1}, \alpha_{i,2}, \dots \in \{0, 1\}$  are  $x_i$ ’s binary digits. By interleaving the bits of all of  $x$ ’s coordinate values, the *Z-address* of  $x$ , denoted by  $Z(x)$ , can be derived [12]:

$$Z(x) = \mathcal{B}0.\alpha_{1,1}\dots\alpha_{d,1}\alpha_{1,2}\dots\alpha_{d,2}\dots.$$

In general, the  $j$ -th bit of a Z-address is determined by the  $((j - 1) \operatorname{div} d + 1)$ -th bit of the  $((j - 1) \operatorname{mod} d + 1)$ -th coordinate value. For example, the Z-address of the three-dimensional point  $(0.125, 0.75, 0.5) = \mathcal{B}(0.001, 0.11, 0.1)$  is  $\mathcal{B}0.0110101$ .

When reversing the above situation, also each Z-address uniquely defines a  $d$ -dimensional point in the (upper open) unit hypercube. We will use this property to assign a rectangular region in space to each finite binary sequence as follows: Given a number  $y = \mathcal{B}0.\beta_1\beta_2\dots \in [0, 1)$  and a non-negative integer  $r$ , we define the  $r$ -th *Z-region* of  $y$ , denoted by  $ZR_r(y)$ , to be the set of all points whose Z-address (in binary) begins with the sequence  $\mathcal{B}0.\beta_1\beta_2\dots\beta_r$ . That is,

$$ZR_r(y) = \left\{ x \in [0, 1)^d \mid Z(x) = \mathcal{B}0.\beta_1\dots\beta_r\gamma_{r+1}\gamma_{r+2}\dots \right\}.$$

In general, any Z-region is a lower closed and upper open hyperrectangle.

Every Z-region can naturally be partitioned into smaller Z-regions as follows: Given a number  $y = \mathcal{B}0.\beta_1\beta_2\dots \in [0, 1)$  as well as two non-negative integers  $r$  and  $b$ , then for any  $\gamma_1, \dots, \gamma_b \in \{0, 1\}$ , the  $(r + b)$ -th Z-region of the number  $\mathcal{B}0.\beta_1\dots\beta_r\gamma_1\dots\gamma_b$  is

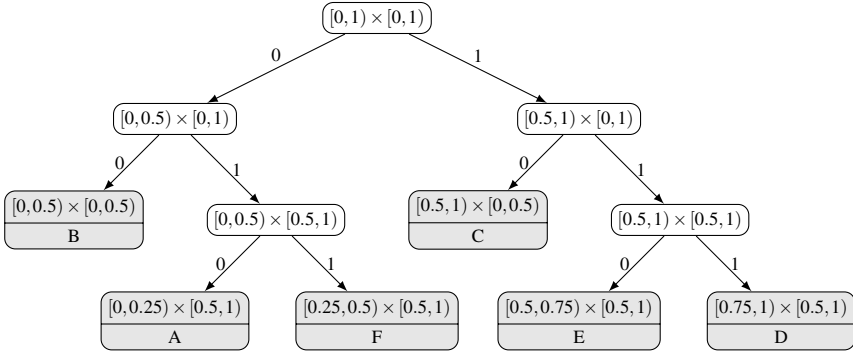


Fig. 3. A SkyMap ( $b = 1, c = 1$ ) indexing our example data set

a subset of  $ZR_r(y)$ . In total, there are  $2^b$  such subsets, which are mutually disjoint. This way, for any  $b$ , the Z-region  $ZR_r(y)$  can be divided into  $2^b$  partitions. In the following, we will refer to these partitions as  $ZR_r(y)$ 's Z-subregions of degree  $b$ . For example, in two-dimensional space, the degree-2 Z-subregions of the Z-region  $ZR_0(0) = [0, 1]^2$  are  $ZR_2(0) = [0, 0.5] \times [0, 0.5]$ ,  $ZR_2(0.25) = [0, 0.5] \times [0.5, 1]$ ,  $ZR_2(0.5) = [0.5, 1] \times [0, 0.5]$ , and  $ZR_2(0.75) = [0.5, 1] \times [0.5, 1]$ .

### 4.3 SkyMap and Its Basic Operations

In this section, we build the skeleton of SkyMap. For this task, we turn the concepts just presented into a recursive scheme for space decomposition.

A SkyMap index is a trie that consists of internal nodes and leaf nodes: *Leaf nodes* store a list of at least one but at most  $c$  data points, where  $c$  is some integer we will refer to as the *capacity* of a leaf node. We assume that all leaf nodes have the same capacity. *Internal nodes* store an array of exactly  $2^b$  pointers to its child nodes, which are indexed by the numbers 0 to  $2^b - 1$ , where  $b$  again is an integer parameter shared by all nodes. We will refer to  $b$  as the *split degree* of an internal node (measured in bits). Null pointers are generally allowed, but empty nodes are not permitted.

The SkyMap index has been designed to resemble the recursive splitting process of Z-regions into all its  $2^b$  Z-subregions of degree  $b$ . In particular, each node represents some Z-region, where the root node corresponds to the Z-region  $ZR_0(0) = [0, 1]^d$ .

Fig. 3 shows a normalized version of our example database and a corresponding SkyMap indexing it. We set  $b = 1$ , which means that the data space is recursively split along a single dimension each. The respective dimension changes from level to level. The capacity  $c$  is set to 1, i.e., each leaf node stores a single point. To illustrate the effect of these parameters, Fig. 4 shows an alternate SkyMap with  $b = 2$  and  $c = 2$ .

In contrast to quadtree-based approaches, where nodes may have up to  $2^d$  children, in a SkyMap index the number of children of each node can easily be controlled by setting the parameter  $b$  accordingly. That way, degradation of the trie structure to a linear list can be avoided in higher dimensional spaces. On the other hand,  $b$  can be chosen large enough to enable a most effective space division and pruning. Moreover,

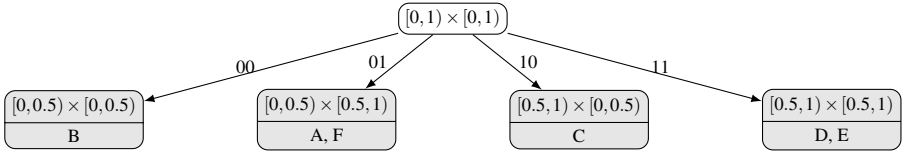


Fig. 4. A SkyMap ( $b = 2, c = 2$ ) indexing our example data set

by tuning the leaf capacity  $c$ , one can avoid excessive ramification of the trie but still receive the benefits of indexing, thus exploiting the full potential of memory caches.

The most important operations required for building and maintaining an index are *bulkloading* of entire data sets as well as *inserting* and *deleting* individual items. As we will show next, these operations can be performed highly efficient in a SkyMap index.

In SkyMap, bulkloading of a given data set  $A$  is performed in top-down fashion. After presorting the data set by Z-addresses, we recursively split the data set with respect to Z-address prefixes of length  $b$  and create internal nodes (if the number of remaining points is larger than  $c$ ) or leaf nodes (otherwise). During this process, we completely avoid expensive multi-dimensional point comparisons and only rely on cheap integer comparisons and bit-shifting operations. The complete algorithm is shown in Fig. 5.

To insert a new point  $p$  into an existing SkyMap (cf. Fig. 5), we traverse the trie structure according to  $p$ 's Z-address and add  $p$  to a matching leaf node. If the leaf node already is filled to its maximum capacity  $c$ , we replace it by a new internal node. Then, the insertion process continues at the new node. Again, we do not perform explicit point comparisons but only need to perform integer comparisons and bit shifts. Moreover, our insertion procedure does not require expensive rebalancing operations.

In a similar fashion, points can be removed from an existing SkyMap index: First, find the corresponding leaf node *leaf* and delete the point. In case *leaf* gets empty by this operation, also remove *leaf* from the index. If necessary, recursively repeat this cleanup process with all parental internal nodes. Moreover, the deletion algorithm keeps track of how many points are located below each internal nodes it visits during the cleanup process. If an internal node is detected that covers only  $c$  points or less, then this internal node is replaced by a corresponding leaf node, thus reducing the SkyMap's depth to the minimum possible value. Due to space limitations, we omit the pseudocode.

#### 4.4 Analysis

Having introduced the basic structure of the SkyMap index, we now demonstrate that it lives up to its promise. Recall the two main requirements: boosting performance and minimizing maintenance overhead. By designing simple insert and delete operations, which only affect a very small number of nodes and do not perform any expensive rebalancing or restructuring, we have already fulfilled our second requirement of easy maintenance. At the same time, due to the use of copulas for normalizing our data set (i.e., distributing it equally across each dimension; see Lemma 1), we are able to achieve a homogeneous distribution of data points across the whole SkyMap tree (given that data dimensions do not exhibit correlation or anti-correlation at an extreme degree).

```

function BULKLOAD(A)
  Presort A by Z-addresses; let P and Z be the resulting lists of points and Z-addresses
  root ← BULKLOAD(P, Z, 0, |A| - 1)

function BULKLOAD(P, Z, from, to)
  size ← to - from + 1
  if size ≤ c then return a new leaf node containing P[from, ..., to]
  else
    node ← a new empty internal node
    Z' ← a copy of Z, in which the first b bits have been removed from each entry
    pos ← from
    zbFrom ← the first b bits of Z[from]
    while pos ≤ to do
      zbPos ← the first b bits of Z[pos]
      if zbPos ≠ zbFrom then
        node.children[zbFrom] ← BULKLOAD(P, Z', from, pos - 1)
        from ← pos
        zbFrom ← zbPos
    node.children[zbFrom] ← BULKLOAD(P, Z', from, to)
  return node

function INSERT(p)
  if root = null then root ← a new leaf node containing only p
  else
    z ← p's Z-address
    INSERT(p, z, root, 0)

function INSERT(p, z, node, depth)
  zb ← the first b bits of z
  if node is an internal node then
    if node.children[zb] = null then node.children[zb] ← a new leaf containing only p
    else
      z' ← z without its the first b bits
      INSERT(p, z', node.children[zb], depth + b)
  else if node is a leaf node containing less than c points then Add p to node
  else
    node' ← a new empty internal node
    for each point q contained in node do
      qz ← q's Z-address without its first depth bits
      INSERT(q, qz, node', depth)
    INSERT(p, z, node', depth)
    Replace node by node' in the SkyMap index

```

**Fig. 5.** Creating a SkyMap by bulkloading and inserting a new point into an existing SkyMap

We now show that SkyMap also possesses two key ingredients of high-performance index structures: immunity against degradation due to database updates and logarithmic depth on average.

**Lemma 2.** *Let the parameters  $b$  and  $c$  be fixed. Then, for any data set  $A$ , there is a unique SkyMap indexing  $A$ , modulo sort order of points within leaves.*

*Proof.* Assume that there are two different SkyMap indexes storing the data set  $A$ . Since by design each point is always stored in a leaf node along its Z-address path, to be different modulo sort order of points within leaf nodes, one of the SkyMap indexes must contain a leaf node that is not present in the other one. Let  $N$  be this node. We assume without loss of generality that  $N$  is located within the first index. Since any point contained in  $N$  must also be present in some leaf node of the second index, there must be a leaf node  $M$  in the second index at the location of one of  $N$ 's parental internal nodes. As  $M$  can contain at most  $c$  points and points are placed in the index along their

Z-addresses, also the subtree rooted at the internal node corresponding to  $M$  in the first trie can contain at most  $c$  points, a situation which is explicitly avoided in the deletion procedure. Since unnecessary internal nodes can only be created by deletion tasks, this directly corresponds to our initial assumption.

The preceding lemma makes clear that there cannot be any degradation in a SkyMap index, no matter what sequence of insertions and deletions is performed. Our next lemma is about index behavior on deletions.

**Lemma 3.** *Let  $S$  be a SkyMap indexing some data set  $A$  and  $B \subseteq A$ . Then, any SkyMap indexing  $B$  only has nodes at those positions where there is a node in  $S$ .*

*Proof.* Due to the uniqueness property of SkyMap indexes, any index  $S'$  storing only a subset of another index  $S$ , can be derived from  $S$  by performing a series of individual point deletions and reordering points in leaves. By design of the delete operation, no new nodes are being constructed, but only underflowing nodes are removed. Therefore,  $S'$  cannot contain a node at some position where there is no node in  $S$ .

**Corollary 1.** *(From Lemma 3) Let  $A$  be data set and  $B \subseteq A$ . Then, the maximum node depth in any SkyMap indexing  $B$  is smaller than or equal to the maximum node depth found in any SkyMap indexing  $A$ .*

Consequently, the maximum depth of a SkyMap indexing only the skyline of  $A$  is at most as large as the maximum depth of any SkyMap indexing the whole data set  $A$ .

Finally, let us consider the case of uniformly distributed data, which usually is a good indicator of the general behavior of skyline data structures.

**Lemma 4.** *Let  $A$  be a random set of size  $n$ , where each  $p \in A$  is a random vector, which has independently and uniformly been drawn from the unit hypercube  $[0, 1]^d$ . Then, the maximum depth of any SkyMap indexing  $A$  is  $O(\log n)$  in expectation.*

*Proof.* (Sketch) Since each point  $p$  has been drawn uniformly from  $[0, 1]^d$ , the bits of  $p$ 's Z-address are independent random variates with equal probabilities for each of the outcomes 0 and 1. Therefore, all points spread uniformly over all possible Z-address prefixes, thus implying logarithmic depth.

**Corollary 2.** *(From Lemma 4 and Corollary 1) The SkyMap index storing the skyline of a random uniformly distributed data set  $A$  of size  $n$  has maximum node depth  $O(\log n)$  in expectation.*

We conclude that SkyMap indexes provide the balancedness needed for quick data access combined with robustness against degeneration caused by frequent updates.

## 4.5 Skyline Algorithms

To enable the actual processing of skyline queries, we still need an effective way to perform dominance tests on the index. Given a data point  $p$ , our ISDOMINATED operation checks whether  $p$  is dominated by some point stored in the SkyMap. The algorithm (cf.

```

function ISDOMINATED( $p$ )
  if  $root = null$  then return false
  else
     $z \leftarrow p$ 's Z-address
    return ISDOMINATED( $p, z, root, 0, \{1, \dots, d\}$ )

function ISDOMINATED( $p, z, node, depth, EQ$ )
  if  $node$  is an internal node then
    ( $zb_0, \dots, zb_{b-1}$ )  $\leftarrow$  the first  $b$  bits of  $z$ 
     $z' \leftarrow z$  without its first  $b$  bits
    for each  $i = 2^b - 1, \dots, 0$  with  $node.children[i] \neq null$  do
      ( $ib_0, \dots, ib_{b-1}$ )  $\leftarrow i$  in  $b$ -bit representation
       $EQ' \leftarrow EQ$ 
      for each  $j = depth, \dots, depth + b - 1$  do
         $dim \leftarrow (j \bmod d) + 1$ 
        if  $dim \in EQ$  then
          if  $zb_j < ib_j$  then  $EQ' \leftarrow EQ' \setminus \{dim\}$ 
          else if  $zb_j > ib_j$  then next  $i$ 
           $\triangleright p \prec_{dim} node$ 
           $\triangleright p \succ_{dim} node$ , continue the outer for loop
        if  $EQ' = \emptyset$  then return true
        if ISDOM. $(p, z', node.children[i], depth + b, EQ')$  then return true
      return false
    else
      for each point  $q$  contained in  $node$  do
        if  $p \prec q$  then return true
      return false

function SKYLINE( $A$ )
  Sort  $A$  by decreasing Z-addresses; let  $P$  be the resulting list of points
   $S \leftarrow$  a new empty SkyMap index
  for  $i = 0, \dots, |A| - 1$  do
     $p \leftarrow P[i]$ 
    if no point in  $S$  dominates  $p$  then Insert  $p$  into  $S$ 
  return the set of points contained in  $S$ 

```

**Fig. 6.** Checking whether a point is dominated and computing the skyline of a given data set

Fig. 6 rests on the following observation: When traversing a SkyMap index while looking for points dominating  $p$ , one can skip any node (along with all its children) whose corresponding Z-region is worse than  $p$  with respect to at least one dimension.

ISDOMINATED works as follows: The SkyMap index is traversed in depth-first search, beginning with those nodes belonging to the largest Z-addresses to visit nodes and points with high domination power first. At each internal node, we scan over all child nodes and compare the length- $b$  Z-address interval used to identify each child to the corresponding interval in  $p$ 's Z-address. This way, we can easily determine whether  $p$  is better in some dimension  $dim$  than the child node. If this is the case, the child can be immediately be excluded from further traversal. While traversing the index, we continuously maintain a set  $EQ$  of dimensions, in which the current node  $node$  and  $p$  have found to be equal with respect to the Z-address prefix processed so far. A dimension is eliminated from  $EQ$  if a child node is known to be better than  $p$  with respect to this dimension. During the traversal only those dimensions still contained in  $EQ$  have to be checked. Whenever we reach a leaf node,  $p$  is compared to each of the node's points.

The leaf node capacity  $c$  typically will be adapted to the current hardware so that all of the leaf node's points can fit into the CPU cache, which supports extremely fast point comparisons. Moreover, the set  $EQ$  can be implemented only by performing bitwise operations on ordinary integers. The same is true for all Z-address comparisons. This way, expensive point comparisons again can largely be avoided.



<pre> <b>function</b> MINSERT(<math>p</math>)   <b>if</b> some point in the skyline index dominates <math>p</math> <b>then</b>     Insert <math>p</math> into the data index   <b>else</b>     <math>P \leftarrow</math> the set of skyline points being dominated by <math>p</math>     <b>for each</b> <math>q \in P</math> <b>do</b>       Delete <math>q</math> from the skyline index       Insert <math>q</math> into the data index     Insert <math>p</math> into the skyline index </pre>	<pre> <b>function</b> MDELETE(<math>p</math>)   <b>if</b> <math>p</math> is not contained in the skyline index <b>then</b>     Delete <math>p</math> from the data index   <b>else</b>     <math>P \leftarrow</math> the set of all data points being dominated by <math>p</math>     Delete <math>p</math> from the skyline index     <b>for each</b> <math>q \in P</math> <b>do</b>       <b>if</b> no point in the skyline index dominates <math>q</math> <b>then</b>         Delete <math>q</math> from the data index         Insert <math>q</math> into the skyline index </pre>
--	--

**Fig. 7.** Maintaining the skyline in case of changing data

Now, we are able to state our SKYLINE algorithm, which combines the insights of traditional list-based skyline algorithm with the pruning power of SkyMap. It also exploits the following important monotonicity property of Z-addresses:

**Lemma 5.** *Let  $p, q \in [0, 1]^d$ . If  $Z(p) > Z(q)$ , then  $p \not\prec q$ .*<sup>1</sup>

Our SKYLINE algorithm works as follows: It first presorts the data set by decreasing Z-addresses, which by the above lemma guarantees that no data point dominates any of its predecessors. Then, an empty SkyMap index is created and the sorted list is scanned linearly. For each point  $p$ , the function ISDOMINATED( $p$ ) is called. If it returns true, then there already is a point in the index dominating  $p$ , thus eliminating  $p$  as a skyline point. If the function returns false,  $p$  must be a skyline point. In this case, we insert  $p$  into the SkyMap index, which always contains the skyline of the data points processed so far. Fig. 6 shows the pseudocode.

For continuously maintaining the skyline of a large database, we propose to use two SkyMap indexes, where the first indexes all current skyline points and the second all remaining database items. To support bulkloading, insertion and deletion, we designed three algorithms: MBULKLOAD, MINSERT, and MDELETE.

Given a data set  $A$ , MBULKLOAD creates a valid initial configuration of the two indexes by first computing the skyline of  $A$  by means of our SKYLINE algorithm, then removes all skyline points from  $A$ , and finally bulkloads  $A$  into the second SkyMap index using the BULKLOAD method.

MINSERT and MDELETE require the helper function FINDDOMINATED, which returns a list of all items in a SkyMap index that are dominated by a given point  $p$ . The returned list is sorted by decreasing Z-addresses. FINDDOMINATED follows the same approach as our ISDOMINATED function, with only two major differences: First, instead of finishing the traversal at the first dominated point, each the search continues until the whole tree has been traversed; second, all bitwise comparisons operations are performed inverted. As FINDDOMINATED can easily be derived from ISDOMINATED's pseudocode, we abstain from providing an extensive description.

With the help of FINDDOMINATED, maintained insertions and deletions can be performed as shown in Fig. 7. When inserting a new point  $p$ , no special action is required if  $p$  is dominated, whereas in case  $p$  is a new skyline point all current skyline item being dominated by  $p$  need to be found and moved to the data index. The opposite happens when deleting an existing point  $p$ . Here, no special action is required if  $p$  is dominated. Otherwise, all index points being dominated by  $p$  need to be found and moved

<sup>1</sup> A proof of this lemma can be found in [9].

to the skyline index if they now become new skyline points. As the result list returned by FINDDOMINATED is ordered by decreasing Z-addresses, we can exploit the same monotonicity property that already proved to be helpful in our SKYLINE method.

We designed our algorithms for in-memory computations. It has been demonstrated in previous work that skylining inherently is CPU-bound and easily become intractable if main memory is scarce (see e.g. [2] or [16]). Fortunately, even the largest data sets so far used in skyline research easily fit into a modern desktop computer's main memory.

## 5 Evaluation

Beside our own method, we implemented the following state-of-the-art algorithms for skyline computation and maintenance: (1) OSPSPF [16] with the pivoting extension for bulkloading introduced in [7], (2) the ZB-tree [9]. Our experimental setup follows the standard methodology used in skyline research.

Regarding our test data sets, we decided to follow the common methodology in the skylining literature and thus used the three data generators IND (independent data dimensions), CORR (correlated), and ANTI (anti-correlated) as proposed in [2]. Given parameters  $d$  and  $n$ , these algorithms randomly create data sets having independent, correlated, or anti-correlated dimensions, respectively. In the experiments reported below, *we did not apply any data normalization by means of copulas*, as as found SkyMap's performance on copula-normalized data to be very similar. This indicates that our approach is quite robust with respect to data skewness.

We also wanted to evaluate our method on real-world data sets, but soon realized that all real-world data sets traditionally used in skyline research (e.g., NBA, Corel, Household) are far too small and simplistic to pose a challenge to modern skyline algorithms and thus allow meaningful comparisons among them, a problem that already became apparent in [16], [7], and [9]. To remedy this issue we decided to use a 60-dimensional data set consisting of texture features extracted from  $n = 275,465$  aerial images, where the skyline consists of 26,817 points. The data set can be downloaded from the web site of the Vision Research Lab at UCSB<sup>2</sup>

For each of the different choices of  $d$  and  $n$  we used in our experiments, we randomly generated 10 data sets with each of the three data generators. All running times reported below are averages over the corresponding 10 skyline computations with each algorithm. We did not report any running times below 100 ms due to measurement uncertainty; numbers below this threshold tend to reflect arbitrary delays in memory allocation and data initialization rather than actual performance and scalability.

Our programming language is Java 6. We carefully profiled and optimized all our code to eliminate weak spots. This also included checking all our implementations against existing original code. For example, we compared our version of the partitioning algorithm to the code used in [16], which the authors kindly made available to us.

All experiments have been conducted on a Linux server system equipped with two Intel Core i7 920 2.67 GHz quad-core processors and 20 GB of main memory. However, all our code is single-threaded and uses only a small fraction of the available memory.

<sup>2</sup> <http://vision.ece.ucsb.edu/download.html>

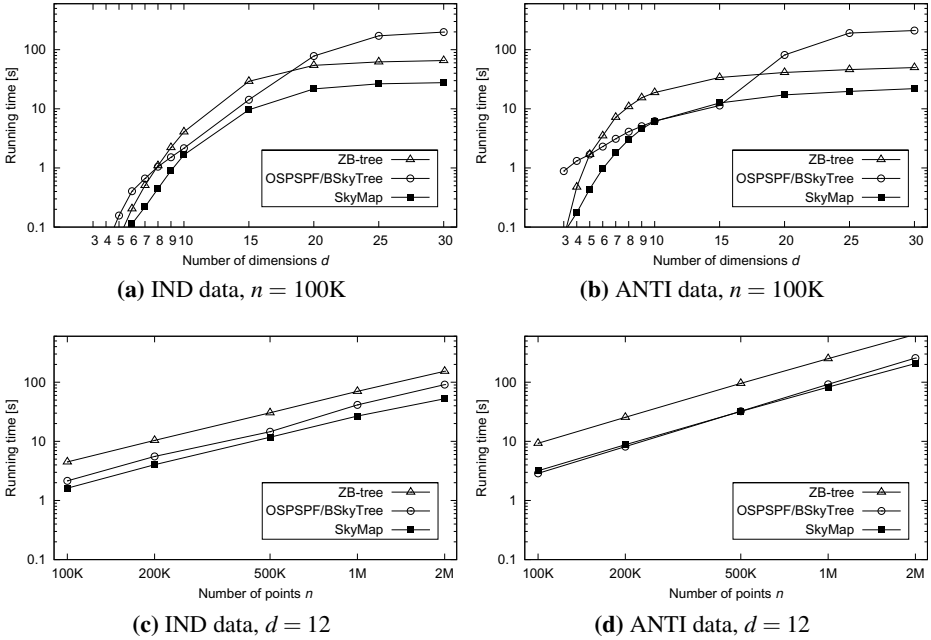


Fig. 8. Skyline queries on synthetic data

Since both the ZB-tree and SkyMap depend on configuration parameters, we tried a variety of different settings and finally ended up with a minimum node size of 20 and a maximum node size of 50 for the ZB-tree; we chose  $b = 2$  and  $c = 10$  for SkyMap.

### 5.1 Skyline Computation

We first evaluated our approach in the traditional setting of skyline queries, that is, given a data set  $A$ , the task is to compute its skyline. To investigate the influence of data dimensionality, we set  $n = 100K$  and varied  $d$  over a large range of values. We also evaluated the scalability of our method with respect to  $n$  by fixing  $d = 12$  and varying  $n$ . Fig. 8 depicts our findings. Due to space limitations, we only report performance numbers for IND and ANTI data; results for CORR data are very similar.

As we can see there is no clear winner in the comparison between the ZB-tree and OSPSPF/BSkyTree. The ZB-tree provides better performance for larger  $d$  but is worse than OSPSPF/BSkyTree for mid-range values of  $d$ . Scalability with respect to  $n$  is not an issue for any of the three algorithms. However, in any case, the SkyMap approach significantly outperforms its competitors. Even in the special case  $d = 12$ , where OSPSPF/BSkyTree’s performance comes closest, SkyMap can easily defend its advantage even for different values of  $n$ . We also measured scalability in  $n$  with respect to many other values of  $d$ , but in no case any of ZB-tree or OSPSPF/BSkyTree has been able to perform better than SkyMap. The results on our real-world data set confirm our findings: Computing the skyline takes 13.7 seconds for the ZB-tree, 10.9 seconds for OSPSPF/BSkyTree, and 8.6 seconds for SkyMap.

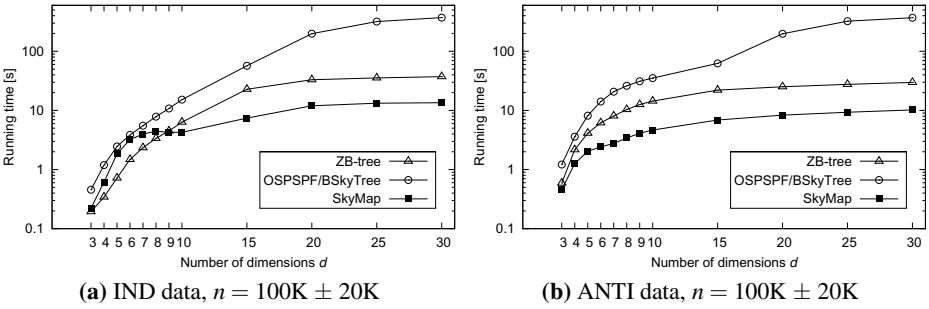


Fig. 9. Skyline maintenance on synthetic data

### 5.2 Skyline Maintenance

To investigate the performance of our method in the setting of skyline maintenance we used to following three-step task: Given  $d$  and  $n$ , first  $n$  data points are bulkloaded into the database and the skyline is computed. Then, 20% of the data are randomly deleted, where after each deletion the skyline has to be maintained. Finally, a new data set of size 20% is created randomly (according to the original data distribution) and inserted into the database, where again after each single insertion the skyline has to be maintained. Our results for scalability with respect to  $d$  as depicted in Fig. 9. We can see that SkyMap consistently performs better than OSPSPF/BSkyTree, sometimes even by two orders of magnitude. Compared to the ZB-tree, the results are twofold. In case of very small skylines, the ZB-tree performs slightly better than SkyMap; however, when it comes to scalability in  $d$  and skyline size, SkyMap clearly outperforms the ZB-tree. We also measured scalability in  $n$  but always received the same scaling behavior as already depicted in Fig. 8. Therefore, we did not include any further graphics illustrating the fact that no method has scalability issues with respect to  $n$ .

### 5.3 Influence of Parameters

Finally, we investigated the influence of different parameters  $b$  and  $c$  on the performance of SkyMap indexes. To illustrate all relevant effects, we chose a skyline query example on IND data with  $d = 20$ . Our results are depicted in Fig. 10. We can see that regardless of the choice of the leaf capacity  $c$ , the retrieval performance quickly degrades for

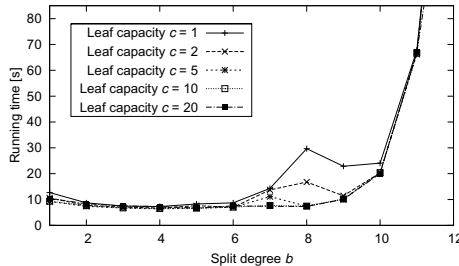


Fig. 10. Influence of parameters  $b$  and  $c$

high split degrees  $b$ , which also explains the bad performance of OSPSPF/BSkyTree in high-dimensional space we observed previously (recall that quadtrees always split the surrounding space into  $2^d$  partitions). Moreover, our results show performance disadvantages for very small leaf capacities, in particular when  $b$  is large. Due to our flexible design, we can always choose a combination of parameters that optimally exploits the specifics of the current hardware. However, our SkyMap approach is robust enough to work well over a wide range of parameters.

## 6 Conclusion

In this paper, we have shown that the current state of the art in skylining processing is characterized by a tradeoff. One can either have high-performance indexing on static data (OSPSPF/BSkyTree) or high-performance skyline maintenance (ZB-tree), but unfortunately not both. However, easy integration of skyline algorithms into existing database systems calls for a single method that efficiently supports both scenarios. With our SkyMap index we have proposed a solution that successfully resolves this issue, and consistently outperforms previous algorithms on most data sets.

## References

1. Bartolini, I., Ciaccia, P., Patella, M.: Efficient sort-based skyline evaluation. *ACM Transactions on Database Systems* 33(4), 31 (2008)
2. Börzsönyi, S., Kossmann, D., Stocker, K.: The Skyline operator. In: *Proceedings of the 17th International Conference on Data Engineering (ICDE 2001)*, pp. 421–430 (2001)
3. Chan, C.Y., Jagadish, H.V., Tan, K.L., Tung, A.K.H., Zhang, Z.: Finding  $k$ -dominant skylines in high-dimensional space. In: *Proceedings of the 32th ACM SIGMOD International Conference on Management of Data (SIGMOD 2006)*, pp. 503–514 (2006)
4. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: *Proceedings of the 19th International Conference on Data Engineering (ICDE 2003)*, pp. 717–719 (2003)
5. Eng, P.K., Ooi, B.C., Tan, K.L.: Indexing for progressive skyline computation. *Data and Knowledge Engineering* 46(2), 169–201 (2003)
6. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: *VLDB 2002*, pp. 275–286 (2002)
7. Lee, J., Hwang, S.: B-SkyTree: Scalable skyline computation using a balanced pivot selection. In: *Proceedings of the 13th International Conference on Extending Database Technology (EDBT 2010)*, pp. 195–206 (2010)
8. Lee, J., Hwang, S., Nie, Z., Wen, J.R.: Navigation system for product search. In: *Proceedings of the 26th International Conference on Data Engineering (ICDE 2010)*, pp. 1113–1116 (2010)
9. Lee, K.C.K., Lee, W.C., Zheng, B., Li, H., Tian, Y.: Z-SKY: An efficient skyline query processing framework based on Z-order. *The VLDB Journal* 19(3), 333–362 (2010)
10. Nelsen, R.B.: *An Introduction to Copulas*, 2nd edn. Springer, Heidelberg (2006)
11. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. *ACM Transactions on Database Systems* 30(1), 41–82 (2005)
12. Sagan, H.: *Space-Filling Curves*. Springer, Heidelberg (1994)
13. Sahni, S.: Tries. In: Mehta, D.P., Sahni, S. (eds.) *Handbook of Data Structures and Applications*, pp. 28-1–28-20. Chapman and Hall, Boca Raton (2005)

14. Tao, Y., Xiao, X., Pei, J.: Efficient skyline and top-k retrieval in subspaces. *IEEE Transactions on Knowledge and Data Engineering* 19(8), 1072–1088 (2007)
15. Viappiani, P., Faltings, B., Pu, P.: Preference-based search using example-critiquing with suggestions. *Journal of Artificial Intelligence Research* 27, 465–503 (2006)
16. Zhang, S., Mamoulis, N., Cheung, D.W.: Scalable skyline computation using object-based space partitioning. In: *Proceedings of the 35th ACM SIGMOD International Conference on Management of Data (SIGMOD 2009)*, pp. 483–494 (2009)

# A Path-Oriented RDF Index for Keyword Search Query Processing

Paolo Cappellari<sup>1</sup>, Roberto De Virgilio<sup>2</sup>, Antonio Maccioni<sup>2</sup>, and Mark Roantree<sup>1</sup>

<sup>1</sup> School of Computing

Dublin City University, Dublin, Ireland

{pcappellari, Mark.Roantree}@computing.dcu.ie

<sup>2</sup> Dipartimento di Informatica e Automazione

Università Roma Tre, Rome, Italy

{dvr, maccioni}@dia.uniroma3.it

**Abstract.** Most of the recent approaches to keyword search employ graph structured representation of data. Answers to queries are generally sub-structures of the graph, containing one or more keywords. While finding the nodes matching keywords is relatively easy, determining the connections between such nodes is a complex problem requiring on-the-fly time consuming graph exploration. Current techniques suffer from poorly performing worst case scenario or from indexing schemes that provide little support to the discovery of connections between nodes.

In this paper, we present an indexing scheme for RDF that exposes the structural characteristics of the graph, its paths and the information on the reachability of nodes. This knowledge is exploited to expedite the retrieval of the sub-structures representing the query results. In addition, the index is organized to facilitate maintenance operations as the dataset evolves. Experimental results demonstrates the feasibility of our index that significantly improves the query execution performance.

## 1 Introduction

In 2006, the Linked Open Data initiative (<http://linkeddata.org/>) inspired practitioners, organizations and universities to either publish or build from scratch RDF (Resource Description Framework, a graph-oriented logical data model) datasets from data that had previously been stored using traditional models [1], contributing to the definition of what is today called the *Web of Data*. The main objectives in searching this web of data are: the location and retrieval of results that are most relevant to the user's search, and to give more relevance to valid results currently missed (or low ranked) by modern search engines. The methodology for achieving these aims is to include as much semantics as possible in datasets and in general, RDF has been used to provide indexes with rich semantics to better interpret user queries.

In a scenario where the online data is constantly increasing, the difficulty for users is locating and retrieving the data that accurately meet their requirements. Having to know how data is organized and the query language to access data represent an obstacle to information access to non expert users. For this reason, keywords search systems are increasingly popular. Many approaches (e.g. [4,8,10,12,14,15]) implement information

retrieval (IR) strategies on top of traditional database systems, with the goal of eliminating the need for users to understand query languages or be aware the data organization. A general approach involves the construction of a graph-based representation where query processing addresses the problems of an exhaustive search over the RDF graph. Two main steps are involved in the approach. Firstly, the system retrieves those parts of the graph that match users keywords with a subsequent identification of any connections across graph segments returned by the query process. Secondly, the system ranks the *combined* graph segments and top-k results are presented from the candidate result set. Clearly, the graph search is a crucial step. Typically it is supported by indexing systems (computed off-line) to guarantee the efficiency of the search. Existing systems [9,13,18] focus mainly on indexing nodes information (e.g. labels, position into the graph, cardinality and so on) to achieve scalability and to optimize space consumption. While locating nodes matching the keywords is relatively efficient using these indexes, determining the connections between graph segments is a complex and time-consuming task that must be solved on-the-fly at query processing time. To provide results in a reasonable time, current approaches do not perform an exhaustive search. As index schemes do not adequately support the discovery of connections between nodes, heuristics must be introduced to assist the search in locating a more complete result set. Thus, while the introduction of RDF-based solutions offers simple user interfaces with genuine semantic search features, the problem now lies with how to associate or connect intermediate result sets and how to manage the cost of determining those associations.

**Contribution.** We present a novel indexing scheme for RDF datasets that captures associations across RDF paths *before* query processing and thus, provides both an exhaustive semantic search and superior performance times. Unlike other approaches involving implementation based solutions, we follow a process that starts with the definition of the index at a conceptual level. It comprises paths and information on the reachability of nodes within the RDF graph. The next step is a logical translation for a relational database. Eventually, at the physical level, we choose optimization techniques based on physical indexing and partitioning. Because of the storage model foundations, the system can easily represent structural aspects of an RDF graph. Moreover we provide a set of procedures to insert or delete nodes (resources) and edges (properties) into the index and thus, support updates to the RDF graph. Such index is deployed in YAANII [5], a novel keyword search framework that leverages a joint use of scoring functions and solution building algorithms to get the best results for the initial result set. Experiments demonstrate how the proposed indexing scheme allows to significantly improve the efficiency of the overall process. The paper is organized as follows: Section 2 discusses related work; Section 3 introduces the basic concepts and illustrates how we model our path-oriented index; based on this model, Section 4 describes how the index is built and maintained in an efficient manner; finally Section 5 provides experimental evaluations and in Section 6, conclusions and future work are presented.

## 2 Related Work

Several proposals [9,13,18] implement in-memory structures that focus on node indexing. Others [16,19] focus on indices for SPARQL query execution and join efficiency, not offering concrete support to graph exploration for keyword search. In [9], authors



provide a *Bi-Level Indexing Keyword Search* (BLINKS) method for the data graph. This approach assumes keywords can only occur in nodes, not in edges, and is based on pre-computed distances between nodes. The system implements two indices: one index stores information on which nodes are reachable from a given node; the other index is a hash table storing the shortest distance between pairs of nodes. In [13], authors propose a linked-list indexing scheme for RDF. The index is composed of a dictionary table, a statement table and a resource table. The dictionary table maintains the association between each resource and its (generated) identifier. Primarily, this table acts as reverse look-up identifier to resource label, to complete the answer to a query. The statement table maintains the list of the  $\langle s, p, o \rangle$  RDF statements, where each statement has three references, each pointing to the next statement using the same  $s, p, o$  respectively. The resource table contains information about all the resources ( $s$  and  $o$ ), linking each one to the first statement in which it occurs, and collecting statistics about statements presenting such resources. In [18], authors propose an approach to keyword search in RDF graph through query computation, implementing a system called SEARCHWEBDB. It is supported by two index structures: a keyword-to-element index and a graph index. The former implements an inverted index to associate each possible keyword to nodes or edges in the graph. To capture semantically similar words such as synonyms, every term is expanded with its similar term as described in WordNet [6]. The latter stores schema information of the graph, that is classes and relations between classes. The authors refer to this type of schema as a *summary graph*. Contrary to those approaches that index the entire graph, SEARCHWEBDB derives the query structure by enriching the summary graph with the input keywords. The search and retrieval process for the *enriched* summary graph, with all its possible distinct paths beginning from some keyword elements, provides a set of queries that once calculated, provides the final sub-graph answers. Other proposals focus on indexing graph substructures (e.g. paths, frequent subgraphs, trees). Typically, these indexes are exploited in approaches dealing with graph matching problems, often to filter out graphs that do not match an input query. Approaches in this area can be classified in: graph indexing and subgraph indexing. In graph indexing approaches, e.g. gIndex [20], TreePi [21], FG-Index [3], the graph database consists of a set of small graphs. The indexing aims at finding all database graphs that contain or are contained by a given query graph. On the other hand, subgraph indexing approaches, e.g. GraphGrep [7], TALE [17], GADDI [22], aims at indexing large database graph, with the goal of finding all (or a subset of) the subgraphs that match a given query efficiently. Our indexing scheme provides an agile solution with respect to graph substructures indexing approaches and enriches traditional node/edges indexing proposals with exhaustive information about connections between nodes.

### 3 Index Modeling

Our goal is to model a generic indexing scheme to support queries execution and keyword based search engines. Generally, standard queries are composed of a set of, possibly, interrelated path expressions. The result to a standard query is the portion of the graph that matches the path-expression(s) provided in the query specification. In the keyword based search paradigm, users are assumed to be agnostic of the schema and

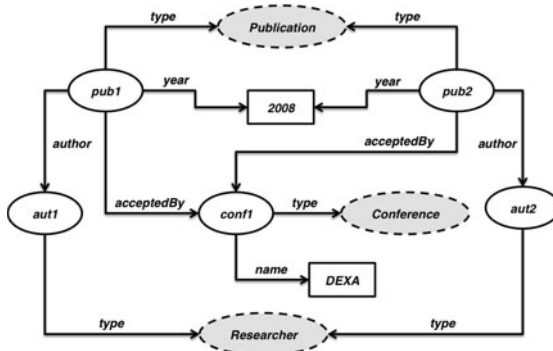


Fig. 1. An example of reference

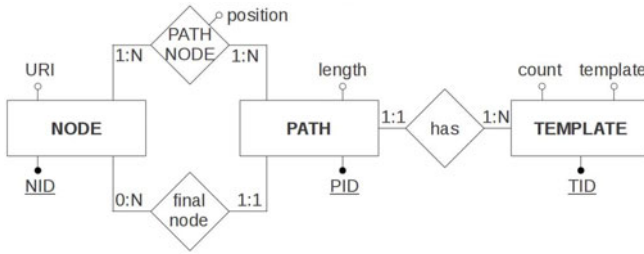
the query is a list of keyword terms. In keyword search systems the focus is on discovering connections between nodes matching keywords, on top of which query answers are built (as subgraphs). We would support systems that perform standard queries, such as path expressions, as well as keyword search, where the emphasis is on discovering connections between those parts of the graph holding information relevant to the input keywords. In particular, we focus on semantic dataset expressed in RDF format, that is a model that describes a directed labeled graph, where nodes are resources (identified by URIs) and edges have a label that expresses the type of connection between nodes.

**Definition 1.** A *labeled directed graph*  $G$  is a three element tuple  $G = \{V, L, E\}$  where  $V$  is a set of nodes,  $L$  is a set of labels and  $E$  is a set of edges of the form  $e(v, u)$  where  $v, u \in V$  and  $e \in L$ .

In  $G$  we call *sources* the nodes  $v_{src}$  with no incoming edges (i.e.  $\nexists e(u, v_{src}) \in E$ ), and *sinks* the nodes  $v_{sink}$  with no outgoing edges (i.e.  $\nexists e(v_{sink}, u) \in E$ ). We call *intermediate node*, a node that is neither a source nor a sink. Consider the example in Fig. 1. It illustrates an ontology about *Publications* written by *Researchers* (i.e. authors) accepted and published into a *Conference*. We have two sources, *pub1* and *pub2*, and four sinks, of which for instance we have *Publication* and *Conference*.

From the data graph point of view, in a RDF graph we have classes and data values (i.e. sinks), URIs (i.e. intermediate nodes and sources) and edges. Since it can be assumed the user will enter keywords corresponding to attribute values such as a name rather than using a verbose URI (e.g. see [18]), keywords might refer principally to edges and sinks of the graph. Therefore in our framework we are interested to index each path starting from a source and ending into a sink. Moreover, any node can be reached by at least one path originating from one of the sources. Paths originating from sources and reaching sinks includes (sub-)paths that stop in intermediary nodes. In case a source node is not present, a fictitious one can be added. To this aim, the so-called *Full-Path* is a path originating in a source and ending in a sink.

**Definition 2 (Full-Path).** Given a graph  $G = \{V, L, E\}$ , a *full-path* is a sequence  $pt = v_1 - e_1 - v_2 - e_2 - \dots - e_{n-1} - v_f$  where  $v_i \in V$ ,  $e_i \in L$  (i.e.  $e_i(v_i, v_{i+1}) \in E$ ),  $v_1$  is a source and the final node  $v_f$  is a sink. We refer to  $v_i$  and  $e_i$  as tokens in  $pt$ .



**Fig. 2.** Conceptual modeling of the index

In Fig. 1 a full-path  $pt_k$  is `pub1-author-aut1-type-Researcher`. The *length* of a path corresponds to the number of its nodes; the *position* of a node corresponds to its position in the presentation order of all nodes. In the example,  $pt_k$  has length 3 and the node `aut1` is in position 2. In the rest of the paper we refer to paths as full-paths.

The sequence of edge labels (i.e.  $e_i$ ) describes the structure of a path. In some sense, the sequence of  $e_i$  is a *schema* for the information instantiated in the nodes. We can say that paths sharing the same structure carry homogeneous information. More properly, we say that the sequence of  $e_i$  in a path represents its *template*. Given a path  $pt$  its template  $t_{pt}$  is the path itself where each node  $v_i$  in  $pt$  is replaced with the wild card `#`. In our example  $pt_k$  has the following template: `#-author-#-type-#`. Several paths can share the same structure: it allows us to cluster paths according to the template they share. For instance the path  $pt_j$  `pub2-author-aut2-type-Researcher` has the same template of  $pt_k$ , that is  $pt_j$  and  $pt_k$  are in the same cluster.

In our framework we follow a three levels modeling. Starting from conceptual level, Fig. 2 shows an ER-diagram modeling the major constructs in our index. We start from the entity `NODES` modeling a node in the graph. The label is characterized by the attribute *URI* and each node is identified by *NID*. We then have the main entity `PATHS` representing a full-path. In particular, each path is identified by *PID*, presents the *length*, and the relation *final node* with `NODES`, representing the sink of the path. Each node *belongs* to a path with a *position* (i.e. the relation *pathnode*). Finally we have the entity `TEMPLATES` identified by *TID*, presenting the sequence of edge labels (i.e. the attribute *template*) and the number of paths sharing the template (i.e. the attribute *count*). Between `PATHS` and `TEMPLATES` a one-to-many relation assigns a template to each path.

At logical level, we have a straightforward transformation to a relational model. Fig. 3 shows the logical modeling of Fig. 2, populated with data from the example in Fig. 1. Each entity is transformed into a relation with the corresponding primary key and with foreign keys for the one-to-many relationship it contains. The many-to-many relationship `PATHNODES` is also transformed to a relation, correlating paths with their nodes, and vice versa, through the attribute *position*.

As RDF datasets are often very large, at physical level we exploit relational DBMS features to tune our schema for better performance. In particular we employ Oracle as relational DBMS. First of all we implement horizontal partitioning on the tables, based on the value of a column (called *range*). In particular `PATHS` is partitioned with respect to the template (i.e. *TID*). In this way each partition is a cluster of homogeneous full-paths.

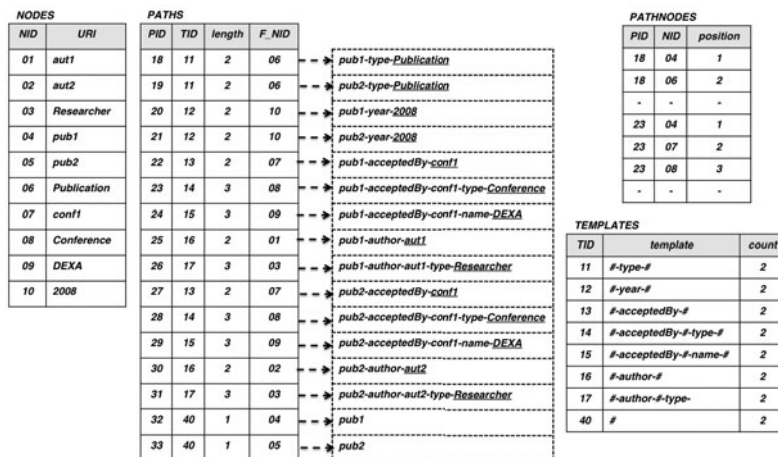


Fig. 3. Logical Modeling of the index

Then we define physical indexes on the single partitions and on the other unpartitioned tables. Specifically, we employ the Oracle Index-organized tables (IOTs), that are a special style of table structure stored in a B-tree index frame. Along with primary key values, in the IOTs we also store the non-key column values. IOTs provide faster access to table rows by the primary key or any key that is a valid prefix of the primary key. Because the non-key columns of a row are present in the B-TREE leaf block itself, there is no additional block access for index blocks. In our case PATHS and PATHNODES are organized as IOTs. The matching is supported by standard IR engines (c.f. Lucene Domain index (LDi) [1]) embedded into Oracle as a type of index. In particular we define a LDi index on the attributes *label* and *template* of tables NODES and TEMPLATES respectively. Further, semantically similar entries such as synonyms, hyponyms and hypernyms are extracted from WordNet [6], supported by LDi.

With this broad goal in mind, our indexing system provides two major advantages: (i) intersections between paths, needed to build the subgraph solutions, are efficiently identifiable (as we will show in the next section), and (ii) the template based classification splits the information from the graph into non-overlapping subsets, as each cluster represents an instance of a different template (or schema).

## 4 Index Management

In this section, we describe the index creation, discuss the processes to maintain the the index when the graph changes and, finally, illustrate how to query the index.

### 4.1 Constructing the Graph Index

Given a graph, the creation of the index requires three steps: (i) node hashing, (ii) source identification, and (iii) computation of full-paths, from source to sink.

<sup>1</sup> <http://www.scribd.com/doc/38406372/Lucene-Domain-Index>

**Algorithm 1.** BFS-based graph exploration algorithm

---

**Input** : Sets *triples* and *Roots*  
**Output**: PATHS, PATHNODES and TEMPLATES populated

```

1 foreach  $r \in \text{Roots}$  do
2    $Queue \leftarrow \emptyset$ ;
3    $pt_r \leftarrow \text{NewID}()$ ;
4    $\text{PATHS} \leftarrow \text{PATHS} \cup \langle pt_r, \text{NewID}(), 1, r \rangle$ ;
5    $\text{PATHNODES} \leftarrow \text{PATHNODES} \cup \langle pt_r, r, 1 \rangle$ ;
6    $\text{Enqueue}(Queue, pt_r)$ ;
7   while Queue is not empty do
8      $\text{Dequeue}(Queue, pt)$ ;
9     if  $\exists \langle pt, \tau, l, n \rangle \in \text{PATHS}$  then
10      foreach  $\langle n, p, o \rangle \in \text{triples}$  do
11        if  $\nexists \langle pt', \tau, l, o \rangle \in \text{PATHS}$  then
12           $pt' \leftarrow \text{NewID}()$ ;
13           $\tau' \leftarrow \text{NewTemplate}(\tau, p)$ ;
14           $\text{PATHS} \leftarrow \text{PATHS} \cup \langle pt', \tau, l + 1, o \rangle$ ;
15          foreach  $\langle m, pt, pos \rangle \in \text{PATHNODES}$  do
16             $\text{PATHNODES} \leftarrow \text{PATHNODES} \cup \langle m, pt', pos \rangle$ ;
17           $\text{PATHNODES} \leftarrow \text{PATHNODES} \cup \langle o, pt', l + 1 \rangle$ ;
18           $\text{Enqueue}(Queue, pt')$ ;

```

---

The input dataset is assumed to be an RDF graph where the information is modeled as a set of triples. Each triple has the form  $\langle s, p, o \rangle$ , where:  $s$  is the *origin resource* (known as *subject*),  $o$  is the *target resource* (*object*),  $p$  is a property linking the origin with the target resource (known as *property*). In the first step, we populate columns URI and NID of table NODES with nodes URIs and IDs, respectively. In the second step, we identify the source nodes, that is all subjects of triples that never occur as objects. Finally in a third step, we explore the graph to build the full-paths: once identified, the path is decomposed to populate tables TEMPLATES, PATHS and PATHNODES. We explore the graph using a concurrent version of the *Breadth-First Search* (BFS) algorithm, shown in Algorithm 1. Starting from each source (line 1), the algorithm creates a one node length path  $pt_r$ , composed of the source node only, associates the path with an ID using the function *newID*, and puts  $pt_r$  in the *Queue* (lines 3-6). While there are elements in the queue (line 7), the algorithm dequeues  $pt$  from *Queue* (line 8):  $pt$  represents the path just indexed and ending into the node  $n$ . Starting from  $n$  the algorithm collects all nodes  $o$ , object into triples  $\langle n, p, o \rangle$  (line 10). If a path ending in  $o$  does not exist (line 11), then new paths  $pt'$  extending  $pt$  with the edge  $\langle n, p, o \rangle$  (lines 13-17) are created. Finally, it enqueues  $pt'$ , that will possibly create new paths of length  $l + 2$ .

## 4.2 Index Maintenance

When the dataset changes, the index must be updated as a consequence. We developed a number of basic maintenance operations: insertion/deletion of a node; insertion/deletion

of an edge; and update of the content of a node/edge label. More sophisticated operations, such as the insertion or removal of a triple or a set of triples, can be implemented as a composition of these basic functions.

Before presenting the algorithms for these basic operations, it is useful to introduce a new concept and a new routine that simplify the discussion of the maintenance operations: the *tailpath* and the *forward-concatenation*. A tailpath is a sub-path of an full-path. Specifically, it is the sub-path starting at an intermediate position and ending with the final node of the full-path (i.e. a sink).

**Definition 3 (TailPath).** A *tailpath* is represented as a pair  $\langle \text{nodes}, \text{sub-template} \rangle$  where *nodes* is a list of pairs  $\langle n, \text{pos} \rangle$ , *n* is a node belonging to the tailpath and *pos* its position. As an extension, *sub-template* is the corresponding template of tailpath.

Tailpaths are not stored in the index, but computed dynamically if and when needed for maintenance operations.

**Definition 4 (Forward-Concatenation).** The *forward-concatenation* function creates new paths by merging a selected path with a tailpath.

The forward-concatenation function uses a new edge, connecting the last node of the selected path, with the first node of the tailpath to form a new Full-Path.

Algorithm 2 illustrates how to compute a tailpath starting from a node at a specific position in a given path *pt*. The tailpath is built by querying the information from tables

---

#### Algorithm 2. TailPath.

---

**Input** : A Path *pt*, an integer *pos*

**Output**: A Tail Path *tailPath*

```

1 set  $\leftarrow \emptyset$ ;
2 foreach  $\langle pt, n_r, pos' \rangle \in \text{PATHNODES} : pos' \geq pos$  do
3    $\setminus$  set  $\leftarrow set \cup \langle n_r, pos' - pos + 1 \rangle$ ;
4  $\tau_{sub} \leftarrow \text{SubTemplate}(pt, pos)$ ;
5 tailPath  $\leftarrow \langle set, \tau_{sub} \rangle$ ;
6 return tailPath;

```

---

PATHNODES (lines 2-3) and TEMPLATES (line 4) from our index. To compute the sub-template we rely on a simple function, *sub-template* that retrieves the template for a path and returns its substring starting at the specified position (where the position is determined by the # symbols in the template). As an example, refer to Fig. 1 and its index in Fig. 3 with the tailpath for path with ID 23 (i.e. `pub-acceptedBy-conf1-type-Conference`) starting at position 2 (i.e. node `conf1`) is `conf1-type-Conference`.

Algorithm 3 describes the forward-concatenation mechanism to append a set of tailpaths *tailPaths* to an existing path *pt*, where an edge *p* connecting the last node of the path with the first node of the tailpath is provided. In the algorithm, for each tailpath (line 1) a new path *pt<sub>new</sub>* is created (lines 2-4). The function *JoinTemplates* concatenates two templates, updates the table TEMPLATES with the newly created template,

**Algorithm 3.** ForwardConcatenation.**Input** : A set  $tailPaths$  of tailpaths, a path  $pt$  to extend, an edge  $p$ **Output**: The tables PATHS, PATHNODES, and TEMPLATES updated.

---

```

1 foreach  $\langle pairs, \tau_{sub} \rangle \in tailPaths$  do
2    $pt_{new} \leftarrow NewID()$ ;
3    $\tau_{new} \leftarrow JoinTemplates(\tau, \tau_{sub})$ ;
4    $\langle n_{max}, pos_{max} \rangle \leftarrow MaxPos(pairs)$ ;
5    $PATHS \leftarrow PATHS \cup \langle pt_{new}, \tau_{new}, n_f + pos_{max}, n_{max} \rangle$ ;
6   foreach  $\langle pt, n, pos \rangle \in PATHNODES$  do
7      $PATHNODES \leftarrow PATHNODES \cup \langle pt_{new}, n, pos \rangle$ ;
8   foreach  $\langle m, pos \rangle \in pairs$  do
9      $PATHNODES \leftarrow PATHNODES \cup \langle pt_{new}, m, pos + l \rangle$ ;

```

---

and returns the identifier of such template. When updating TEMPLATES, if the template is already defined then no row is added to the table; the count value for the existing template is incremented and its identifier returned. Otherwise, a new row for the new template is inserted in the table. The function  $MaxPos$  is used to retrieve the pair  $\langle n, pos \rangle$  with highest position value from the set of nodes belonging to the tailpath. With this information defined, we can store the new path in PATHS (line 5).

Let us provide an example. Assume  $pt = \text{pub1-acceptedBy-conf1}$ ,  $tailPaths$  as  $\{DEXA\}$  and  $p$  as name. The new path is  $pt_{new} = \text{pub1-acceptedBy-conf1-name-DEXA}$ , with template  $\#-acceptedBy-\#-name-\#$ . To complete the definition, we must associate the newly created path with its nodes. Nodes belonging to  $pt_{new}$  are: those belonging to  $pt$  (lines 6-7), and those belonging to the tailpath (lines 8-9).

**Edge deletion.** The edge to be deleted is specified as a parameter to the Algorithm 4 in the form of an RDF triple  $\langle s, p, o \rangle$ .

**Algorithm 4.** Delete an edge.**Input** : A triple  $\langle s, p, o \rangle$  representing the edge with label  $p$  between nodes  $s$  and  $o$ .**Output**: The updated index.

---

```

1 foreach  $\langle pt, n_o, pos \rangle \in PATHNODES$  do
2   foreach  $\langle pt, n_s, pos - 1 \rangle \in PATHNODES$  do
3     if  $p = PropByPos(pt, pos-1)$  then
4        $TailPaths \leftarrow TailPaths \cup TailPath(pt, pos)$ ;
5        $PATHS \leftarrow PATHS \setminus \langle pt, -, - \rangle$ ;
6        $PATHNODES \leftarrow PATHNODES \setminus \langle pt, -, - \rangle$ ;
7        $DelTemplate(pt)$ ;
8 if  $\exists \langle pt, n_o, - \rangle \in pathnodes$  then
9    $ForwardConcatenation(TailPaths, \emptyset, \emptyset)$ ;

```

---

This operation requires deleting all paths that contain edge  $p$ . In our index, edges are not stored explicitly: they are encoded in the templates associated with the paths. In order to identify the paths containing  $p$ , we select those  $pt$  in which  $n_s, n_o$  (i.e. IDs associated to nodes  $s$  and  $o$  respectively) are directly connected (lines 1-2). Then, for each  $pt$  we verify if the edge between  $n_s$  and  $n_o$  is  $p$ . To this aim, we use the function *PropByPos* that takes as input  $pt$  and the position ( $pos - 1$ ) of  $n_s$  (i.e.  $n_o$  is in position  $pos$ ) and returns the edge outgoing from  $s$  (i.e. by accessing the template corresponding to  $pt$ ).

Let us now give an example using the graph depicted in Fig. 1. Assume we want to remove property `acceptedBy` between `pub1` and `conf1` (i.e. corresponding to the triple  $\langle pub1, acceptedBy, conf1 \rangle$ ). The IDs of the paths having nodes `pub1` immediately preceding `conf1` are 22, 23 and 24 (see Fig. 3). Consider the path with ID 24; `conf1` occurring at position 2. Thus, *PropByPos* (i.e. *PropByPos*(24, 1)) will access the template with ID 15 (i.e. `#-acceptedBy-#-name-#`) and return the sub-string `acceptedBy` between nodes in positions 1 and 2. Similarly we process also paths with IDs 22 and 23. Continuing the discussion of the algorithm, since  $n_o$  could become a source, we need to build the tailpaths from  $n_o$  (line 4) before deleting all  $pt$ . Thus we delete all  $pt$  and all corresponding nodes (lines 5-6). We must also update the corresponding COUNT in TEMPLATES by using the function *DelTemplate*. If  $n_o$  became a source (line 8), we then invoke the forward-concatenation on the tailpaths (line 9) to build all paths from  $n_o$ .

**Edge insertion.** The input to Algorithm 5 is a triple  $\langle s, p, o \rangle$  where  $p$  is the label for the new edge to add between (existing) resources  $s$  and  $o$ . We have to create the new

---

#### Algorithm 5. Insert a new edge

---

**Input** : A triple  $\langle s, p, o \rangle$  representing the edge with label  $p$  between nodes  $s$  and  $o$ .

**Output**: The updated index.

```

1 TailPaths  $\leftarrow \emptyset$ ;
2 foreach  $\langle pt_1, n_o, pos \rangle \in \text{PATHNODES}$  do
3   TailPaths  $\leftarrow \text{TailPaths} \cup \text{TailPath}(pt_1, pos)$ ;
4 foreach  $\langle pt_1, \tau, l, n_s \rangle \in \text{PATHS}$  do
5   if  $\exists \langle pt_1, n_o, \_ \rangle \in \text{PATHNODES}$  then
6     ForwardConcatenation(TailPaths,  $pt_1, p$ );
7 foreach  $\langle pt_1, n_o, 1 \rangle \in \text{PATHNODES}$  do
8   PATHS  $\leftarrow \text{PATHS} \setminus \langle pt_1, \_ , \_ , \_ \rangle$ ;
9   PATHNODES  $\leftarrow \text{PATHNODES} \setminus \langle pt_1, \_ , \_ \rangle$ ;
10  DelTemplate( $pt_1$ );

```

---

paths involving  $p$ . To this aim we have to concatenate paths ending in  $s$  (i.e. node  $n_s$ ) and tailpaths built from  $o$  (i.e. node  $n_o$ ) as shown in (lines 2-6).

For instance, suppose we want to re-introduce the property `acceptedBy` we have removed in the previous example. The input triple is  $\langle pub1, acceptedBy, conf1 \rangle$ . Only



the path  $pt$  with ID 31 ends in `pub1`. The tailpaths from node `conf1` are: (1) sub-path `conf1` with sub-template `#`, (2) sub-path `conf1-type-Conference` with sub-template `#-type-#`, and (3) `conf1-name-DEXA` with sub-template `#-name-#`. Connecting  $pt$  with such tailpaths through property `acceptedBy`, we obtain back paths with IDs 22, 23, 24. If  $n_o$  becomes a source, we delete all paths rooted in  $n_o$  (lines 7-10).

**Remaining maintenance operations.** The rest of the maintenance operations are simpler and intuitive, and are now discussed briefly. In *Node insertion*, we basically have to insert a new entry into `NODES`. Since it is not yet linked to other nodes, the new node is a source with an associated path (of length 1). Therefore, we must insert a path updating `PATHS`, `PATHNODES` and possibly, `TEMPLATES`. *Node deletion* is the inverse operation to node insertion. Assuming the node is not linked to any other, we have to delete one entry from tables: `NODES`, `PATHS`, `PATHNODES` and possibly, from `TEMPLATES` if the count in `TEMPLATES` reaches zero for the associated template. *Edge update* requires as input, the triple to identify which edge to update and a new value for such edge. Since edge information is encoded in templates, we must access and parse the template strings. Therefore, we retrieve the paths containing the input triple and we generate a new template for them with where the old value is replaced by the new one. In *Node update*, once identified the node, we only have to update its URI in table `NODES`.

**Computational Complexity.** In this section we present a discussion about the computational complexity of the creation and maintenance of our indexing scheme. Before commencing the discussion, let us introduce the notation we will use in the remainder of this section. Let  $R$  be the number of sources,  $E$  be the number of edges and  $V$  the number of vertexes. We indicate with  $PT$  the number of paths in the index. With  $PT_n$  we denote the number of paths containing a specific node  $n$ , and with  $PT_{n_1, n_2, \dots, n_k}$  the number of paths containing the node sequence  $n_1, n_2, \dots, n_k$ . Finally,  $TP$  indicates the number of tailpaths in the forward concatenation and  $L$  the length of a path.

The Index Creation is  $O(R \times (E + V))$ . Such algorithm is an implementation of BFS (i.e. notoriously it has complexity  $O(E + V)$ ) and it is invoked once for each source. Let us remark that our approach, initially, does not compute all the possible paths between a source and a node (leaf or intermediate), but only that ones ending into a sink: thus the complexity for the BFS is much lower than  $O(E + V)$ . Index update operations on a single node (e.g. insertion, deletion or update of a node), are trivial and it is straightforward to verify that they have complexity  $O(1)$ . The function `ForwardConcatenation` defined in Algorithm 3 is  $O(TP \times L)$ . In fact, it depends on how many tail paths ( $TP$ ) we have to concatenate in the creation of the new path; and the creation of a new path depends on its length ( $L$ ). In practical cases (also attested by experiments)  $L$  is rather smaller than  $TP$ . Therefore we can reformulate in  $O(TP)$ . The Algorithm 4 to delete an existing edge  $p$  connecting nodes  $s, n$  is  $O(PT_{s,o} \times L)$ . Deleting the path having the edge of interest implies to delete the occurring nodes (i.e.  $L$ ). Since the paths containing the triple  $\langle s, p, o \rangle$  are  $PT_{s,o}$ , deleting this information for all such paths spends  $O(PT_{s,o} \times L)$ . In case the node  $o$  becomes a source, then we also have the `ForwardConcatenation` on the tailpaths from  $o$ . Although this operation is merely a copy, its complexity is still  $O(PT_{s,o} \times L)$ . Summarizing, the overall complexity is  $2 \times (PT_{s,o} \times L) \in O(PT_{s,o} \times L)$ . Since  $L$  is rather small, we have  $O(PT_{s,o})$ . The update

```

Q1:
SELECT DISTINCT pn.PID
FROM PATHS AS pn
WHERE pn.F_NID in (
  SELECT NID
  FROM NODES AS n
  WHERE lcontains (URI,
    <inputTerm > ) )

Q2:
SELECT pn.PID ,
       pn.position
FROM PATHNODES AS pn
WHERE pn.NID =
  <inputNodeID>

Q3:
SELECT pn1.PID
FROM PATHNODES AS pn1
WHERE pn1.NID in (
  SELECT pn2.NID
  FROM PATHNODES AS pn2
  WHERE pn2.PID =
    <inputPathID> )

```

**Fig. 4.** Sample queries on the index

of an edge in a path has complexity  $O(PT_{s,o})$ . We assume the function *ReplaceEdge* to have complexity  $O(1)$ . Because the edge could belong to many paths, *ReplaceEdge* must be performed more times. Since the number of paths with such edge is  $PT_{s,o}$ , thus the complexity is  $O(PT_{s,o})$ . Algorithm 5 defines the insertion of a new edge  $p$  between two existing nodes  $s, o$ . The algorithm first takes the  $PT_o$  tailpaths from  $o$ : we have  $O(PT_o \times L)$ . Then it performs a forward-concatenation of the above tailpaths with all the paths ending in  $s$  (i.e. in the worst case  $PT_s$  paths). Thus the forward concatenation (i.e.  $O(PT_o \times L)$ ) must be performed for each path ending in  $s$ ; we conclude the overall complexity is  $O(PT_s \times PT_o \times L)$ . As we noticed before, in practical cases,  $L$  is rather small, which leads us to say that the complexity of the operation is  $O(PT_s \times PT_o)$ . In an extreme case we could have  $PT_o + PT_s = PT$ ; in this situation most of the paths ends in  $o$  and/or  $s$ . Therefore  $O(PT_o)$  or  $O(PT_s) \in O(PT)$ . In practical cases it is rare that the two nodes  $s, o$  concentrate all the paths, then  $O(PT_o)$  or  $O(PT_s) \ll O(PT)$ .

### 4.3 Index Querying

In this section, we provide a few examples, shown in Fig. 4, on how to query our index. Query **Q1** retrieves all paths having at least one node matching the input keyword `inputTerm`. This query uses the function `lcontains`, part of the Oracle SQL syntax, that exploits the Lucene full-text search capability. Although we use Oracle, as described in Section 5, this query can easily be adjusted to suit other SQL dialects. In fact, many modern DBMS supports Lucene by implementing their own variation on the SQL syntax. The query limits its attention to the final node (`F_NID`) of the paths because keywords only occurs in final nodes, as intermediary nodes only contain URI references. In a variation, **Q1** can be join with `TEMPLATES` to perform keyword search on paths' template, i.e. searching for keywords on the edges. The second query, **Q2**, retrieves all paths containing a generic node `inputNodeID` along with its position in each path. The node can be any between a source, intermediary or sink. This information can be exploited when the position of a node is a critical information to perform further analysis, like: reachability of such node from other nodes, the computation of the paths starting from or ending in such node. Last query, **Q3**, retrieves all the paths intersecting with a given path `inputPathID`. It first locates the nodes belonging to `inputPathID`, then the paths sharing such nodes. In keyword search systems, this query is useful when searching for connections (on the graph) between the nodes (or edges) matching some keyword. A keyword search system builds a solution (i.e. a subgraph) assembling intersecting paths with nodes (or edges) matching some keyword.

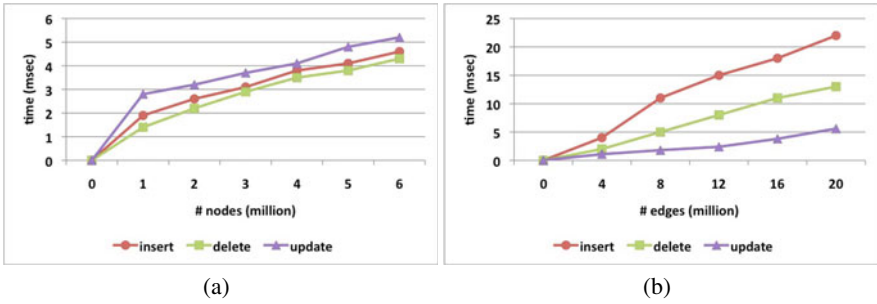


Fig. 5. Maintenance Scalability with respect to #nodes (a) and #edges (b)

## 5 Implementation

We implemented our path-oriented index into YAANII [5], a Java system for keyword search query over RDF datasets. All procedures for building and maintenance of our index have been serialized in PL/SQL operations and deployed in Oracle 11g v2. With the experiments discussed in this section we have measured how much the index improves query performance in YAANII. We used a widely-accepted benchmark of ten queries on DBLP for keyword search evaluation. DBLP (Digital Bibliography & Library Project, a computer science bibliography) is a dataset containing 27M triples about computer science publications. Due to space limitations we omit the query list here (see [11]). Experiments were conducted on a dual quad core 2.66GHz Intel Xeon, running Linux RedHat, with 8 GB of memory, 6 MB cache, and a 2-disk 1Tbyte striped RAID array. We evaluated the performance of: index building, index update and query execution. On top of DBLP we indexed roughly 17M of entries into the table PATHS, 28M into PATHNODES, 0,6M into TEMPLATES and 6M into NODES. The building task took a total 37 hours, and includes: the import of dataset from a single file encoding DBLP in NTRIPLE, and the execution of the BSF algorithm to compute the fullpaths. The final disk space required by the storage of the index was 718MB.

Fig. 5 collects the performance of maintenance operations (i.e. insertion, deletion and update of nodes and edges). The figure reports the scalability of such operations with respect to the increasing number of nodes (i.e. Fig. 5(a)) and edges (i.e. Fig. 5(b)). At each group of nodes (e.g. 1, 2, . . . , 6 millions) or edges (e.g. 4, 8, . . . , 27 millions), we inserted and updated 100 nodes (edges) and then we deleted them. Then we measured the average response time (ms) for one node (or edge). Opposite to the building step, the maintenance of the index follows good performance, satisfying practical scenarios with frequent updates of the dataset.

For query execution evaluation, we integrated our index in YAANII and we compared performance with the most related approaches: SEARCHWEBDB [18], bidirectional search [11] (we refer to it as BIDIRECT) and the several techniques based on graph indexing, i.e. 1000 BFS, 1000 METIS, 300 BFS, 300 METIS (see details in [11]). We ran the queries in [11] ten times and measured the average response time. Precisely, the total time of each query is the time for computing the top-10 answers. The query run-times are shown in Figure 6. In general BIDIRECT performs poorly, SEARCHWEBDB

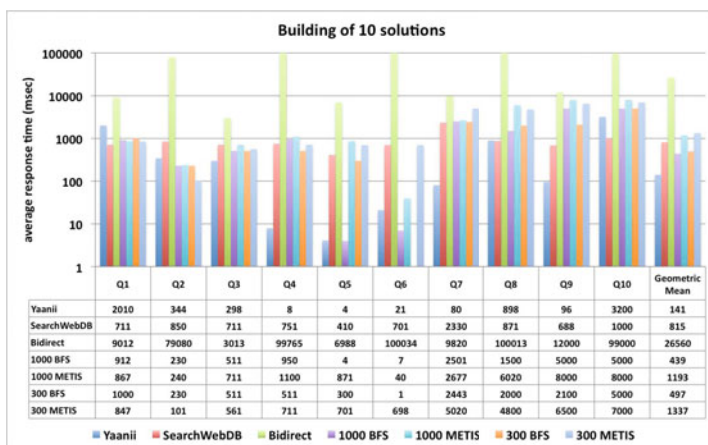


Fig. 6. Response Times

is comparable with BFS and better than METIS. YAANII with our index implemented is the best (on average) among the cited approaches: it was consistently the best for most of the queries; it outperforms all competitors by a large margin, improving times by nearly a factor varying from 3 to 188 in the geometric mean. As demonstrated in [2], the complexity of YAANII algorithm outperforms other approaches, but by employing our index the entire process, requiring frequent queries similar to Q1, Q2 and Q3 discussed in Section 4.3, speed-up significantly.

## 6 Conclusion and Future Work

The web of data is a powerful mechanism for both users and organisations but due to its size (and the size of many individual information components) provides a significant challenge when searching for information needs. In this paper, we presented a path-oriented indexing scheme for the large graph structures that contain the semantic datasets comprising the web of data. The key feature of the index is that it exposes the structural characteristics of the graph: its paths, the structure (schema) of these paths, and the information on the reachability of nodes. By exploiting this information, we can expedite query execution, especially for keyword based query systems, where query results are built on the basis of connections between nodes matching keywords. As graph exploration is a complex and time consuming task, usually computed on-the-fly during query processing, our index facilitates a far more efficient query process. We developed a prototype system by integrating our index into YAANII, a system for keyword searching query RDF using path computations. Results show that the index significantly increases the performance of YAANII, outperforming other approaches while still providing the desired, exhaustive search. Current research is focused on: an investigation into mathematical properties to weight relevant paths and templates with respect to the graph; a more compact index and compression technique to reduce space consumption; and further optimizations for both index creation and maintenance.

## References

1. Bizer, C., Cyganiak, R.: D2R server: Publishing relational databases on the semantic web. In: Proc. of ISWC (2006)
2. Cappellari, P., De Virgilio, R., Maccioni, A., Miscione, M.: Keyword based search over semantic data in polynomial time. In: Proc. of ICDE Workshops, pp. 203–208 (2010)
3. Cheng, J., Ke, Y., Ng, W., Lu, A.: Fg-index: towards verification-free query processing on graph databases. In: Proc. of SIGMOD, pp. 857–872 (2007)
4. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: Xsearch: a semantic search engine for xml. In: Proc. of VLDB, pp. 45–56 (2003)
5. De Virgilio, R., Cappellari, P., Miscione, M.: Cluster-based exploration for effective keyword search over semantic datasets. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 205–218. Springer, Heidelberg (2009)
6. Fellbaum, C. (ed.): WordNet: an electronic lexical database. MIT Press, Cambridge (1998)
7. Giugno, R., Shasha, D.: Graphgrep: A fast and universal method for querying graphs. In: Proc. of ICPR, pp. 112–115 (2002)
8. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: ranked keyword search over xml documents. In: Proc. of SIGMOD, pp. 16–27 (2003)
9. He, H., Wang, H., Yang, J., Yu, P.S.: Blinks: ranked keyword searches on graphs. In: Proc. of SIGMOD, pp. 305–316 (2007)
10. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient ir-style keyword search over relational databases. In: Proc. of VLDB, pp. 850–861 (2003)
11. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desa, R., Karambelkar, H.: Bidirectional expansion for keyword search on graph databases. In: Proc. of VLDB, pp. 505–516 (2005)
12. Kimelfeld, B., Sagiv, Y.: Finding and approximating top-k answers in keyword proximity search. In: Proc. of PODS, pp. 173–182 (2006)
13. Kolas, D., Emmons, I., Dean, M.: Efficient linked-list rdf indexing in parliament. In: 5th Int. Workshop on Scalable Semantic Web Knowledge Base Systems, SSWS (2009)
14. Liu, F., Yu, C., Meng, W., Chowdhury, A.: Effective keyword search in relational databases. In: Proc. of SIGMOD, pp. 563–574 (2006)
15. Markowetz, A., Yang, Y., Papadias, D.: Reachability indexes for relational keyword search. In: Proc. of ICDE, pp. 1163–1166 (2009)
16. Neumann, T., Weikum, G.: x-rdf-3x: Fast querying, high update rates, and consistency for rdf databases. In: PVLDB, vol. 3(1), pp. 256–263 (2010)
17. Tian, Y., Patel, J.M.: Tale: A tool for approximate large graph matching. In: Proc. of ICDE, pp. 963–972 (2008)
18. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In: Proc. of ICDE, pp. 405–416 (2009)
19. Weiss, C., Karras, P., Bernstein, A.: Hexastore: sextuple indexing for semantic web data management. In: PVLDB, vol. 1(1), pp. 1008–1019 (2008)
20. Yan, X., Yu, P.S., Han, J.: Graph indexing: A frequent structure-based approach. In: Proc. of SIGMOD, pp. 335–346 (2004)
21. Zhang, S., Hu, M., Yang, J.: Treepi: A novel graph indexing method. In: Proc. of ICDE, pp. 966–975 (2007)
22. Zhang, S., Li, S., Yang, J.: Gaddi: distance index based subgraph matching in biological networks. In: Proc. of EDBT, pp. 192–203 (2009)

# Variable Length Compression for Bitmap Indices

Fabian Corrales<sup>1</sup>, David Chiu<sup>2</sup>, and Jason Sawin<sup>1</sup>

<sup>1</sup> Department of Mathematics and Computer Science, University of Puget Sound

<sup>2</sup> School of Engineering and Computer Science, Washington State University

**Abstract.** Modern large-scale applications are generating staggering amounts of data. In an effort to summarize and index these data sets, databases often use *bitmap indices*. These indices have become widely adopted due to their dual properties of (1) being able to leverage fast bit-wise operations for query processing and (2) compressibility. Today, two pervasive bitmap compression schemes employ a variation of run-length encoding, aligned over bytes (BBC) and words (WAH), respectively. While BBC typically offers high compression ratios, WAH can achieve faster query processing, but often at the cost of space. Recent work has further shown that reordering the rows of a bitmap can dramatically increase compression. However, these sorted bitmaps often display patterns of changing run-lengths that are not optimal for a byte nor a word alignment. We present a general framework to facilitate a variable length compression scheme. Given a bitmap, our algorithm is able to use different encoding lengths for compression on a per-column basis. We further present an algorithm that efficiently processes queries when encoding lengths share a common integer factor. Our empirical study shows that in the best case our approach can out-compress BBC by 30% and WAH by 70%, for real data sets. Furthermore, we report a query processing speedup of  $1.6\times$  over BBC and  $1.25\times$  over WAH. We will also show that these numbers drastically improve in our synthetic, uncorrelated data sets.

## 1 Introduction

Many research projects, vital for advancing our understanding of the world, have become prohibitively data-intensive. For example, exploration within bioinformatics generates terabytes of data per day [22]. In the field of high energy physics, the Large Hadron Collider at CERN is projected to generate 15 petabytes of data annually [7]. To facilitate data analysis, such projects may store their results in databases which employ advanced indexing techniques. Since the bulk of this scientific data is read-only, infrequently updated, and relatively easy to categorize into groups, bitmaps [18, 15] are highly amenable and widely used to index such data sets.

A bitmap index is a two dimensional array  $B[m, n]$  where the columns denote a series of  $n$  bins and the rows correspond to  $m$  tuples in a relation. To transform a table into a bitmap, each attribute is first partitioned into a series of bins that might denote a point or a range of values. An element  $b_{i,j} \in B = 1$  if the  $j$ th attribute in the  $i$ th tuple falls into the specified range, and 0 otherwise. While their representation can be large in terms of space, bitmaps can be queried using highly efficient low-level bitwise operations. To exemplify, consider an *age* attribute that might be partitioned into the following three bins:  $a_1 = [0, 20]$ ,  $a_2 = [21, 40]$ ,  $a_3 = [41, \infty]$ .

Table 1 shows a simple bitmap index on two attributes: *age* and *gender* after binning. To find everyone under 21 or over 40, the processor can simply apply a bitwise *OR* of  $a_1$  and  $a_3$ , then retrieve the succeeding tuples  $t_2$  and  $t_3$  from disk.

**Table 1.** An Example Bitmap

Tuples	Bins				
	Age			Gender	
	$a_1$	$a_2$	$a_3$	$g_f$	$g_m$
$t_1$	0	1	0	0	1
$t_2$	1	0	0	1	0
$t_3$	0	0	1	0	1

The tradeoff to supporting such fast querying, is storage costs. Bitmap indices can become very large. But fortunately, they are typically sparse and highly suitable for compression via run-length encoding (RLE), where bit sequences can be summarized using a nominal most significant bit (MSB) followed by the length of its run. A well-known problem of RLE is the fact that it may sometimes yield larger bitmaps. For instance, assuming a byte-based RLE, uniformly distributed bit patterns, such as (1010), will be “compressed” into (10000001 00000001 10000001 00000001). Clearly, the original bit sequence could be more efficiently stored as a *literal*. Today’s compression schemes utilize a run-length *hybrid*, i.e., a sequence of encoded bits can denote either a *literal* or a *run/fill* of  $n$  bits.

While the overhead of decoding initially appears to conflict with query performance, current memory-aligned compression schemes have found ways to improve both compression and query performance simultaneously [2,20]. The Byte-aligned Bitmap Compression (BBC) [2] and the Word Aligned Hybrid Code (WAH) [20] are commonly used to compress and query bitmaps in databases. While both BBC and WAH are memory-aligned, they differ in granularity (byte versus word), which affects compression ratio and query execution time. As such, BBC typically generates smaller bitmaps, and WAH, due to its word-based representation, excels at compressing extremely long runs and allows for faster queries by amortizing multiple reads to extract bytes from words.

Because both BBC and WAH are run-length dependent, and tuple ordering is arbitrary in a database, efforts in row reorganization can be leveraged to produce longer runs and achieve better compression. One drawback to certain tuple reorganization schemes, such as Gray code ordering, is that average run length degrades as dimensionality increases [16]. In Gray code ordering the first several bit vectors of a bitmap may contain very long runs, but they become progressively shorter in each additional bit vectors. In fact, the highest dimensional bit vectors may still exhibit a uniform distribution of bits. We posit that it may be sensible to apply coarser encodings (e.g., WAH) to the initial bins for aggressive compression of longer runs. As dimensionality increases, and runs gradually become shorter, progressively finer encoding schemes may be used to achieve greater compression. In the highest dimensionality, where runs are even less infrequent, it may again be favorable to coarsen the encodings, because word-aligned encodings can store and process literals more efficiently.

This paper makes the following contributions:

- We have designed and implemented a novel *generalized* framework, Variable Length Compression (VLC), for encoding variably granular bitmap indices. Given a bitmap, our VLC is able to use different encoding lengths for compression on a per-column basis.
- We have designed an algorithm to compress bit vectors, which inputs a *tuning parameter* that is used to tradeoff encoding space and querying time.
- We have conducted an extensive analysis of VLC on several real data sets and compare results against state-of-the-art compression techniques. We show that VLC

can out-compress Gray code ordered BBC and WAH by 30% and 70% respectively in the best case, on real data sets. We also report a speedup of  $1.6\times$  over BBC and  $1.25\times$  over WAH.

These contributions provide a method for faster querying on large databases as well as greater compression ratios. Our proposed VLC scheme also allows users tune the compression of their bitmap indices. For example, if certain columns are to be queried at higher rates they can be compressed using the larger encoding lengths to achieve faster queries. To maintain compression efficiency the less frequently queried columns can be compressed with smaller encoding lengths.

The remainder of this paper is organized as follows. In Section 2, we present the necessary background on bitmap compression, including BBC, WAH, and tuple re-ordering. Section 3 describes our Variable Length Compression framework in depth, detailing both compression and query processing algorithms. We present our experimental results in Section 4. Section 5 discusses related efforts in bitmap compression, and we conclude our findings in Section 6.

## 2 Background

Bitmap compression is a well-studied field, with its roots anchored in classic run-length encoding (RLE) schemes. However, traditional run-length techniques cannot be directly applied to bitmap indices because the bit vectors must first be decompressed to answer queries. This overhead would quickly dominate query processing time. Therefore, it is highly desirable to have run-length compression schemes that can answer queries by directly examining the bit vectors in their compressed state. In this section we present the background on current techniques used to compress bitmap indices that achieve this fast querying.

### 2.1 Byte-Aligned Bitmap Code (BBC)

Run-length encoding schemes achieve compression when sequences of consecutive identical bits, or “runs”, are present. BBC [2] is an 8-bit hybrid RLE representation in the form of a *literal* or a *fill*. The MSB, known as the *flag* bit, marks the encoding type. In turn, a byte  $0xxxxxxx$  denotes that the least significant 7 bits is a literal representation of the actual bit string. In contrast,  $1xxxxxxxx$  encodes a fill which compactly represents runs of consecutive  $x$ 's. Here,  $x$  is the *fill bit* which encodes the value of the bits in the run, and the remaining 6 bits are used for the length (in multiples of 7), e.g.,  $11001010$  represents the sequence of 70 1's.

BBC is compelling in that the query execution time is directly proportional to the rate of compression. For example, suppose a database contains 77 rows and two bit vectors:  $v_1$  and  $v_2$ . Assume that  $v_1$  contains the literal  $0101010$  followed by a run of 70 consecutive 1's. Let  $v_2$  contain a sequence of 70 0's followed by the literal  $0100000$ . In BBC format,  $v_1$  would be encoded as  $(00101010\ 11001010)$  and similarly,  $v_2 = (10001010\ 00100000)$ . Now envision a query which invokes  $v_1 \wedge v_2$ . The query processor would read the first byte from both  $v_1$  and  $v_2$ . By decoding the most significant bit, the query processor determines that it has read a 7-bit literal from  $v_1$  and a run of  $(10 \times 7) = 70$  0's from  $v_2$ . Next, the literal from  $v_1$  is AND'ed with a fill of seven  $0000000$  from  $v_2$ . Progressing further, the query processor reads and decodes the next byte from  $v_1$ . It is important to note that only seven 0's have been processed from



the fill in  $v_2$ . Thus, all that is required is simply decrement of the fill count from 10 to 9. This demonstrates why BBC fills must be a multiple of 7. The next byte of  $v_1$  is decoded as a run of 70 consecutive 1's. The next 9 AND operations can be carried out in one step by making the AND comparison once and reporting its results in the same compressed form. The run-length count for  $v_1$  is updated to 1, and  $v_2$  to 0. Thus  $63 = (9 \times 7)$  bits have been compared without having to decode even once. After the 9th iteration,  $v_2$ 's fills are exhausted, prompting a read of the next byte from  $v_2$ . Finally, the remaining 7 bits from both bins are AND'ed to complete the query. BBC's efficiency comes from the presence of fills, which effectively allows the processor to amortize the number of necessary memory accesses.

## 2.2 Word-Aligned Hybrid Code (WAH)

WAH [20, 19], unlike BBC, uses a 31 bit representation (32 bits including the flag bit). This representation offers several benefits over BBC—one being that for certain bitmaps, WAH can achieve significant speedup in query processing time when compared to BBC. This speedup is due to the fact that memory is typically fetched by the CPU a word at a time. By using a word-aligned encoding, WAH avoids the overhead of further extracting bytes within a word that is incurred by BBC. Thus, WAH not only compresses literals more efficiently than BBC (using 4 less flag bits per 31 bits), but it can also process bitwise operations much faster over literals by avoiding the overhead of byte extraction and parsing/decoding to determine if the byte is indeed a literal.

In terms of compressing runs, however, WAH typically pales compared to BBC. This is often due to the fact that WAH's fills can encode  $2^{30} - 1$  multiples of 31 consecutive identical bits (i.e., a maximum fill length of 33,285,996,513). In practice, runs of this size are unlikely, which implies that many of the fill bits are unused. On the other hand, note that the maximum number of consecutive bits that a BBC fill can represent is  $(2^6 - 1) \times 7 = 441$ . In large-scale or highly sparse databases, it is likely that a run can continue far beyond this threshold, which means there can still be cases where WAH will yield more efficient encodings for runs.

## 2.3 Row Reordering of Bitmaps

As described above, WAH and BBC can achieve greater compression for bitmaps that contain longer average run-lengths.

Recent work has shown that the average run-length of a bitmap can be greatly improved by reordering the rows [12, 16, 10, 3]. Finding an optimal order of rows, however, has been proven to be NP-Complete, and lexicographical and Gray code ordering are widely used heuristics. Pinar, Tao, and Ferhatosmanoglu showed that compression ratio's can be improved by a factor of 10 for some bitmap indices if a Gray code (i.e., consecutive rows differ only by a single bit) ordering is applied [16]. Figure 1 shows the effects of lexicographical and Gray code ordering on bitmaps containing 3 vectors  $v_1, v_2, v_3$ . The white space represents 0's and the black represents 1's. Notice that both reordering algorithms tend to produce longer runs in the first few bit vectors, but deteriorate into shorter runs (and worse, a random distribution) of bits for the higher vectors.

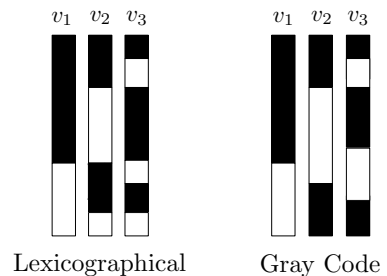


Fig. 1. Row Ordering Techniques

In situations like this, it would be desirable to employ a varying sized compression scheme for each bit vector. In this work, we assume that row reordering is a preprocessing step, and we implemented Gray code ordered bitmaps in our experimental results.

### 3 Variable Length Compression

In this section, we initially discuss how using variable bit-segment lengths presents opportunities to improve compression ratios beyond current state-of-the-art techniques, e.g., BBC and WAH. We then describe our compression technique, *Variable Length Compression (VLC)*, in detail.

Due to their use of fixed bit-segment lengths to encode bit vectors, neither WAH nor BBC generate optimal compression. To exemplify, recall that row reordered bitmaps produce long runs in the first several bit vectors, but increasingly shorter runs in the later vectors. WAH's 31-bit *segment length* (32 bits including the 1 flag bit) is ideal for the first several bit vectors that potentially contain extremely long runs. But after these first few vectors, the rest might tend to have an average run-length smaller than 62 (the shortest run-length multiple that WAH can compress), there is a higher likelihood that many shorter runs must be represented as WAH literals, which squanders compression opportunities. Conversely, BBC's maximum fill code,  $1x1111111$ , can only represent a run of  $63 \times 7 = 441$   $x$ 's. With its 7-bit fixed *segment length*, BBC cannot efficiently represent the long runs of the first several vectors. Any run longer than 441 would thus require another byte to be used.

We posit that we can attain a balanced tradeoff between these representations by using variably-sized bit *segment lengths*. To this end, we propose a novel run-length compression scheme *Variable Length Compression (VLC)* that can vary the segment lengths used for compression on a per bit vector basis. The flexibility of VLC enables us to compress the initial bit vectors of a row reordered bitmap using a longer segment length, while using a shorter length on later bit vectors. While a more robust compression can be expected using VLC, a challenge is maintaining efficient query processing speeds.

#### 3.1 Variable Compression Scheme

For each bit vector in the bitmap VLC compression performs the following steps: (1) Determining segment length for bit-vector compression, (2) Vector segmentation, and (3) Word packing.

The goal of the *Segment Length Determination (SLD)* algorithm is to determine an optimal segment length for a given bit-vector. In general, given a bit vector,  $v = (b_1, \dots, b_n)$ , SLD returns an integer value  $seg\_len \mid L < seg\_len < H$ . In this paper, we assume  $H$  to be the word-size,  $H = 32$ , and  $L = 2$ , since 2-bit segments cannot represent fills. In this work we consider two SLD approaches: `All Possible` is a brute force algorithm which simulates the compression of  $v$  using all possible segment lengths,  $seg\_len = 3, \dots, 31$ . For each segment length, `All Possible` computes the number of words needed to store the compressed bit vector, and it returns the length that would achieve the best compression. If multiple segment lengths generate the same compression ratio, then the largest segment length is returned to reduce the amount of parsing required when the bit vector is queried—this follows the same insight behind WAH versus BBC.

Another SLD heuristic we consider is `Common Factor`, which is similar to `All Possible` in that it simulates the compression of individual bit vectors. However,

Common Factor also takes as input an integer parameter, *base*, which is used to determine the set of segment lengths that will be used in the simulation. Specifically, it will only consider *seg\_len* where  $seg\_len \in \{x | 3 \leq x \leq 31 \wedge x \equiv 0 \pmod{base}\}$ . The basis for this heuristic is to ensure that  $gcd(v_1, v_2) \geq base$  any two compressed bit vectors  $v_1$  and  $v_2$ . As it will become clear later, this property greatly improves query processing speed.

After *seg\_len* is determined, the *Bit Vector Segmentation* process is applied. A compressed bit segment  $v_c$  is defined as a sequence of *seg\_len* bits,

$$v_c = \begin{cases} 0 \bullet x_1 \bullet \dots \bullet x_{seg\_len} & \text{(literal)} \\ 1 \bullet x \bullet n_1 \dots \bullet n_{seg\_len-1} & \text{(fill)} \end{cases}$$

where  $\bullet$  denotes concatenation. In the former case, the initial bit 0 denotes an uncompressed segment from  $v$ , and the succeeding sequence  $x_1, \dots, x_{seg\_len}$  denotes the literal. In the second case,  $v_c$  can represent a run by specifying 1 as the flag bit. The next bit  $x$  is the fill bit, and  $n_1 \dots, n_{seg\_len-1}$  is a number (base 2) denoting the multiple of a run of *seg\_len* consecutive  $x$  bits. For example, if  $seg\_len = 4$ , the segment 10011 is the compressed representation of 000000000000, i.e., a run of  $3 \times 4 (= 12)$  0's.

Given this code representation, the algorithm proceeds as follows. Beginning with the first bit in an uncompressed vector  $v$ , we let  $v'$  denote the next *seg\_len* bits in  $v$ . We initially encode  $v'$  as a literal, that is,  $v_c = 0 \bullet v'$ . Next,  $v'$  is assigned the subsequent *seg\_len* bit sequence in  $v$ . If  $v'$  is a sequence of identical bits  $x$ , then we verify if it can be coalesced with  $v_c$ . If  $v'$  is not a run of *seg\_len* bits, the  $v_c$  is first written to disk, and then again assigned the literal  $0 \bullet v'$ . If  $v_c$  indeed is a single literal containing a (*seg\_len*)-bit run of  $x$ , then  $v_c$  is converted to a fill segment,  $1 \bullet x \bullet 0 \dots 010$ . A more general case occurs if  $v_c$  is already a fill-segment. When this occurs, we simply add 1 to the run length portion. As  $v'$  continues to be assigned subsequent segments, the above steps are repeated.

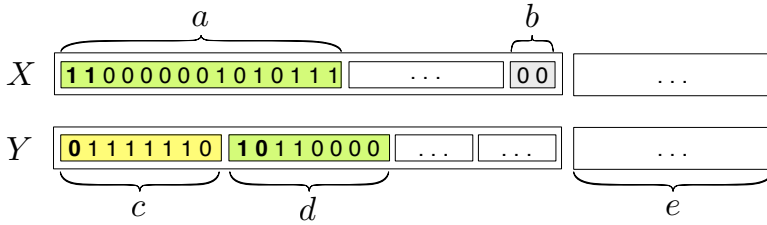
Finally, *Word Packing* is used to reduce the parsing cost when executing a query over VLC segments. Segments (both runs and literals) are *fit* into words. For example, if  $seg\_len = 3$ , then VLC packs 8 segments (including their flag bits) in one 32 bit word. If  $32 \not\equiv 0 \pmod{seg\_len + 1}$  then VLC appends  $32 \pmod{seg\_len + 1}$  0's to the end of each word. These superfluous bits are called *pad bits* and they are ignored by the query algorithm. Our approach requires that each compressed bit vector be prefaced a header byte, which stores the segment encoding length used.

In Figure 2, we show two 32-bit words, in vectors  $X$  and  $Y$ , sharing a common  $gcd$  of 7.  $X$  is coded in  $seg\_len = 14$ , and  $Y$  in  $seg\_len = 7$ . Segment  $a$  is a fill, denoting a run of  $seg\_len \times 87 (= 1218)$  1's. In other words,  $a$  is 87 consecutive 14-bit segments of 1's. Because each 14-bit segment is coded using 15 bits, two such segments can be packed into a word, with the last remaining 2 bits  $b$  being padding. In the bottom word  $Y$ , segment  $c$  is a 7-bit literal 1111110, and  $d$  is a run of  $seg\_len \times 48 (= 336)$  0's.  $e$  simply denotes the remaining words in either vector.

### 3.2 Query Processing

Algorithms 1 and 2 are the query processing procedures over two bit vectors,  $X$  and  $Y$  from Figure 2. As a running example, we consider performing query using a logical *op* between  $X$  and  $Y$ .

Initially, we declare an empty vector  $Z$  to hold the results, and assign its base to  $gcd(X, Y) = 7$  (Alg1:lines 1-3). Next, we loop through all segments of  $X$  and  $Y$



**Fig. 2.** Example of VLC with  $segLen = 14$  and  $gcd = 7$

(Alg1:line 4), and in our example,  $X.active$  is assigned to  $a$  and  $Y.active$  is assigned to  $c$  (Alg1:line 6). Next, both segments are decoded (see Algorithm 2), and this subprocedure invokes one of two cases. If the given segment  $seg$  is a fill, then its base  $reduced$  down to the  $gcd$ , effectively producing  $(base/gcd) \times$  its current segment run-length (Alg2:lines 4-6). In the example, recall that  $a$  represents 87 consecutive 14-bit segments of 1's, so  $a.runLen = 87$ . However, since the  $gcd$  is 7, we must reduce  $a$ 's base down to 7, which effectively produces  $87 \times (14/7) = 174$  7-bit segments in  $a$ . On the other hand, if  $a$  were a literal, then it is split into  $(base/gcd)$  separate  $gcd$ -bit literals (Alg2:lines 3-9).

---

**Algorithm 1.** ColumnQuery( $X, Y, op$ )

---

```

1: Let  $Z$  be the result vector (compressed form)
2:  $g \leftarrow gcd(X.segLen, Y.segLen)$ 
3:  $Z.segLen \leftarrow g$ 
4: while  $X.hasNextSegment()$  or  $Y.hasNextSegment()$  do
5:   {*modify the segments to match the GCD*}
6:   Let  $X.active$  and  $Y.active$  be the next segment read from  $X$  and  $Y$ 
7:   decode( $X.active, X.segLen, g$ ); decode( $Y.active, Y.segLen, g$ );
8:   if isFill( $X.active$ ) and isFill( $Y.active$ ) then
9:      $n \leftarrow \min(X.active.runLen, Y.active.runLen)$ ;
10:     $Z.appendFill(n, (X.active.fillBit \ op \ Y.active.fillBit))$ ;
11:     $X.active.runLen \leftarrow X.active.runLen - n$ ;
12:     $Y.active.runLen \leftarrow Y.active.runLen - n$ 
13:   else if isFill( $X.active$ ) and isLit( $Y.active$ ) then
14:      $Z.appendLit(Y.getNextLiteral() \ op \ X.active.fillValue)$ ;
15:      $Y.litCount \leftarrow Y.litCount - 1$ ;  $X.active.runLen \leftarrow X.active.runLen - 1$ ;
16:   else if isLit( $X.active$ ) and isLit( $Y.active$ ) then
17:      $Z.appendLit((X.active.getLiteral() \ op \ Y.active.getLiteral()))$ 
18:      $X.litCount \leftarrow X.litCount - 1$ ;  $Y.litCount \leftarrow Y.litCount - 1$ ;
19:   end if
20: end while
21: return  $Z$ 

```

---

Returning to Algorithm 1, the above decoding procedure prepares the two vectors to share the same base, which allows for easy application of the bitwise  $op$ . Then there are three cases: (1) both  $X$  and  $Y$  are fill words (Alg1:lines 8-11), (2) only  $X$  is a fill and  $Y$  is a literal (Alg1:lines 12-14), and (3) both  $X$  and  $Y$  are literals (Alg1:lines

**Algorithm 2.** `decode(seg, base, gcd)`


---

```

1: if isFill(seg) then
2:   seg.runLen  $\leftarrow$  seg.runLen  $\times$  (base/gcd)
3: else
4:   temp  $\leftarrow$  seg.getLiteral(); { * Binary representation of literal * }
5:   seg.litCount  $\leftarrow$  base/gcd;
6:   for each partition p of gcd consecutive bits in temp do
7:     seg.addLiteral(p);
8:   end for
9: end if

```

---

15-18).<sup>8</sup> In our example from Figure 2, Case 2 is invoked on segments *a* and *c*. We apply the bitwise *op* across *X.fillValue* and *Y.nextLiteral*(), which results in appending (1111111) *op* (1111110) to *Z*. Because we have only processed one literal, *X*'s run-length counter is decremented by 1 (Alg1:line 15). However, while *Y* is subsequently assigned the next parsed segment *d*, *X* avoids this overhead, which can be significant if *d* happened to exit in the next word, causing *Y* to read from memory.

If Case 1 is invoked, that is, when both *X* and *Y* are fills, we can simply apply *op* to the single fill bit, and implicitly know that its result applies to all bits until the end of either *X*'s or *Y*'s current segment.<sup>s</sup> Thus, without corresponding memory accesses, we can process  $\min(X.\text{active.runLen}, Y.\text{active.runLen}) \times \text{gcd}$  bits in  $O(1)$  operation of updating the fill counts (Alg1:lines 8-12). Because of this property, the query processing time for extremely long runs can be done in sublinear time.

## 4 Experimental Evaluation

In this section, we evaluate the compression ratios among BBC, WAH, and our VLC variants over both real and synthetic data sets. We further analyze query processing performance. All experiments in this section were executed on a Java 1.6 implementation with -Xmx1024m set running on a 32-bit Windows 7 machine with 2.0 GB RAM and an Intel Dual Core 2.4GHz processor. The bitmap data sets on which we experiment in this study are described as follows.

- HEP (272MB) is from a real high-energy physics application containing 12 attributes. Each attribute was split into ranges from 2 to 12 bins, which results in a total of 122 columns comprised of 2,173,762 rows.
- Histo (21MB) is from an image database. The tuples represent images, and 192 columns have been extracted as color histograms of these images. The bitmap contains 112,361 rows and 192 columns.
- Landsat (238MB) is derived from real satellite images, whose SVD transformation produces 275,465 rows and 522 columns.
- Stock (6.7MB) contains approximately 1080 days' worth of stock data for 6,500 companies, resulting in a high-dimensional bitmap containing 6,500 rows and 1080 columns.
- Uni (10.3MB) is a synthetic dataset generated with random bit distribution over 100,000 rows and 100 columns.

---

\* Note that a 4th case exists also, which is the reverse of case (2), but it is redundant here.

As an optimizing preprocessing step, all above data sets have initially been Gray code ordered using the algorithm presented in [16].

#### 4.1 Data Compression Analysis

We implemented WAH, BBC, and VLC and compressed all aforementioned data sets. With VLC, we varied across all segment lengths  $3 \leq \text{seg\_len} \leq 31$ . However, due to space constraints, we only report four representative configurations: `vlc-opt`, `vlc-4`, `vlc-7`, and `vlc-9`. The `vlc-opt` setting corresponds to using the All Possible Segment Length Determination (SLD) algorithm to find the optimal segmentation length for each bit vector. The compression generated by `vlc-opt` represents the best possible compression our implementation of VLC can achieve. The `vlc-4`, `vlc-7`, and `vlc-9` results are produced by using the Common Factor SLD algorithm on a *base* of 4, 7, and 9 respectively. For instance, bitmaps compressed in `vlc-4` may contain bit vectors encoded in segment lengths of 4, 8, 12, 16, 20, 24, and 28.

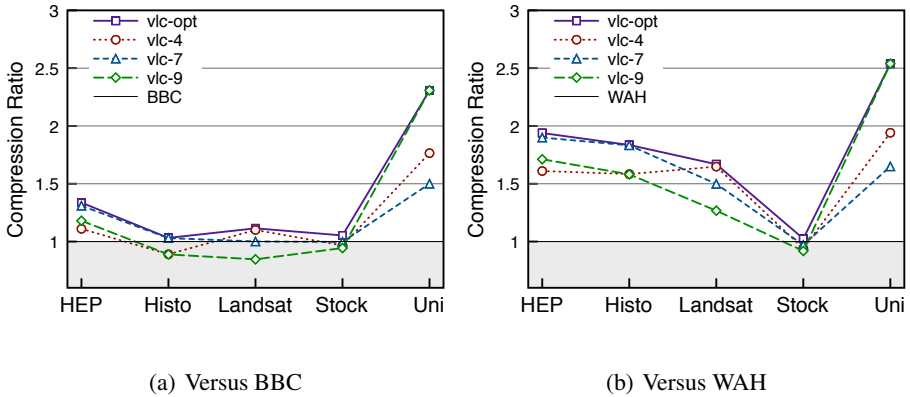
**Table 2.** Data Compression Size

Dataset	Compression (MB)						
	Orig	WAH	BBC	vlc-opt	vlc-4	vlc-7	vlc-9
HEP	272.0	2.251	1.552	1.161	1.398	1.185	1.315
Histo	21.4	1.05	0.59	0.572	0.663	0.573	0.664
Landsat	238.0	28.001	18.699	16.779	16.99	18.683	22.098
Stock	6.7	0.62	0.637	0.605	0.659	0.639	0.675
Uni	10.2	0.033	0.03	0.013	0.017	0.02	0.013

Table 2 presents the size (in MB) of the compressed data sets. As expected, we observe that `vlc-opt` outperforms all configurations, but because the bit vectors are not *gcd*-aware, the misalignment of the segments will adversely affect query performance. Thus, we emphasize the “closeness” of the `vlc-*` versions to `vlc-opt` and that most `vlc-*` configurations also out-compress both WAH and BBC. For further comparison, we juxtapose the compression ratio of VLC over BBC and WAH in Figures 3(a) and 3(b) respectively. Expectedly, most `vlc-*` versions provide a modest improvement when compared to BBC for most data sets. We can observe that for *Histo*, *Landsat*, and *Stock*, BBC slightly out-compresses `vlc-9`. We believe this is due to BBC’s aggressive compression of shorter runs, which appears frequently in these data sets.

Figure 3(b) depicts the comparison of VLC to WAH. Because of WAH’s longer fixed segments, it is expected that all `vlc` versions should out-compress WAH, unless in the presence of massive amounts of *exponentially* long runs, which are rare. In the best case, we can observe improvement of  $1.7\times$  (for real data) and  $2.54\times$  (for synthetic data) the compression rates of WAH using VLC. We can observe the best results for the synthetic *Uni* dataset, where it becomes very clear that the optimal compression segment length is in between the extremes of BBC and WAH. Out of 100 bit vectors, the `vlc-opt` algorithm compressed 50 of these in base-9, 3 in base-13, and 47 in base-14. In `vlc-9`, which is near optimal, is compressing 50 bit vectors in base-9, 48 columns in base-18, and 2 columns in base-27.

An anomalous data set is *Stock*, where compression gains appear difficult to achieve. We believe there are two reasons contributing to this observation. Due to the



**Fig. 3.** Comparison of Compression Ratio

Stock data’s dimensionality (1080 columns), after the first several columns, Gray code tuple ordering eventually deteriorates and begins generating columns with very short runs, or worst case, uniformly distributed bits. Adding to this effect is that Stock’s number of rows is small, which implies the opportunities for longer runs is made further infrequent. This means most of the later columns are probably being represented as literals in all compression schemes. This theory is supported by the fact that WAH actually out-compresses most schemes (an outlier). Due to the fact that WAH only uses one flag bit per 31-bit literal, it is by far the most efficient way to store long literals in all schemes.

### 4.2 Evaluation of Query Processing Times

In this subsection, we present the query processing times. For each data set, a set of 10 queries, which vary in the amount of tuples retrieved, was generated. To execute these queries, we split each bitmap vertically into 4 equal-sized sets of bit vectors  $B_1, B_2, B_3, B_4$ . Each query inputs 2 bit vectors  $X$  and  $Y$  such that  $\forall_{i,j} : \text{select } X \in B_i, Y \in B_j$  randomly. In other words,  $X$  and  $Y$  are randomly selected from each set of bit vectors, and we query against all combinations of bit vectors. We submitted the set of 10 queries repeatedly for all  $B_i, B_j$  combinations, and averaged the execution time to process all 10 queries.

This experimental protocol was selected in an attempt to avoid any segmentation length bias. By randomly selecting compressed bit vectors from different quadrants of the bitmap, we increase the likelihood that, under  $\text{vlc-}*$ , the queried bit vectors would be in segmentation lengths. This protocol also ensures that not all the queried columns were selected from the first (or last) few bit vectors which would favor WAH, or conversely, BBC in inner regions.

Figures 4(a) and 4(b) display the query time ratios between  $\text{vlc-}*$  versions and BBC and WAH respectively. We did not include the results from  $\text{vlc-opt}$  in the graphs as it was, on average, 10 times slower than the next slowest algorithm. This was expected, since many of the column pairs used in the queries had  $\text{gcd} = 1$ , meaning they had to be fully decompressed before logical operation could be applied.

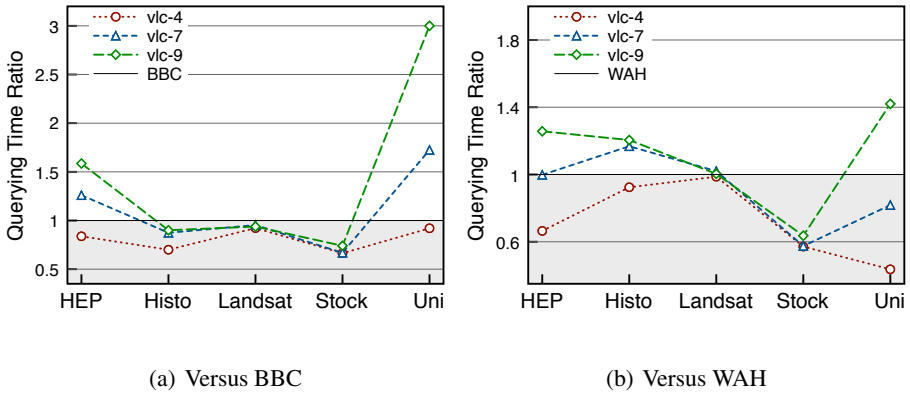


Fig. 4. Comparison of Query Processing

As can be seen in both Figure 4(a) and 4(b), *vlc-4* provides a less than optimal querying efficiency. Recall that this configuration can produce segment lengths of 4, 8, 12, 16, 20, 24, and 28. While this flexibility sometimes allows for robust compression rates, in terms of query performance, if  $X$  and  $Y$  vectors vary in segment lengths (which occurs frequently due to the large number of available base-4 options), they must be decoded to base-4 *gcd*, which is costly. This observation is supported by the performance gains from using the longer segment lengths of *vlc-7* and *vlc-9*. Because  $seg\_len = 7$  and  $seg\_len = 9$  only generate  $\{7, 14, 28\}$  and  $\{9, 18, 27\}$  bases, there will be higher probabilities where any two random bit vectors  $X$  and  $Y$  will match base, and thus *not* requiring *gcd*-base decoding.

Furthermore, for vectors that still require base decoding, the inherently larger bases of 7 and 9 allow bitwise operations to be amortized over a smaller base 4. These features combined with the increase in compression allowed *vlc-7* to match or improve upon WAH's querying times for 3 of the 5 data sets and *vlc-9* out-performs WAH on 4 of the data sets (Figure 4(b)). When compared to BBC, *vlc-7* and *vlc-9* had comparable query times for *Histo* and *Landsat*, and they performed significantly better on *Hep* and *Uni* (Figure 4(a)). For these latter data sets, *vlc-7* and *vlc-9* resulted in better compression than BBC and used larger segmentation lengths for most columns. The result is that much less parsing was required during query processing.

It is interesting to note that BBC querying actually out-performed WAH for *Histo* and *Landsat*. A closer investigation revealed that, for each of these data sets, a small number of the 10 queries were skewing the results. In both cases, the bit vectors in these outlier queries came from quadrants  $B_3$  and  $B_4$  of the bitmap. We surmise that these are the columns in a Gray code ordered bitmap that led to inefficient WAH compression. In these instances, it appears that WAH had to use a large number of literals to represent bit vectors. Essentially, this meant that WAH had to perform a logical operation for each bit. Conversely, due to BBC's ability to compress short runs it was able to perform fewer operations.

**Summary and Discussion:** In summary of our analysis, the results of our empirical study indicate that VLC can, on average, offer higher compression rates than either BBC and WAH. Although not initially expected, we also presented cases where VLC outperforms both BBC and WAH. Ignoring *vlc-opt* due to its prohibitive query



processing times, in the *best case* for real data sets, `v1c-9` out-compresses BBC and WAH by a factor of  $1.3\times$  and  $1.71\times$  respectively. These numbers jump to  $2.3\times$  and  $2.54\times$  respectively for BBC and WAH for synthetic uniform data. In all of our compression experiments, the *worst case* was a 15% loss in compression, albeit this was rare. In terms of query performance, we managed a speedup factor of  $1.6\times$  over BBC and  $1.25\times$  over WAH in the *best case* for real data sets. These numbers grow to a  $3\times$  speedup for BBC and  $1.42\times$  speedup for WAH for our synthetic data. In the *worst case*, around a  $0.6\times$  slowdown for `Stock` can be seen compared to either BBC or WAH. This suggests that VLC's *gcd* decoding renders it inefficient for high dimensional data sets with small numbers of rows, which are rare for large-scale data-intensive applications.

Our results also echo such efforts as [12], which sought a deeper understanding of how row ordering (as well as potentially many other optimizations) truly affects bitmap performance and compression. We view our findings as yet another example of how careful consideration of the segmentation length used for compression is especially important for Gray code ordered bitmaps.

## 5 Related Work

There is a large body of research related to bitmap indices and compression. In this section we focus on the class of techniques most relevant to our work.

Bitmap indices [18, 15], which are closely related to inverted files [14] (frequently occurring in information retrieval) have long been employed in traditional OLAP and IR systems. Over the years, seminal efforts have made practical the integration of bitmaps for general data management. For instance, works have addressed bitmap encoding issues, which include considerations on bit-vector cardinality and representation to optimize query processing [5, 17]. The focus in efforts are tangential to the work presented in this paper. We offer a generalized word aligned compression scheme.

*Run-length encoding* (RLE) techniques [9] were popularized early through the ubiquitous bit representations of data, and were particularly useful for compressing sparse bitmaps. For example, a bit vector can be transformed to a vector containing only the positions of 1's, and thereby implicitly compressing the 0 bits. When the sequence is further transformed to the differences of the positions, coding schemes, such as Elias's  $\delta$  and  $\gamma$  codes and its variations [8, 13, 4], can be used to map the expectedly small integers to correspondingly small bit representations.

These efforts, however, do not consider the impact of memory alignment, which significantly slows the performance of bitwise logical operations, e.g., a vector which spans two bytes would require two separate reads. Such decoding overheads would not be very suitable for database query processing. Thus, effort towards generating memory-aligned, CPU friendly compressed bitmaps ensued. The Byte-aligned Bitmap Code (BBC) [2] exploits byte alignment and allows direct bit-wise comparisons of vectors in their compressed state. Wu, *et al.* proposed Word-Aligned Hybrid Code (WAH) [20, 19], which is even more amenable for processing.

Due to WAH's success in accelerating query processing times, many variations of WAH have also been proposed. One example is the Word-aligned Bitmap Code (WBC), also seen in literature as Enhanced WAH (EWAH) [21]. This scheme uses a *header* word to represent fill runs and literal runs. A 31-bit header word (plus the 1 MSB for flagging) is split into two halves. The upper 16 bits following the flag bit is used to denote the fill, and the run length, just as before. The lower 15 bits are used to denote the run length of literals following the fill run. This optimization can be significant

for query processing. For instance, long literal runs can be ignored (without accessing them) when *AND*'d with 0's. If a logical comparison is indeed required, the decoding phase would know *a priori* that the next  $n$  words are literals without parsing the flag bit.

Deliege and Pederson proposed a Position List extension to WAH (PLWAH), which exploits highly similar words [6]. Their insight is that, often, only a single bit can compromise a much longer run, and typically, it is highly unlikely that all fill bits are used in a word. Thus, their scheme first separates a bitmap into segments of  $wordsize - 1$  bits. Words that are “nearly identical” to a fill word are identified and appended onto a fill, rather than representing it as a literal. The five most significant bits following the fill word's *fill* and *flag* bits are further used to identify the position of the bits in the nearly identical literal. We surmise that PLWAH can be used in conjunction to our VLC scheme, which is currently being implemented.

There has also been significant amounts of work on the issue of *bitmap row reordering*. Because tuple order is arbitrary in a database relation, its corresponding bitmap can thus be reordered to maximize runs. Pinar, Tao, and Ferhatosmanoglu explored the tuple reordering problem in [16]. They proved that tuple reordering is NP-Complete, and proposed the Gray code ordering heuristic. More recently, Lemire *et al.* presented an *extensive* investigation into reordering efforts (including column reordering, which is not being considered in our work) over large bitmaps [12]. Among various contributions, they made several interesting observations. For one, the authors experimented with 64-bit words and observed an interesting space-time tradeoff: while 64-bit word compression expectedly generates indices that are twice as large, the queries are slightly faster.

The above efforts are orthogonal to VLC — our technique allows bit vectors to be compressed and queried using varying segment encoding lengths. We have shown that VLC achieves greater compression in our experiments than both WAH and BBC in most cases, when the correct segment length is chosen. Our scheme thus provides an option to the user to encode a bitmap using specific encoding lengths to greater optimize compression, or to use encoding lengths that would allow for faster querying on certain columns that may be queried often. Thus, VLC is a *tunable* approach, which allows users to trade-off size and performance.

## 6 Conclusion and Future Work

Bitmap indices have become a mainstay in many database systems, popularized due to its fast query processing and compressibility. However, as high dimensional data continues to grow at today's astounding pace, bitmap compression becomes increasingly more important to minimize disk accesses when possible.

In this paper, we proposed a novel bitmap compression technique, *Variable Length Compression (VLC)*, which allows for robust run-length compression of ordered bitmaps. We offer two simple heuristics on selecting encoding length per given bit vector. Our query processing algorithm automatically decodes bit vectors to the same coding base so that queries can be carried out efficiently. We ran an experimental study on 4 sets of real data and 1 synthetic data set. We showed that VLC can out compress both BBC and WAH, two of today's state-of-the-art bitmap compression schemes, by around  $2.5\times$  in the best case. We concede cases where VLC compression rates are less than BBC, but only marginally. In terms of query performance, the expectation was that VLC would lie somewhere between BBC and WAH, but remain competitive to WAH.

While this assumption was shown to be true, we also observed interesting results that show clearly, there are cases where VLC outperforms and out compresses both BBC and WAH. Again, we acquiesce that there are opportunities for improvement on certain data sets.

During the evaluation process, some future work opportunities emerged. For example, we can (and should) *adapt* the segment encoding lengths to query history. Because longer segments tend to query much faster, we can dynamically relax compression rates for frequently queried columns (and conversely, compress infrequently queried columns more aggressively). To ensure word alignment, we currently *pad in* the last unused bits of a word, a necessary storage cost. An alternative could be to fit as many representations fit into one word, then begin a representation in one word and finish it in the following word, “stitching” the representation together. While we expect that the stitching may result in a slowdown in query times, but may also provide a substantial gain in compression. Some obvious experimental extensions would include using larger data sets, different row ordering algorithms combined with column re-ordering which has been shown to increase run-lengths [11, 12], and range queries.

**Acknowledgments.** We would like to thank Hakan Ferhatosmanoglu (Ohio State University), Guadalupe Canahuete (The University of Iowa), and Michael Gibas (Teradata) for their valuable insights and comments on this paper.

This work was generously supported in part by an Amazon Web Services (AWS in Education) Research Award to D. Chiu at Washington State University and a Martin Nelson Summer Award for Research to J. Sawin at the University of Puget Sound.

## References

1. Abadi, D., Madden, S., Ferreira, M.: Integrating compression and execution in column-oriented database systems. In: ACM SIGMOD International Conference on Management of Data, pp. 671–682 (2006)
2. Antoshenkov, G.: Byte-aligned bitmap compression. In: DCC 1995: Proceedings of the Conference on Data Compression, p. 476. IEEE Computer Society, USA (1995)
3. Apaydin, T., Tosun, A.Ş., Ferhatosmanoglu, H.: Analysis of basic data reordering techniques. In: Ludäscher, B., Mamoulis, N. (eds.) SSDBM 2008. LNCS, vol. 5069, pp. 517–524. Springer, Heidelberg (2008)
4. Brisaboa, N.R., Ladra, S., Navarro, G.: Directly addressable variable-length codes. In: Karlgren, J., Tarhio, J., Hyvrö, H. (eds.) SPIRE 2009. LNCS, vol. 5721, pp. 122–130. Springer, Heidelberg (2009)
5. Chan, C.-Y., Ioannidis, Y.E.: An efficient bitmap encoding scheme for selection queries. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of data SIGMOD 1999, pp. 215–226. ACM, New York (1999)
6. Deliege, F., Pederson, T.: Position list word aligned hybrid: Optimizing space and performance for compressed bitmaps. In: Proceedings of the 2010 International Conference on Extending Database Technology (EDBT 2010), pp. 228–239 (2010)
7. Donno, F., Litmaath, M.: Data management in wlcg and egee. worldwide lhc computing grid. Technical Report CERN-IT-Note-2008-002, CERN, Geneva (February 2008)
8. Elias, P.: Universal codeword sets and representations of the integers. IEEE Transactions on Information Theory, 21(2), 194–203 (1975)
9. Golomb, S.W.: Run-Length Encodings. IEEE Transactions on Information Theory 12(3), 399–401 (1966)

10. Kaser, O., Lemire, D., Aouiche, K.: Histogram-aware sorting for enhanced word-aligned compression in bitmap indexes. In: ACM 11th International Workshop on Data Warehousing and OLAP, pp. 1–8 (2008)
11. Lemire, D., Kaser, O.: Reordering columns for smaller indexes. *Information Sciences* 181 (2011)
12. Lemire, D., Kaser, O., Aouiche, K.: Sorting improves word-aligned bitmap indexes. *Data and Knowledge Engineering* 69, 3–28 (2010)
13. Moffat, A., Zobel, J.: Parameterised compression for sparse bitmaps. In: SIGIR, pp. 274–285 (1992)
14. Moffat, A., Zobel, J.: Self-indexing inverted files for fast text retrieval. *ACM Transactions on Information Systems* 14, 349–379 (1996)
15. O’Neil, P.E.: Model 204 architecture and performance. In: Gawlick, D., Reuter, A., Haynie, M. (eds.) HPTS. LNCS, vol. 359, pp. 40–59. Springer, Heidelberg (1989)
16. Pinar, A., Tao, T., Ferhatosmanoglu, H.: Compressing bitmap indices by data reorganization. In: Proceedings of the 2005 International Conference on Data Engineering (ICDE 2005), pp. 310–321 (2005)
17. Sinha, R.R., Winslett, M.: Multi-resolution bitmap indexes for scientific data. *ACM Trans. Database Syst* 32 ( August 2007)
18. Wong, H.K.T., Fen Liu, H., Olken, F., Rotem, D., Wong, L.: Bit transposed files. In: Proceedings of VLDB 1985, pp. 448–457 (1985)
19. Wu, K., Otoo, E., Shoshani, A.: An efficient compression scheme for bitmap indices. *ACM Transactions on Database Systems* (2004)
20. Wu, K., Otoo, E.J., Shoshani, A.: Compressing bitmap indexes for faster search operations. In: Proceedings of the 2002 International Conference on Scientific and Statistical Database Management Conference (SSDBM 2002), pp. 99–108 (2002)
21. Wu, K., Otoo, E.J., Shoshani, A., Nordberg, H.: Notes on design and implementation of compressed bit vectors. Technical Report LBNL/PUB-3161, Lawrence Berkeley National Laboratory (2001)
22. Zaki, M.J., Wang, J.T.L.: Special issue on bioinformatics and biological data management. *Information Systems* 28, 241–367 (2003)

# Modeling View Selection as a Constraint Satisfaction Problem

Imene Mami, Remi Coletta, and Zohra Bellahsene

LIRMM, University Montpellier 2  
161 Rue Ada  
F-34095 Montpellier, France  
`{imen.mami,coletta,bella}@lirmm.fr`

**Abstract.** Using materialized views can highly speed up the query processing time. This paper deals with the view selection issue, which consists in finding a set of views to materialize that minimizes the expected cost of evaluating the query workload, given a limited amount of resource such as total view maintenance cost and/or storage space. However, the solution space is huge since it entails a large number of possible combinations of views. For this matter, we have designed a solution involving constraint programming, which has proven to be a powerful approach for modeling and solving combinatorial problems. The efficiency of our method is evaluated using workloads consisting of queries over the schema of the TPC-H benchmark. We show experimentally that our approach provides an improvement in the solution quality (i.e., the quality of the obtained set of materialized views) in term of cost saving compared to genetic algorithm in limited time. Furthermore, our approach scales well with the query workload size.

## 1 Introduction

The information stored at the warehouse is often organized in materialized views which represent pre-computed portions of the most frequently asked queries [3]. Using materialized views can improve the performance and speed up the processing of queries since the access to materialized views can be much faster than recomputing the views. However, these materialized views have to be maintained in response to changes to the underlying base relations. In most cases it is wasteful to maintain a view by re-computing it from scratch. Often, it is cheaper to compute only the changes in the view to update its materialization which is called the incremental view maintenance.

Materializing all the input queries can achieve the lowest query cost but the highest view maintenance cost which can cause overhead to the system. Besides, the query result can be too large to fit in the available storage space. Hence, there is a need for selecting a set of views to materialize by taking into account three important features: query cost, view maintenance cost and storage space.

The problem of choosing which views to materialize that minimize the total query cost given a limited amount of resource such as total view maintenance

cost and/or storage space is known as the view selection problem. This is one of the most challenging problems in data warehousing [28]. The view selection problem is a NP-complete problem since the search space for the optimal solution grows exponentially as the problem size increases [11,12].

In this paper, we propose a new approach to the view selection problem which minimizes the total query cost under the case where (i) the limited resource is the total view maintenance cost, assuming unlimited amount of storage space if we consider that storage space is cheap and not regarded as a critical resource anymore and (ii) both space and maintenance cost constraints exist.

Although, several heuristic algorithms have been proposed in literature to solve the view selection problem such as deterministic algorithms i.e., greedy algorithms [11,29,12,25,20,21,26,2], randomized algorithms i.e., genetic algorithms [31,18,13,30,16,5] and simulated annealing algorithms [14,7,8] or hybrid algorithms [32] which combine the strategies of pure deterministic algorithms and pure randomized algorithms.

These heuristic algorithms provide reasonably good solutions. However, there is no guarantee of performance because the greedy nature or the random characteristic of the algorithms may make them converging to poor local minima. An exact resolution for the view selection problem is prohibited since an exhaustive search cannot compute an optimal solution within a reasonable time due to the complexity of the problem.

Yet, over the past ten years, effective paradigms for exact resolution of NP-complete problems have been proposed, such as constraint programming (CP), structured around annual competitions [17]. Furthermore, constraint programming has proven to be a powerful technique for modeling and solving combinatorial problems [9]. We have designed a new approach for solving the view selection problem involving constraint programming. Our approach consists in modeling the view selection as a Constraint Satisfaction Problem (CSP). Our main contributions are:

1. We propose a new approach to the view selection problem. We model this problem as a constraint satisfaction problem (CSP). Then, a constraint programming (CP) solver can be applied to set up the search space by identifying a set of views that minimizes the total query cost.
2. We address the view selection under the case where (i) the limited resource is the total view maintenance cost, assuming unlimited amount of storage space if we consider that storage space is cheap and not regarded as a critical resource anymore and (ii) both space and maintenance cost constraints exist.
3. We highlight the *anytime* behavior of our approach which is able to provide a near optimal solution to the view selection problem during a given time interval. The quality of this solution may be improved over time (if the CPU time is available).
4. We have implemented our approach and compared it with a randomized method (i.e., genetic algorithm). We experimentally show that our approach provides better performance resulting from evaluating the quality of the solutions in term of cost savings and scales well with the query workload.

The rest of this paper is organized as follows. Section 2 describes the problem of view selection and the framework used for representing the queries of the workload. Section 3 provides an overview of our approach and describes how to model the view selection problem as a constraint satisfaction problem (CSP). In section 4, are provided the experiments results. Section 5 contains a brief survey of related work. Finally, in section 6 we conclude and plan for future work.

## 2 Problem Specification

The general problem of view selection is to select a set of views to be materialized that minimizes the cost of evaluating the query workload modeled by the frequently asked queries, given a limited amount of resource, e.g., total view maintenance cost and/or storage space. In this paper, we consider selection-projection-join (SPJ) queries that may involve aggregation and group by clause as well.

In order to detect overlapping between queries of workload and capture the dependencies among the queries, the view selection is represented as an AND-OR view graph [11]. The union of all possible execution plan of each query forms an AND-OR view graph where the common sub-expression are represented once. The AND-OR view graph is a Directed Acyclic Graph (DAG) composed of two types of nodes: Operation nodes (Op-nodes) and Equivalence nodes (Eq-nodes). Each Op-node represents an algebraic expression (Select-Project-Join) with possible aggregate function. An OR-node represents a set of logical expressions that are equivalent (i.e., that yield the same result). The Op-nodes have only Eq-nodes as children and Eq-nodes have only Op-nodes as children. The root nodes are equivalence nodes representing the queries of workload and the leaf nodes represent the base relations. Equivalence nodes in the AND-OR view graph correspond to the views that are candidates for materialization.

A sample AND-OR view graph is shown in figure 1. Circles represent operation nodes and boxes represent equivalence nodes. For example, in this figure, view  $v_1$ , corresponding to a single query, can be computed from  $v_3$  and  $r_2$  or  $v_4$  and  $r_3$ . If there is only one way to answer or update a given query, the graph becomes an AND view graph. We explore the view selection problem in the context of AND-OR view graph, which allows a single query to be answered and updated from multiple paths, since a good selection of materialized views can only be found by considering the optimization of both global processing plans and materialized view selection [32].

To each equivalence node which represents a view, is associated the following metadata:

- Query cost  $Qc$  which represents the cost of computing a view from its related base relations and/or views.
- Maintenance cost  $Mc$  which is the cost required for updating a view when the related base relation is changed.

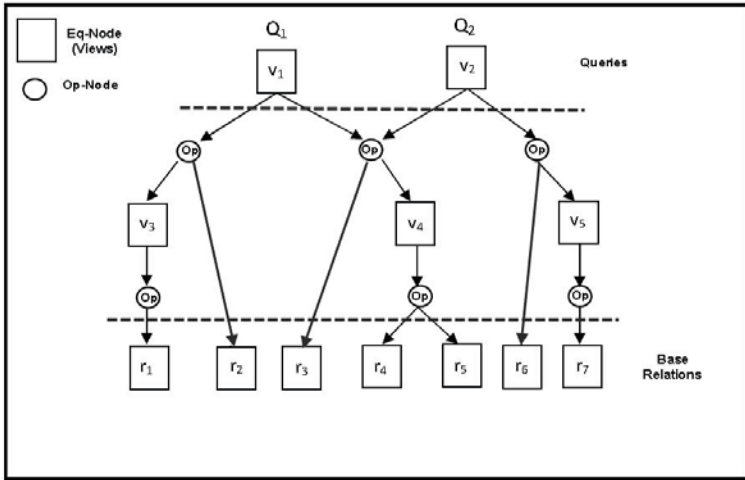


Fig. 1. The AND-OR view graph of the two queries  $Q_1$  and  $Q_2$

- Read cost  $Rc$  that denotes the size of the view.
- Query frequency  $f_q$  (if the equivalence node is a root node) which describes the frequency of posing a given query.
- Update frequency  $f_u$  which represents the frequency of updating a view in response to change to the underlying data.

The general view selection problem for AND-OR view graphs can be formulated as follows: Given an AND-OR view graph  $G$ , select the set of views to materialize that minimizes the total query cost under the following cases:

- Where only the maintenance cost constraint is considered. Here, we address the view selection problem by constraining the total maintenance cost needed to update the materialized views while assuming unlimited amount of storage space.
- Where both maintenance cost and space constraints exist. In this case, we have a total maintenance cost limit but also a bound on the total storage space required to materialize the selected views.

### 3 A New Approach to the View Selection Problem

#### 3.1 Motivations

In this section, we present our approach for selecting a set of views to be materialized. The optimal solution is the one which selects the right materialized views (equivalence nodes) of the AND-OR view graph that minimizes the total query cost subject to certain constraints such as space and maintenance cost



constraints. However, the search space for the optimal solution is very large since it entails a great number of comparisons between all possible subsets of views.

Our motivation to use constraint programming in solving the view selection problem is that it is known to be a powerful approach for modeling and solving combinatorial problems such as Job Shop Scheduling [4]. The success of using constraint programming for combinatorial optimization is due to its combination of these three features [27]:

- *High level modeling.* Constraint programming provides a rich constraint language to model the problem as a Constraint Satisfaction Problem. In the following subsections, we give a formal definition of CSP and show how to model the view selection problem as a CSP.
- *Constraint propagation.* This leads to a reduction of the search space, by excluding solutions where the constraints become inconsistent. For example, in the case of solving the view selection problem under the maintenance cost constraint, all the view combinations which violate this constraint are discarded.
- *Search.* Constraint programming offers facilities to control the search behavior deciding which alternative (i.e., views combination) to try first.

### 3.2 Preliminaries

In this subsection, we introduce the CSP model that we have used in our approach. A constraint satisfaction problem (CSP) is composed of:

- A set of variables  $VAR = \{var_1, var_2, \dots, var_n\}$
- Each variable  $var_i$  has a set of values which is called the domain of values  $DOM = \{d_1, d_2, \dots, d_n\}$
- A set of constraints  $CST = \{c_1, c_2, \dots, c_n\}$  describes the relationship between subsets of variables. Formally, a constraint  $C_{ijk}$  between the variables  $var_i, var_j, var_k$  is any subset of the possible combinations of values of  $var_i, var_j, var_k$ , i.e.,  $C_{ijk} \subset d_i \times d_j \times d_k$ . The subset specifies the combinations of values that the constraint allows.

A CSP consists in finding solutions by assigning values to its variables that satisfy all its constraints. Our approach consists in modeling the view selection problem as a CSP. Its resolution is supported automatically by constraint solver embedded in the constraint programming language.

### 3.3 Modeling View Selection Problem as a CSP

Formulating the view selection problem as a constraint satisfaction problem (CSP) consists in specifying the variables of the CSP, their domains, and the constraints that are over them in the context of view selection. In the following, we describe each part of the specification.

### 3.3.1 Variables and Domains

The variables of the CSP considered in modeling the view selection problem are:

- $Mat_{v_i}$  which denotes for each view  $v_i$  (equivalence node in the AND-OR view graph) , if it is materialized or not materialized. It is a binary variable,  $d_{Mat_{v_i}} = 0,1$  (0:  $v_i$  is not materialized, 1:  $v_i$  is materialized).
- $Qc(v_i)$  that represents the query cost corresponding to a view  $v_i$ . The domain is a finite subset of  $\mathbb{N}^*$  such as  $d_{Qc(v_i)} \subset \mathbb{N}^*$ .
- $Mc(v_i)$  which is the maintenance cost corresponding to a view  $v_i$ , where  $d_{Qc(v_i)} \subset \mathbb{N}^*$ .

### 3.3.2 Constraints

The constraints which describe the relationship between the variables defined above are:

- The query and maintenance cost corresponding to a view is implemented by using a depth-first traversal of the AND-OR view graph. We use the cost formulae described in [24,21] to compute these two costs (defined in a recursive way).

#### Query Cost:

$$Qc(v_i) = \begin{cases} ComputingCost(v_i) & \text{if } Mat_{v_i} = 0 \\ \min(ComputingCost(v_i), ReadingCost(v_i)) & \text{otherwise} \end{cases}$$

$$ComputingCost(v_i) = \min_{op_j \in child(v_i)} \left( cost(op_j) + \sum_{v_k \in child(op_j)} Qc(v_k) \right)$$

The query cost corresponding to view  $v_i$  which is an equivalence node in the AND-OR view graph, is the minimum cost paths from  $v_i$  to its related views or base relations, if  $v_i$  is not materialized. Otherwise, if  $v_i$  is materialized, we use the minimum of the cost of reading  $v_i$  and the minimum cost paths as defined above.

Each minimum cost path is composed of all the cost of executing the operation nodes on the path and the query cost corresponding to the related views or bases relations of  $v_i$ . The costs of executing the operations: selection, join, projection and aggregation are estimated according to the formulas given in [6] for cost operation estimation. In this paper, these costs are calculated in terms of number of tuples in the involved relations.

#### View maintenance cost:

$$Mc(v_i) = \begin{cases} 0 & \text{if } Mat_{v_i} = 0 \\ \sum_{dr_l \in diffRelations(v_i)} Mcost(v_i, dr_l) & \text{otherwise} \end{cases}$$

$$Mcost(v_i, dr_l) = \min_{op_j \in child(v_i)} \left( cost(op_j, dr_l) + \sum_{v_k \in child(op_j)} UpdatingCost(v_k, dr_l) \right)$$

$$UpdatingCost(v_k, dr_l) = \begin{cases} Mcost(v_k, dr_l) & \text{if } Mat_{v_k} = 0 \\ \min(Mcost(v_k, dr_l), \delta(v_k, dr_l)) & \text{otherwise} \end{cases}$$

The maintenance cost of view  $v_i$ , if it is materialized, is computed by summing the number of changes in the base relations from which  $v_i$  is updated. If  $v_i$  is not materialized, then there is no maintenance cost. We assume incremental maintenance to estimate the view maintenance cost. Therefore, the maintenance cost is the differential results of materialized views given the differential (updates) of the bases relations. Let  $\delta(v_i, dr_l)$  denotes the differential result of view  $v_i$ , with respect to update  $dr_l$ .

The view maintenance cost is computed similarly to the query cost, but the cost of each minimum path is composed of all the cost of executing the operation nodes with respect to update  $dr_l$  on the path and the maintenance cost corresponding to the related views of  $v_i$ . We have been inspired by the formula given in [21] for estimating the cost of executing the operation nodes in response to changes to the base relations.

- The total maintenance cost of the set of materialized views is less than  $U$  which is the total view maintenance cost limit.

$$\sum_{v_i \in V(G)} (Mat_{v_i} * (f_u(v_i) * Mc(v_i))) \leq U$$

Note that  $V(G)$  represents all the views in the AND-OR view graph,  $Mc(v_i)$  is the cost of maintaining a materialized view  $v_i$  and  $f_u(v_i)$  is the update frequency of the view  $v_i$ . Here, the view selection is decided under the maintenance cost constraint.

- The total space occupied by the materialized views  $M$  is less than  $S$  which is the maximum storage space.

$$\sum_{v_i \in V(G)} (Mat_{v_i} * Rc(v_i)) \leq S$$

Recall that  $Rc(v_i)$  is the size of the view  $v_i$ . If the space constraint is considered, views are selected to be stored only if the necessary space for their materialization is at most  $S$ .

- Minimize the total query cost.

$$\text{minimize} \left( \sum_{v_i \in Q(G)} (f_q(v_i) * Qc(v_i)) \right)$$

In this formula,  $Q(G)$  represents all the queries (root nodes in the AND-OR view graph),  $Qc(v_i)$  is the query cost of the view  $v_i$  and  $f_q(v_i)$  is the query frequency of the view  $v_i$ . The total query cost is computed by summing over the cost of processing all the queries of workload.

## 4 Experimental Evaluation

We have implemented our approach and compared it with a randomized method because in previous studies [30,7,8], it was shown that randomized algorithms provide a significant improvement in the performance compared to deterministic algorithms. The most commonly used randomized algorithms in the context of view selection are simulated annealing algorithms and genetic algorithms. The motivation to compare our approach with genetic algorithm is based on the observation that genetic algorithm in contrast with simulated annealing algorithm use a multi-directional search which allows to find a point near the global optimum [18]. A comparison with hybrid approaches has not been made because of their excessive computation time [32]. The goal of the experiments is the comparison of the solution quality resulting from evaluating the quality of the obtained set of materialized views in term of cost savings between our method and genetic algorithm.

### 4.1 Experiment Settings

The computer used for experimentation was an Intel Core 2 Duo P8600 CPU @ 2.40 GHz machine running with 3GB of RAM and Windows XP Professional SP3. The program was written in Java using JDK/JRE 1.6.0. We chose a workload of one hundred queries defined over the database schema of the TPC-H benchmark [1]. We then randomly assigned values to the frequencies for access and update based on uniform distribution.

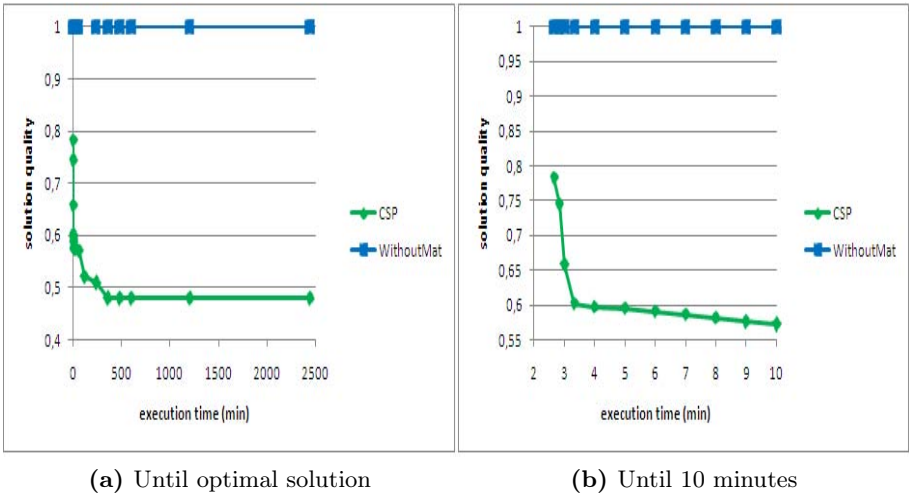
In all experiments, the quality of the solutions found by the genetic algorithm and our method was respectively measured as a ratio of the total query cost obtained using the genetic algorithm and the constraint solver over the total query cost when all the views are not materialized. Thus, we consider the "Without-Mat" approach which does not materialize views and always recomputes queries as a benchmark for our normalized results. The ratio was computed and averaged over several runs for the genetic algorithm because of his probabilistic behavior.

We have implemented the genetic algorithm presented in [5] by incorporating space and maintenance cost constraints into the algorithm. The values for population size and probabilities of the crossover and mutation operators are assigned based on studies conducted by [10,19]. In order to let the genetic algorithm converge quickly, we generated an initial population which represents

a favorable view configuration rather than a random sampling. Favorable view configuration such as the views which satisfy the maintenance cost and space constraints if they exist are most likely selected for materialization.

## 4.2 Experiment Results

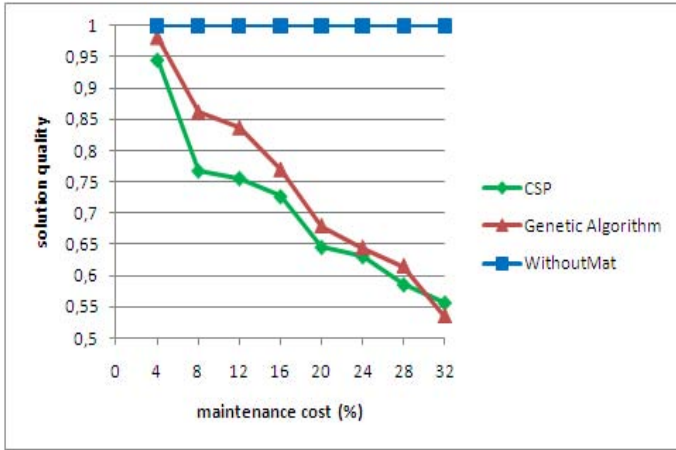
The constraint solver for solving the view selection problem as a constraint satisfaction problem (CSP) is aimed to provide the optimal solution (see figure 2(a)). However, this is not feasible at large scale because of the great number of comparisons between all possible subsets of views which are candidate for materialization. One way to avoid an explosion in the search space is to explore a strictly limited number of possibilities. In this case, the constraint solver is aimed at simply finding a feasible solution in which all constraints are satisfied. Figure 2(b) shows the quality of the solutions returned by our approach involving 30 queries as a function of execution time. We can see, our approach provides near optimal solutions quickly. Indeed, after only few minutes (i.e., 3 minutes and 24 seconds), the solution found lies within 79,84% of the optimal solution. The quality of this solution is improved over time. In the next experiments, the constraint solver was left to run until the convergence of the genetic algorithm.



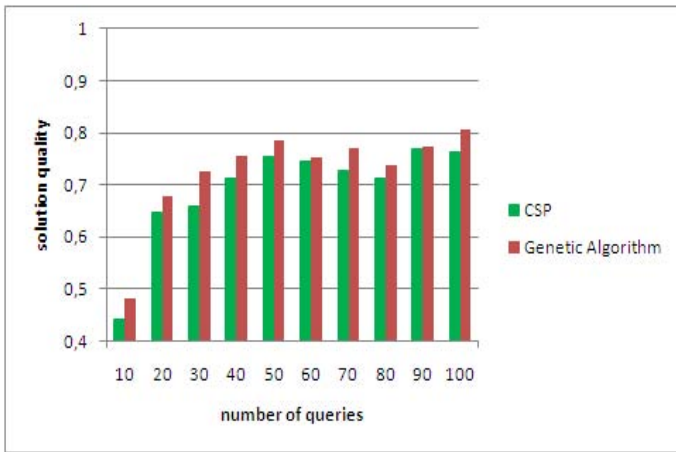
**Fig. 2.** The solution quality of our approach over time (nb of queries=30)

### 4.2.1 Experiments under the Maintenance Cost Constraint

In this section, we compare the performance resulting from evaluating the quality of the solutions found by our approach and genetic algorithm in the context where the view selection is constrained by a total view maintenance cost limit  $U$ . In these experiments,  $U$  is computed as a function of  $U(Q)$  which is the total maintenance cost when all queries (i.e., root nodes in the AND-OR view graph) are materialized [14]. We set  $U$  to be 4%, 8%, 16%,..., 32% of  $U(Q)$ .



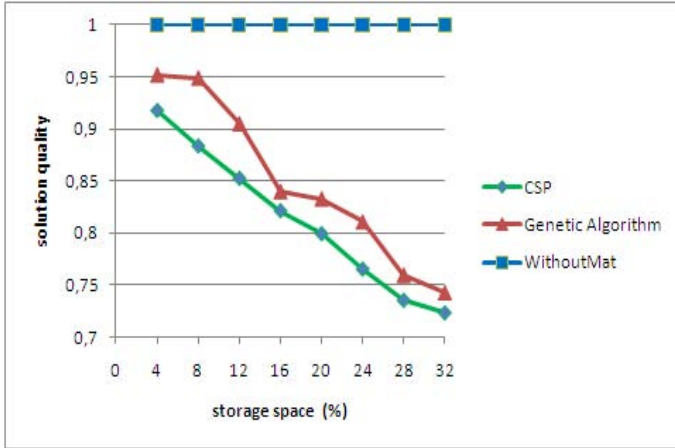
**Fig. 3.** Comparison of the view selection methods while varying the maintenance cost constraint (nb of queries=70)



**Fig. 4.** Evaluating the performance while varying the number of queries ( $U=16\%$  of  $U(Q)$ )

Figure 3 compares experiment results of our approach with that of genetic algorithm while varying the values of  $U$  for a workload involving 70 queries. We can see in figure 3 that our approach generates solutions with cost less than that returned by the genetic algorithm when  $U < 32\%$  of  $U(Q)$ . For large values of  $U$  ( $U \geq 32\%$  of  $U(Q)$ ), solutions returned by the genetic algorithm have better quality compared with our approach. This is because the genetic algorithm converges faster when we relax the maintenance cost constraint and our approach needs more time to achieve better quality. However, this would not be seen as a drawback since the time bound for the update is usually very tight compared

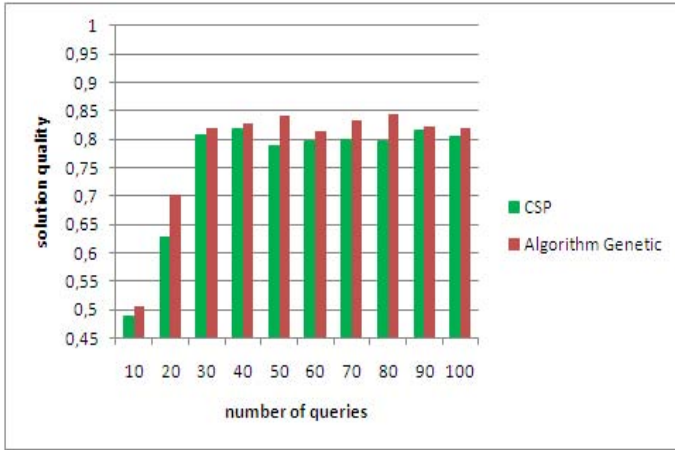
with the time required for maintaining all the query workload. Figure 4 shows the costs of the workload involving 10,20,30,40,50,60,70,80,90 and 100 queries in order to test the scalability of the view selection methods according to the number of queries. The maintenance cost constraint  $U$  was set to 16% of  $U(Q)$ . This graphics shows that our approach provides an improvement in the quality of the obtained set of materialized views in term of cost savings compared with the genetic algorithm. Furthermore, our method supports scalability when the number of queries increases.



**Fig. 5.** The solution quality as a function of available storage space (number of queries=70,  $U=50\%$  of  $U(Q)$ ).

#### 4.2.2 Experiments under the Maintenance Cost and Space Constraints

In order to compare the performance of the view selection methods in the context where both maintenance cost and space constraints exist, we set  $U$  to 50% and give restrictive values to the space constraint  $S$ . In these experiments,  $S$  is computed as a function of  $S(Q)$  which is the size of the whole workload [14]. We varied  $S$  between 4% and 32% of  $S(Q)$ . Figure 5 illustrates the quality of the solutions produced by the two methods for various values of  $S$  for a workload involving 70 queries. The graphic shows that our approach generates better solutions than the genetic algorithm in the case where the storage space is the restrictive constraint. Experiment results depicted in figure 6 shows how well the view selection methods scale with the problem size. The maintenance cost constraint was set to 50% of  $U(Q)$  and the space constraint to 20% of  $S(Q)$ . The size of the problem varied from 10 to 100 queries. We observe that our approach provides the lowest query cost while varying the number of queries. Therefore, our approach outperforms the genetic algorithm for all query workload size. We also show that our approach can be applied to large number of queries.



**Fig. 6.** The quality of results as a function of the number of queries ( $U=50\%$  of  $U(Q)$ ,  $S=20\%$  of  $S(Q)$ ).

## 5 Related Work

As mentioned in the introduction, the view selection problem is a NP-complete problem. Several view selection methods have been proposed in the literature to address the view selection problem. They can be classified into three major groups:

### 5.1 Deterministic Algorithms Based Methods

Such methods usually provide a solution to the view selection problem either by applying exhaustive search or by applying heuristics i.e., greedy algorithms to reduce the search space. In [23], an exhaustive approach is presented for finding the best set of views to materialize. [15] presents an optimal algorithm based on A\* algorithm [22] that vastly prune the search space compared to the algorithm proposed in [23]. However, an exhaustive search cannot compute an optimal solution within a reasonable time due to the complexity of the problem. Many approaches [11,29,12,25,20,21,26,2] using a kind of greedy strategy to avoid having to traverse the solution space in an exhaustive search manner have been designed. However, greedy algorithms are unsatisfactory in term of the solution quality i.e., the quality of the obtained set of materialized views because the greedy nature of the algorithm makes it susceptible to poor local minima (initial solutions influence the solution greatly). In contrast with these work, our approach provides a suitable trade-off between the computation time and the solution quality. Indeed, we have shown that our approach provides a good solution quality in a limited time. This is due to the use of constraint propagation technique and facilities for controlling search behavior that are the features of constraint programming.



## 5.2 Randomized Algorithms Based Methods

Randomize algorithms such as simulated annealing algorithms [7,8,14] and genetic algorithms [5,13,18,30,16] have been used and explored for the selection of materialized views in order to improve the quality of the solution. These algorithms use a randomized search strategy for deciding which views to materialize. Randomized algorithms provide a better solution quality than greedy algorithms. However, they may have a tendency to converge towards local optima. Besides, their successes often depend on the set-up of the algorithm as well as the extremely difficult fine-tuning of algorithm that must be performed during many test runs. In our approach, we simply model view selection as a constraint satisfaction problem (CSP) and its resolution is supported automatically by constraint solver embedded in the constraint programming language. Besides, our approach provides better results compared with genetic algorithm in term of the solution quality.

## 5.3 Hybrid Algorithms Based Methods

Hybrid algorithms combine the advantages of pure deterministic algorithms and pure randomized algorithms. A hybrid approach has been applied in [32] for the view selection problem which combine heuristic algorithms i.e., greedy algorithms and genetic algorithms. They prove that hybrid algorithms provide better performance than either the genetic algorithms or heuristic algorithms used alone in terms of solution quality. However, they often require longer computation time and may be impractical due to their excessive computation time.

# 6 Conclusion

In this paper, we have presented a new approach to the view selection problem which is based on constraint programming. More specifically, the view selection problem has been modeled as a constraint satisfaction problem (CSP). Its resolution has been supported automatically by constraint solver embedded in the constraint programming language. We have performed several experiments and comparison with a randomized method i.e., genetic algorithm. The experiment results have shown that our approach provides better performances compared with the genetic algorithm in term of the solution quality (i.e., the quality of the obtained set of materialized views) in a limited time. More precisely, we have demonstrate experimentally that our approach provides better results compared with genetic algorithm in term of cost savings when the view selection is decided under the case where (i) only the maintenance cost constraint is considered, assuming unlimited amount of storage space and (ii) both maintenance cost and space constraints exists. We have also shown that our approach supports scalability when the number of queries increases. As a future work, we are planning to apply our approach to a distributed database setting. Our current approach simply takes into account the space and maintenance cost constraints. These

constraints will be per machine in a distributed context. Also, resource constraints such as CPU, IO, network bandwidth and the location of materialized views will have to be taken into consideration. These new constraints will easily be handled with our approach.

## References

1. TPC-R Benchmark Standard Specification 2.01 (January 1999), <http://www.tpc.org>
2. Baril, X., Bellahsene, Z.: Selection of materialized views: A cost-based approach. In: CAiSE, pp. 665–680 (2003)
3. Bello, R.G., Dias, K., Downing, A., Feenan Jr., J.J., Finnerty, J.L., Norcott, W.D., Sun, H., Witkowski, A., Ziauddin, M.: Materialized views in oracle. In: VLDB, pp. 659–664 (1998)
4. Caseau, Y., Laburthe, F.: Improved clp scheduling with task intervals. In: ICLP, pp. 369–383 (1994)
5. Chaves, L.W.F., Buchmann, E., Hueske, F., Böhm, K.: Towards materialized view selection for distributed databases. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology EDBT 2009, pp. 1088–1099. ACM, New York (2009)
6. Chirkova, R., Halevy, A.Y., Suciu, D.: A formal perspective on the view selection problem. VLDB J. 11(3), 216–237 (2002)
7. Derakhshan, R., Dehne, F.K.H.A., Korn, O., Stantic, B.: Simulated annealing for materialized view selection in data warehousing environment. In: Databases and Applications, pp. 89–94 (2006)
8. Derakhshan, R., Stantic, B., Korn, O., Dehne, F.: Parallel simulated annealing for materialized view selection in data warehousing environments. In: Bourgeois, A.G., Zheng, S.Q. (eds.) ICA3PP 2008. LNCS, vol. 5022, pp. 121–132. Springer, Heidelberg (2008)
9. Dincbas, M., Simonis, H., Van Hentenryck, P.: Solving large combinatorial problems in logic programming. The Journal of Logic Programming 8(1-2), 75–93 (1990)
10. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley, Reading (1989)
11. Gupta, H.: Selection of views to materialize in a data warehouse. In: ICDT, pp. 98–112 (1997)
12. Gupta, H., Mumick, I.S.: Selection of views to materialize under a maintenance cost constraint. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 453–470. Springer, Heidelberg (1998)
13. Horng, J.-T., Chang, Y.-J., Liu, B.-J.: Applying evolutionary algorithms to materialized view selection in a data warehouse. Soft Comput. 7(8), 574–581 (2003)
14. Kalnis, P., Mamoulis, N., Papadias, D.: View selection using randomized search. Data Knowl. Eng. 42(1), 89–111 (2002)
15. Labio, W., Quass, D., Adelberg, B.: Physical database design for data warehouses. In: Proceedings of the Thirteenth International Conference on Data Engineering ICDE 1997, pp. 277–288. IEEE Computer Society, USA (1997)
16. Lawrence, M.: Multiobjective genetic algorithms for materialized view selection in olap data warehouses. In: GECCO, pp. 699–706 (2006)
17. Lecoutre, C., Roussel, O., van Dongen, M.R.C.: Promoting robust black-box solvers through competitions. Constraints 15(3), 317–326 (2010)

18. Lee, M., Hammer, J.: Speeding up materialized view selection in data warehouses using a randomized algorithm. *Int. J. Cooperative Inf. Syst.* 10(3), 327–353 (2001)
19. Michalewicz, Z.: *Genetic algorithms + data structures = evolution programs*, 3rd edn. Springer, London (1996)
20. Mistry, H., Roy, P., Ramamritham, K., Sudarshan, S.: Materialized view selection and maintenance using multi-query optimization. CoRR, cs.DB/0003006 (2000)
21. Mistry, H., Roy, P., Sudarshan, S., Ramamritham, K.: Materialized view selection and maintenance using multi-query optimization. In: *SIGMOD Conference*, pp. 307–318 (2001)
22. Nilsson, N.J.: *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill Pub. Co., New York (1971)
23. Ross, K.A., Srivastava, D., Sudarshan, S.: Materialized view maintenance and integrity constraint checking: Trading space for time. In: *SIGMOD Conference*, pp. 447–458 (1996)
24. Roy, P., Seshadri, S., Sudarshan, S., Bhobe, S.: Efficient and extensible algorithms for multi query optimization. CoRR, cs.DB/9910021 (1999)
25. Roy, P., Seshadri, S., Sudarshan, S., Bhobe, S.: Efficient and extensible algorithms for multi query optimization. In: *SIGMOD Conference*, pp. 249–260 (2000)
26. Valluri, S.R., Vadapalli, S., Karlapalem, K.: View relevance driven materialized view selection in data warehousing environment. In: *Australasian Database Conference* (2002)
27. Wallace, M.: Practical applications of constraint programming. *Constraints* 1, 139–168 (1996); 10.1007/BF00143881
28. Widom, J.: Research problems in data warehousing. In: *CIKM*, pp. 25–30 (1995)
29. Yang, J., Karlapalem, K., Li, Q.: Algorithms for materialized view design in data warehousing environment. In: *VLDB*, pp. 136–145 (1997)
30. Yu, J.X., Yao, X., Choi, C.-H., Gou, G.: Materialized view selection as constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 33(4), 458–467 (2003)
31. Zhang, C., Yang, J.: Genetic algorithm for materialized view selection in data warehouse environments. In: Mohania, M., Tjoa, A.M. (eds.) *DaWaK 1999*. LNCS, vol. 1676, pp. 116–125. Springer, Heidelberg (1999)
32. Zhang, C., Yao, X., Yang, J.: An evolutionary approach to materialized views selection in a data warehouse environment. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 31(3), 282–294 (2001)

# Enabling Knowledge Extraction from Low Level Sensor Data\*

Paolo Cappellari<sup>1</sup>, Jie Shi<sup>1</sup>, Mark Roantree<sup>1</sup>,  
Crionna Tobin<sup>2</sup>, and Niall Moyna<sup>2</sup>

<sup>1</sup> Interoperable Systems Group, Dublin City University  
{pcappellari,jshi,mark.roantree}@computing.dcu.ie

<sup>2</sup> School of Health and Human Performance, Dublin City University  
{crionna.tobin9,niall.moyna}@dcu.ie

**Abstract.** While sensor networks play a significant role in the modern information society, they output data in proprietary format and with little or no associated semantics. As a consequence, sensed data must be managed on a case by case basis, requiring significant human efforts. In this paper, we present an approach that: seamlessly supports any kind of network by exposing sensed data in a standard format; enables users to specify at a high level how to enrich sensed data with the semantics in which data is generate; facilitates end users in transforming data to meet their analytical requirements.

## 1 Introduction

Recently, the area of personal health where monitors are attached to the human body, has seen a significant increase in popularity. The challenge in performing the necessary experiments is the very high volumes of data that will be generated. In our area of focus, coaches of high performance athletes use sensors to detect levels of fatigue and stress in preparation for upcoming matches and events. Often, exercise physiologists will spend days analysing spreadsheets simply to arrive at basic calculations such as average heart rates over specific periods of time, or comparisons of heart rates for selected athletes over different training sessions. In other words, there is a significant gap between high level user requirements and the data generated by sensors, and when this data is generated in high volumes, it provides a significant barrier to knowledge extraction.

**Data Collection and User Requirements.** This paper present a collaboration between the exercise physiologists and researchers from the Interoperable Systems Group (both groups at Dublin City University). The role of the physiologists was to collect the data using a set of sensors and ensure that participants followed the template for the activity. The challenge for the computer scientists was to close the *semantic gap* between end user requirements and raw datasets, by providing a number of processors and algorithms to transform the data to

---

\* Jointly funded by Enterprise Ireland Grants TD-2007-201 and CFTD-2008-231.

a format that could be queried using a high level query language. The goal of the study was to compare the effect of two exercise regimes on the athletes. In particular, the physiologists were interested in analysing the data produced by the following queries:

- Calculate the average HR for the first 2 minutes during the run period
- Find any 30 second period where the athlete’s HR was at 90% of his maximal
- Find any 20 second period where the athlete’s HR was at 90% of his maximal
- Find any 30 second period where the athlete’s HR was at 80% of his maximal
- Find any 20 second period where the athlete’s HR was at 80% of his maximal
- Find the point at which the athlete’s HR did not drop during the rest period

**Contribution.** The contribution presented in this paper is two-fold. Firstly, we provide a framework for the contextual enrichment of sensor data using a generic approach that eliminates the need for new sensor “wrappers” for each experiment. We also allow users to specify how to merge context and raw sensor data. Secondly, we provide data mining primitives to extract levels of knowledge required for the more complex user queries. In the evaluation, both user requirements (queries) and datasets are provided by the exercise physiologists and reflect the precise information needs to demonstrate the contrast between the short high intensity sprinting and the longer endurance tests.

**Paper Structure.** The paper is organised as follows. In §2 we give an overview of our approach; §3 presents the integration of raw sensor data with the context information in which has been generated; in §4 we discuss how non-expert users can extract knowledge from the sensed data; in §5 we provide experimental results and a discussion on the effectiveness of our proposal; §6 provides a discussion of the related research; finally, in §7 we draw our conclusions. More details on enrichment, evaluation and related research discussed in this paper are available in a longer version [4].

## 2 High Level Overview

The motivation for our system is to allow users to specify at a high level, how to structure, enrich, transform and query sensor data. The architecture of our framework is depicted in Fig. 1 as consisting of 4 major modules.

Data generated from the sensors is passed to the *Sensor Enablement*, which is responsible for structuring and mapping sensed data from their native, raw, format to a standard format. Raw data is transformed into OGC [8] by a template mechanism inspired from [9]. This mechanism intercepts and wraps raw data tokens into XML syntax, which in our case is OGC compliant. adopting the OGC standard, facilitates the interoperability and integration of the sensed data with other OGC compliant data. Within the template a user can define functions to perform time and data format transformation or timestamps generation.

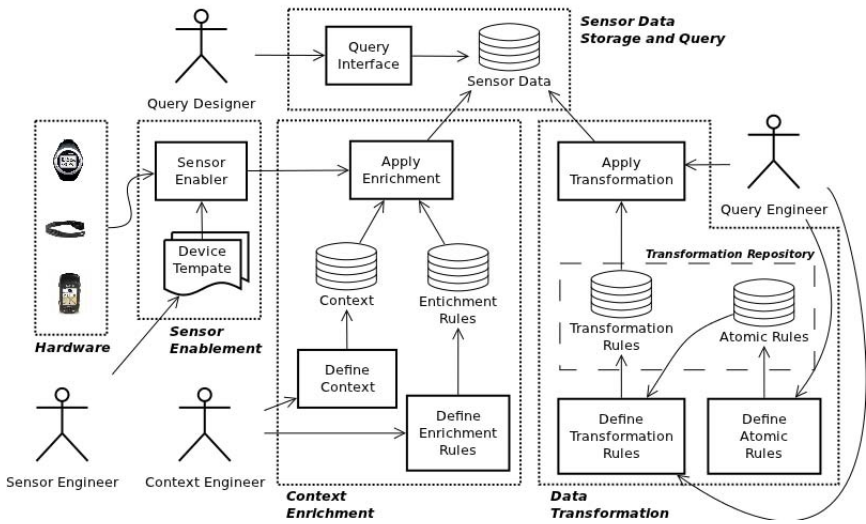


Fig. 1. Overview Architecture

The *Context Enrichment* merges the structured sensed data with the context information in which the data has been generated. At this point data is made available to both *Data Transformation* and *Query Interface*. Users can now access enriched sensed data with simple queries through the query interface by means of XQuery expressions. However, some queries are complex and require to transform or further enrich the data before answering these queries.

Context Enrichment integrates structured sensor data with context information. Sensor readings are devoid of context information: they are associated with some device identification rather than with some more useful information such as what is the athlete’s name generating the values, what training session is, what group the athlete belongs to, etc. As a result, a trainer who wants to know a simple information such as “what is the performance level of a (specific) athlete,” must first verify which sensor the athlete is wearing, then specify the query. More complex queries involving, for instance, information about teams or specific parts of an activity, require a significant amount of work in order to retrieve the data of interest. This module allows domain expert to specify context integration at a high level. Details are provided in §3.

Data Transformation allows to specify transformations on the dataset. It offers low-level data mining primitives (to calculate for peak, troughs, distances between peaks and troughs, search rolling averages and also calculate group averages) that can be assembled into complex programs, which executed transform the dataset. This operation is needed when queries are complex and cannot be expressed on the sensed-enriched data directly, and a transformation to expose specific features of the data is needed in order to answer such queries.

### 3 Providing Context for Low Level Sensor Data

Because the deploying scenario varies often, we focus on the provision of a method that enables non-expert users to specify how to enrich sensor data with context information. We enable users to specify context enrichment in an Event-Condition-Action (ECA) like paradigm. Once defined, ECA-like rules are transformed into standard XQuery expressions, that can be applied to the input data by any XML engine. Rules have a rather intuitive syntax. We use two pre-defined variables to refer to the input sources: *sensor* refers to the raw sensor data, while *context* refers to the context data. To navigate XML nested elements we adopt the . (dot), borrowing the notation from the object modelling.

A rule template is shown below. Each rule has three main sections: **on**, **when** and **do**. The **on** section specifies the event to which apply the rule. The event is specified by its name, such as “3min Running Test,” or if the rule is generic and applies all the events then the keyword **any** is used. The **when** section specifies the condition under which the rule must be executed. Generally, this is a condition on either the raw sensor data, the context data, or both. The **do** section specifies which action to perform as a manipulation of the input data.

```

on   : <Event>
when: <Condition>
do  : <Action>

```

**Simple Mapping.** Simple mapping rules specify integration of basic information, such as athlete’s name, training group, etc., with sensed data. Let us consider a rule that adds personal information about the athletes. For the sake of clarity in the presentation, let us assume that the rule only adds the athlete’s name. We can specify additional rules to further enrich the sensed data. To integrate the name of the athletes in the sensed data, we specify the following rule:

```

on   : 3min Running Test
when: sensor.deviceID = context.deviceID
do  : sensor.user = context.athelte.name

```

This rule applies to the “3min Running Test” data: when the device ID from the sensed data matches one from the context data, the rule enriches the stream with the athlete’s name corresponding to the such device ID. It is then converted to the following XQuery expression that can be executed by any XML engine:

```

let $c := collection("3_min_running")
for $x in $c//athletes/athlete, $y in $c//sos:sensorData
where $x/deviceID = $y/sos:deviceID
return do insert <sos:user>$x/name</sos:user> after $c//sos:eventTime

```

**Complex Mapping.** In our scenario, we have a “3 minutes test” organised in 6 consecutive sessions of 3 minutes each in which the athletes alternatively run or rest. We have to associate heart rate values with the session (state) in which they have been generated according to the static definition of the test. Values

before the beginning of the test are marked as *warmup*; the first run session is denominated *round 1* and the first resting session *break 1*; the second running session is *round 2* and so on; values after the completion are associated with the state *warmdown*. This association can be implemented as a sequence of ECA rules, one for each state. We offer a straightforward implementation through the *case* construct. The *case* allows to express multiple conditions, each associated with a single action, plus a default action if none of the conditions is satisfied. The following rule implements the described enrichment.

```

on : 3min Running Test
do: case
  when : sensor.time < context.start_time
  do   : sensor.state = "warmup"
  when : sensor.time >= context.start_time and
        sensor.time < (context.start_time + context.duration)
  do   : sensor.state = context.state.name
  default: sensor.state = "warmdown"
endcase

```

The XQuery equivalent is shown below. Note that the state information is inserted as an attribute into the measurement element. Listing 1.1 shows an example of enriched sensor data: it is explicit that this data has been generated from “Bryan” during the “3min Running Test” activity.

```

let $c := collection("3_min_running")
for $x in $c/experiment, $y in $c//sos:sections/sos:section/
  sos:measurement
return if ($y/@time < $x/start_time)
then do insert attribute state {'warmup'}
  as last into $c//sos:measurement[1]
else if ($y/@time >= $x/start_time)
  and ($y/@time < ($x/start_time + $x/states/state/duration))
then do insert attribute state {$x/states/state/name}
  as last into $c//sos:measurement[1]
else if ($y/@time > ($x/start_time + $x/states/state/duration))
then do insert attribute state {'warmdown'}
  as last into $c//sos:measurement[1]
else()

```

## 4 Enabling Knowledge Extraction

Contextual enrichment facilitates a range of basic queries but cannot handle more complex analyses such as those presented in this study. Given the simple nature of sensor data (often a single value generated at standard intervals) it is possible to predefine a series of operations to generate new knowledge. A set of data mining primitives are used to run a standard set of analyses to which end users can add their own semantics. The primitives we currently provide are:

- Count(startTime,endTime) - Number the readings for a participant over the activity between times;
- Min(state) - Minimal reading for a participant over the activity for a state;
- Max(state) - Maximum reading for a participant over the activity for a state;



```

<?xml version="1.0" encoding="UTF-8" ?>
<sos:GetResult xmlns:sos="http://www.opengis.net/sos/1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/sos/1.0 http://schemas.
  opengis.net/sos/1.0.0/sosAll.xsd" service="SOS" version="1.0.0">
<sos:ObservationTemplateId>urn:DCU:ObservationTemplate:HeartRate</
  sos:ObservationTemplateId>
  <sos:eventTime>
    <ogc:TM_After>
      <gml:TimeInstant>
        <gml:timePosition>2010-08-04T16:00:00Z</gml:timePosition>
      </gml:TimeInstant>
    </ogc:TM_After>
  </sos:eventTime>
<sos:user>Bryan</sos:user>
<sos:session>3min Running Test</sos:session>
<sos:device>HRM</sos:device>
<sos:deviceID>S1_1</sos:deviceID>
<sos:startTime>1209826519000</sos:startTime>
<sos:interval>5000</sos:interval>
<sos:sections>
  <sos:section name="Params">
    <sos:parameter><sos:key>Version</sos:key>
      <sos:value>106</sos:value>
    </sos:parameter>
    ... (other parameters)
  </sos:section>
  <sos:section name="HRData">
    <sos:measurement offset="0" state="Run 1" time="1209826519000">
      <sos:reading ordinal="">
        <sos:key>HeartRate</sos:key>
        <sos:value>71</sos:value>
      </sos:reading>
    </sos:measurement>
    ... (measurement element repeats)
  </sos:section>
</sos:GetResult>

```

**Listing 1.1.** Sensor data enriched with context information

- As % of Max(startTime,endTime) - Readings as a percentage of the maximal for a participant during the activity between times;
- Average(startTime,endTime) - The average reading of a participant over the activity between times;
- RollingAvg(startTime,endTime) - The N minutes rolling average for each sensor for each participant over the activity between times.
- Diff(a,b) - Difference between two evaluated measures or two readings.

In our experimental setting, exercise physiologists were interested in answering the queries provided in §1. These queries can be answered by selecting one of the built-in functions or by composing a few of them in a sequence. For example to answer the first query only the function Average(startTime,endTime) is needed. On the other hand, answering the second one requires a combination of these functions. Specifically, we first need Max(state) to find the maximum sensor readings for each athlete data stream. Then function As % of Max(startTime,endTime) calculates the 90% of the maximum sensor readings. Finally, function RollingAvg(startTime,endTime) is used to calculate a 30-second rolling average for each sensor reading. Results produced at each

transformation step are inserted into the enriched sensor dataset and queries through relatively simple XPath/XQuery expressions.

Data transformation primitives are stored in our atomic rule repository and can be retrieved to compose more complex programs (macro transformation, sequence of atomic rules). Thus, whenever the functions or the rules available in the system are not enough, a user can extend (or customise) the rule portfolio by providing new functions or rules. Note that in this process there is never the need to alter the system itself: as rules are stored in their own repositories, they are completely decoupled from the system implementation.

## 5 Evaluation

In this section we report on the experiments performed to validate our approach and its impact on the exercise physiologists work. Queries listed in §1 are converted into XQuery expressions. Table 1 shows, as an example, the expression for first query only. The full table with the expressions for all the queries is available in [4]. Let us emphasise the following two facts. Queries are executed on the sensor data obtained after the above discussed transformations: without such transformations these queries are either difficult or impossible to express on the original dataset. Queries are implemented as standard XQuery expressions: if needed, they can be modified and customised using a simple text editor thus the system implementation does not need to be altered.

**Table 1.** Full Query Expressions.

#	Query as XQuery expression	Results
1	<pre>let \$c := collection("3_min_running") for \$r in \$c//sos:measurement where \$r/@offset &gt;= 0 and \$r/@offset &lt;= 1200000 return if (\$r/@state="Run") then fn:avg(\$r//sos:measurement/sos:reading [sos:key[text()='HeartRate']]/sos:value/text()) else()</pre>	<p>AvgHR = 156.7</p>

Exercise physiologists demonstrated a favourable opinion on the prototype. While avoiding manual error prone processes, they have been able to extract the knowledge and perform the analysis of interest rather quickly. They also realised the possibility of having access to a wider information and analysis capacity they had before using this prototype. The queries posed during this study enabled physiologists to determine how hard each subject trained and whether her level of training was adequate to induce the training adaptations required.

## 6 Related Research

In [1] authors develop a wearable light device capable of measuring specific vital signs of the elderly, detecting falls and location, and communicating automatically

in real-time. Both [2] and [3] investigate how Sensor Web enablement services work in healthcare sensor networks. They enrich multiple sources of sensor data into a data model that represents different sensors and data as a series of observations. These approaches focus on simple live sensor data from multiple sources, obtaining query results on the basis of a series of simple rules applied to the streams. On the other hand, we focus on analysing offline sensor data to facilitate far more complex queries by non-IT domain experts.

TinyDB [7] and Cougar [11] require end users to understand the operators running over the raw sensor data and interpret the meaning of the results. Works [10,6,5] operate on raw, proprietary, format. In [10], raw data is enriched into “semantic streams” and processed as they are generated. However, this work is still theoretical. In [6], node locality is exploited to answer queries. In [5], which addresses contextual synthesis of sensor networks in the sports domain. All these approaches require to develop ad hoc programs to implement the queries. In our approach, data is stored in XML, it is enriched and the transformations processes allow domain experts to extract knowledge by far simpler queries in standard languages (XQuery), which is decoupled from the system implementation.

## 7 Conclusions

Sensors often generate large volumes of data, making analysis very difficult. There is no standard format for query output, no method of merging results from different tests or integrating results across groups, and a full query interface to datasets does not exist. In this paper, we presented a framework for capturing low level sensor data and through a number of enrichment processes to transform data to a position where high level query expressions were possible, simplifying information extraction task as demonstrated with our evaluation.

## References

1. Artur, R., Angelo, M., Jose, C., et al.: Innovations in Health Care Services: The CAALYX System. *International Journal of Health Geographics* (2010)
2. Churcher, G., Foley, J., Bilchev, G., et al.: Experiences applying Sensor Web Enablement to a practical Telecare application. In: ISWPC (2008)
3. Churcher, G., Foley, J.: Applying and Extending Sensor Web Enablement to a Telecare Sensor network Architecture. In: ICST (2009)
4. Cappellari, P., Shi, J., Roantree, M., Tobin, C., Moyna, N.: Enabling Knowledge Extraction from Low Level Sensor Data, Technical Report No. ISG-11-01, pp. 1-15, at: isg (2011), <http://www.computing.dcu.ie>
5. Devlic, A., Koziuk, M., Horsman, W.: Synthesizing Context for a Sports Domain on a Mobile Device. In: Roggen, D., Lombriser, C., Tröster, G., Kortuem, G., Havinga, P. (eds.) EuroSSC 2008. LNCS, vol. 5279, pp. 206–219. Springer, Heidelberg (2008)
6. Kotidis, Y.: Processing Proximity Queries in Sensor Networks. In: DMSN, pp. 1–6 (2006)
7. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.T.: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. In: OSDI, pp. 131–146 (2002)

8. Mike, B., George, P., Carl, R., John, D.: Sensor Web Enablement: Overview and High Level Architecture (2006)
9. McCann, D., Roantree, M.: A Query Service for Raw Sensor Data. In: Barnaghi, P., Moessner, K., Presser, M., Meissner, S. (eds.) EuroSSC 2009. LNCS, vol. 5741, pp. 38–50. Springer, Heidelberg (2009)
10. Whitehouse, K., Zhao, F., Liu, J.: Semantic Streams: a Framework for Composable Semantic Interpretation of Sensor Data. In: Römer, K., Karl, H., Mattern, F. (eds.) EWSN 2006. LNCS, vol. 3868, pp. 5–20. Springer, Heidelberg (2006)
11. Yao, Y., Gehrke, J.: Query processing for sensor networks. In: CIDR (January 2003)

# Join Selectivity Re-estimation for Repetitive Queries in Databases

Feng Yu<sup>1</sup>, Wen-Chi Hou<sup>1</sup>, Cheng Luo<sup>2</sup>, Qiang Zhu<sup>3</sup>, and Dunren Che<sup>1</sup>

<sup>1</sup> Southern Illinois University, Carbondale, IL 62901, USA  
{fyu,hou,dche}@cs.siu.edu

<sup>2</sup> Coppin State University, Baltimore, MD 21216, USA  
cluo@coppin.edu

<sup>3</sup> University of Michigan, Dearborn, MI 48128, USA  
qzhu@umich.edu

**Abstract.** Repetitive queries refer to those queries that are likely to be executed repeatedly in the future. Examples of repetitive queries include those that are used to generate periodic reports, perform routine maintenance, summarize data for analysis, etc. They can constitute a large part of daily activities of the database system and deserve more optimization efforts. In this paper, we propose to collect information about joins of a repetitive query, called the trace, during execution. We intend to use this information to re-estimate selectivities of joins in all possible execution orders. We discuss the information needed to be kept for the joins and design an operator, called the extended full outer join, to gather such information. We show the sufficiency of the traces in computing the exact selectivities of joins in all plans of the query. With the exact selectivities of joins available, the query optimizer can utilize them to find truly the best join order for the query in its search space, guaranteeing “optimal” execution of the query in the future.

**Keywords:** Join Selectivity Estimation, Query Re-optimization.

## 1 Introduction

A primary problem in query optimization is to find the most efficient execution plan for a query, which is mainly determined by the join orders. In order to find the best join order, accurate cost estimations of alternative join orders must be known. Query optimizers generally use statistics stored in the database catalogs, such as histograms [3–5], etc., and assumptions about attribute values [1, 7] to estimate the cost. Unfortunately, due to the complexity of queries, sufficiency of statistics, and validity of assumptions, query optimizers often cannot find the most efficient join orders for the queries in their search spaces. Studies [1, 2] have shown that it could be orders of magnitude slower in speed when executing queries with sub-optimal plans. Thus, some database systems, like Sybase and Oracle, allows users to force the join orders; some, e.g., Sybase, even allows users

to explicitly edit the plans [6]. Unfortunately, such measures cannot guarantee success and can also be quite cumbersome and slow for complex queries. Clearly, there is a tremendous need for a mechanism that can automatically refine the execution plans of queries.

Repetitive queries refer to those that are likely to be posted repeatedly in the future. Many useful queries, such as those used for generating periodical reports, performing routine maintenances, summarizing and grouping data for analysis, etc., are repetitive queries. They are often stored in databases for convenient reuses for the long term. Any sub-optimality in the execution plans of such queries could mean repetitive and continued waste of system resources and time in the future. The efficiency of the executions of repetitive queries has a paramount effect on the performance of the system and thus deserves more optimization efforts. In this research, we attempt to gather information about a repetitive query while it is being executed. We show that the information gathered is sufficient for optimizers to compute the selectivities of joins accurately in all execution (or join) orders. It is worth mentioning that we do not intend to modify the search strategy of the optimizer, but just to provide it with sufficient and accurate information so that it can find truly the best join order for the query in its solution space, conveniently called the optimal plan here.

In this paper, we introduce the notion of the trace of a query, which contains information about how tuples from input relations are joined in the query. We propose to collect the trace of a query during execution and use it to re-estimate selectivities of joins in all alternative plans of the query derived by exchanging the input relations. We have designed operators to collect sufficient information in the traces so that the exact join selectivities in all execution orders can be computed. With the exact join selectivities known, the query optimizer can find the best execution plan for a repetitive query for future executions. Substantial saving can be obtained from running such queries with optimal plans, not to mention running them repeatedly. This work makes a major stride in the research of query re-optimization and can make significant improvement on the performance of the system.

The rest of the paper is organized as follows. Terminologies and definitions used in the paper are introduced in Section 2. Section 3 discusses selectivity estimation using traces for queries with acyclic join graphs. Due to space limitation, interested readers are referred to [8] for queries with cyclic join graphs. The analysis of overhead is included in section 4. Section 5 presents the conclusions and future work.

## 2 Framework and Terminology

In this section, we describe the selectivity re-estimation framework and introduce terminology used in the paper.

## 2.1 Join Selectivity Re-estimation Framework

We attempt to gather information about how tuples are joined in a query while the query is being processed. The information to gather here is called the query (or join) trace.

We assume that there are mechanisms in the database that can differentiate a repetitive query from an ordinary query. We also assume an optimizer knows how to compute the selectivities of joins from the trace, which will be discussed in the next two sections.

Figure 1(a) depicts the framework. A query is first optimized by the query optimizer as usual. If it is a repetitive query, its trace will be gathered when the query is executed. Once the execution completes, the trace collected will be provided to the optimizer to compute selectivities of joins of alternative plans and select the best plan. Physical execution plans are generated based on the best logical plans by the optimizer and stored in the database for future invocations of the query. Certainly, after the database has gone through substantial changes, the process of trace gathering and re-optimization can be re-invoked.

In this paper, we discuss the information to be gathered, and how to gather them. In addition, we show the sufficiency of the gathered information for computing the exact selectivities of joins of alternative plans.

## 2.2 Join Graph

A query can be modeled by a graph, called the join graph, that describes the join relationships among participating relations.

**Definition 1 (Join Graph).** *The join graph  $G(V, E)$  of a query consists of a set of vertices  $V = \{R_1, R_2, \dots, R_n\}$  and a set of edges  $E$ . Each vertex denotes an input relation of the query and each edge  $(R_i, R_j) \in E, R_i, R_j \in V$ , denotes the existence of join conditions between  $R_i$  and  $R_j$  in the query.*

*Example 1 (Join Graph).* Fig. 1(b) shows the join graph of a query where there are join conditions placed between  $R_1$  and  $R_2$ ,  $R_2$  and  $R_3$ , and  $R_3$  and  $R_4$ .

If the join graph of a query is disconnected, we can consider each connected component separately (and then merge them by Cartesian products). Therefore, we shall assume hereafter all queries have connected join graphs or all join graphs are connected. For simplicity, we further assume all joins are equi-join, though our approach can be applied to other joins, such as non-equi joins, and Cartesian products.

In this research, we assume every (execution) plan  $P$  is in the form of a left-deep tree  $P = (\dots((R_1 \bowtie R_2) \bowtie R_3)\dots) \bowtie R_n$  [6] because most, if not all, commercial database systems generate such plans for executions. We assume all  $\bowtie$ 's are equi-joins and no Cartesian product appears in  $P$ . It is worth mentioning that the method proposed is applicable to bushy trees and right-deep trees as well.

Let  $G(V, E)$  be the join graph of a query  $Q$ . In this research, we are interested in estimating the selectivities of all possible subqueries that are joins of some or all of the relations in  $V$ . Note that we assume all subgraphs  $G'(V', E')$  are connected, and no Cartesian product exists in any of the subqueries or plans.

**Definition 2 (Joinable Relations).** *A pair of relations  $R_i$  and  $R_j$  are said to be joinable in a query if there is an edge  $(R_i, R_j)$  in the join graph of the query.*

**Definition 3 (Joinable Tuples).** *A pair of tuples  $t_i$  and  $t_j$ ,  $t_i \in R_i, t_j \in R_j$ , are said to be joinable if  $t_i$  and  $t_j$  have the same value for the join attributes of  $R_i$  and  $R_j$ . Joinable tuples are also referred to as match tuples.*

**Theorem 1.** *Given a connected join graph for a query and an execution plan  $P = (\dots((R_1 \bowtie R_2) \bowtie R_3)\dots) \bowtie R_n$ , each relation  $R_k, 2 \leq k \leq n$ , has join attributes with exactly one prior relation  $R_i, i < k$ , in  $P$  if and only if the join graph of the query has no cycle.*

*Proof.* See [8].

### 2.3 Query Trace

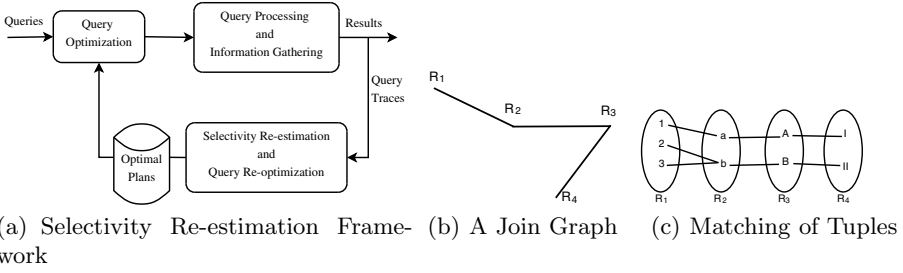
When a query is being processed, information about how tuples are joined is gathered. We intend to use this information, called query (or join) trace, to estimate selectivities of joins in all execution orders.

*Example 2.* Fig. 1(b) is the join graph of a query, and Fig. 1(c) shows the matching of join attribute values between tuples. For simplicity, we have represented a tuple only by its added IDs without reference to other attribute values, that is,  $R_1 = \{1, 2, 3\}, R_2 = \{a, b\}, R_3 = \{A, B\}, R_4 = \{I, II\}$ . For example, tuple 1 of  $R_1$  matches tuple a of  $R_2$ , and tuples 2 and 3 match tuple b of  $R_2$ . Tuples a and b of  $R_2$  match tuples A and B of  $R_3$ , respectively. Finally, tuples A and B of  $R_3$  match tuples I and II of  $R_4$ , respectively.

Consider a left-deep tree execution plan  $P = ((R_1 \bowtie R_2) \bowtie R_3) \bowtie R_4$ . To generate the trace, an ID attribute is added to every relation and the attribute is to be preserved in the outputs of all operators. Thus, the result of  $R_1 \bowtie R_2$ , as shown in Fig. 1(d), besides its normal set of attributes, denoted by Result-Attrs, has additional attributes  $R_1$ -ID and  $R_2$ -ID, called the trace of  $R_1 \bowtie R_2$ , denoted by  $T(R_1 \bowtie R_2)$ .

Fig. 1(f) shows the trace of  $((R_1 \bowtie R_2) \bowtie R_3) \bowtie R_4$ , denoted by  $T(((R_1 \bowtie R_2) \bowtie R_3) \bowtie R_4)$ . Once a query is completely processed, we can extract the final trace, e.g.,  $T(((R_1 \bowtie R_2) \bowtie R_3) \bowtie R_4)$  in Example 2, from the “extended” query result by a simple projection on all the added ID attributes.





Result-Attrs	R <sub>1</sub> -ID	R <sub>2</sub> -ID
...	1	a
...	2	b
...	3	b

(d) Result and Trace of R<sub>1</sub> ⋈ R<sub>2</sub>

R <sub>1</sub> -ID	R <sub>2</sub> -ID	R <sub>3</sub> -ID
1	a	A
2	b	B
3	b	B

(e) Trace of (R<sub>1</sub> ⋈ R<sub>2</sub>) ⋈ R<sub>3</sub>

R <sub>1</sub> -ID	R <sub>2</sub> -ID	R <sub>3</sub> -ID	R <sub>4</sub> -ID
1	a	A	I
2	b	B	II
3	b	B	II

(f) Trace of ((R<sub>1</sub> ⋈ R<sub>2</sub>) ⋈ R<sub>3</sub>) ⋈ R<sub>4</sub>

Fig. 1. Query Traces

### 3 Selectivity Estimation for Acyclic Join Graphs

In this and next sections, we discuss information that needs to be incorporated into the traces in order to estimate selectivities of joins accurately.

Let  $Q$  be a query with an acyclic join graph  $G(V, E)$  and  $P$  an execution plan of the query. Let  $T(P)$  be the final trace of  $P$ . Let  $G'(V', E')$  be a vertex-induced connected subgraph of  $G(V, E)$ , in which  $V' = \{R_{i_1}, \dots, R_{i_m}\} \subseteq V$  and  $E' \subseteq E$ , representing a subquery  $Q'$  of  $Q$ . The selectivity of  $Q'$  can be estimated as

$$\widetilde{sel}(Q') = \frac{|\pi_{R_{i_1}\text{-ID}, \dots, R_{i_m}\text{-ID}} T(P)|}{|R_{i_1}| \times \dots \times |R_{i_m}|} \tag{1}$$

in which  $\pi_{R_{i_1}\text{-ID}, \dots, R_{i_m}\text{-ID}} T(P)$  is the projection of trace  $T(P)$  on attributes  $R_{i_1}\text{-ID}, \dots, R_{i_m}\text{-ID}$ , without duplicate.

#### 3.1 No Dangling Tuples in the Joins

Here, we assume no dangling tuple exists in any of the joins in the query, i.e. every tuple in one relation finds at least one matching tuple in another relation with which there is a join edge in the join graph. The relations in Fig. 1(c) satisfy this condition.

**Theorem 2.** *Let  $P$  be an execution plan of a query  $Q$  with a connected acyclic join graph  $G(V, E)$ . Let  $Q'$  be a subquery of  $Q$  that has a vertex-induced connected join subgraph  $G'(V', E')$ ,  $V' = \{R_{i_1}, \dots, R_{i_m}\} \subseteq V$ . If there is no dangling tuple in any join of  $P$ , Eq. (1) derives the exact selectivity of  $Q'$  from  $T(P)$ .*

*Proof.* See [8].

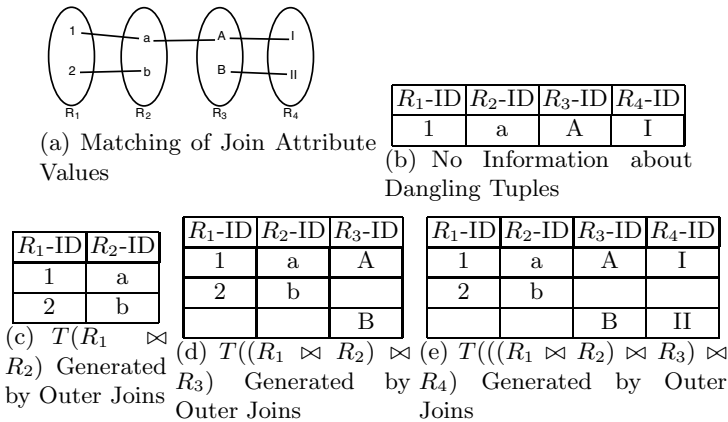


Fig. 2. Dangling Tuples in Relations

overlap	90%		80%		70%	
Rel. size	time	result size	time	result size	time	result size
10k	3.12%	21.03%	3.87%	47.60%	4.98%	81.51%
100k	2.46%	20.23%	5.46%	45.79%	6.30%	78.19%

Fig. 3. Overhead: outerjoin vs. join

### 3.2 Dangling Tuples in Joins

Dangling tuples are lost in the joins. To retain matching information about dangling tuples, we replace the joins in the query by the full outer joins ( $\bowtie$ ). Fig. 2(c) to 2(d) show the traces generated at different stages of query execution, where the joins are replaced by the full outer joins. The trace in Fig. 2(c) is the same as it were generated by a join because there is no dangling tuple in the join. The trace in Fig. 2(d) retains information about dangling tuples b in  $R_2$  and B in  $R_3$  by the outer join. Fig. 2(d) is the final trace that will be retained and used in later selectivity estimation.

The estimated selectivities for  $R_1 \bowtie R_2$ ,  $R_2 \bowtie R_3$ , and  $R_3 \bowtie R_4$  are now, by Eq. (1),  $\frac{1}{2}(= \frac{2}{2 \times 2})$ ,  $\frac{1}{4}(= \frac{1}{2 \times 2})$ , and  $\frac{1}{2}(= \frac{2}{2 \times 2})$ , respectively, which are exact. Note that, as mentioned earlier, a trace tuple having a null for any of the projected attributes is not accounted for in the respective  $|\pi_{R_{i_1}\text{-ID}, \dots, R_{i_1}\text{-ID}} T(P)|$  because a null in a  $R_{i_j}$ -ID column of a trace tuple indicates that there is no match found in  $R_{i_j}$  for the respective combination of tuples to generate an output in the (sub)query. One can easily verify that the estimated selectivities for all other subqueries are all exact.

**Theorem 3.** *Let  $P$  be an execution plan of a query  $Q$  with a connected acyclic join graph  $G(V, E)$ . Let  $Q'$  be a subquery of  $Q$  that has a vertex-induced connected*

join subgraph  $G'(V', E')$ ,  $V' = \{R_{i_1}, \dots, R_{i_m}\} \subseteq V$ . Eq. (1) derives the exact selectivity of  $Q'$  from the trace obtained by replacing the joins in the query with the full outer joins, denoted by  $T(P)$ .

*Proof.* See [8]

## 4 Preliminary Experimental Results

### 4.1 Preliminary Experimental Results

We test two cases where the synthetic relations have 10K and 100K tuples. Each input relation and the result relation has 5 attributes. By overlapping parts of the domains of join attributes, we generate match tuples, partial match, and no-match tuples.

Relations are read into memory for processing. The CPU cost accounts for the cost of all processing and the writing of outputs to memory. We use result size as a measure for potential I/O cost if the result cannot fit in memory. Table 3 shows the CPU and result size (in terms of the number of tuples) overheads for the outer join operator. The overheads are computed as  $(OJ - J)/J$ , where  $OJ$  and  $J$  represent the CPU time and the result sizes of outer join and join, respectively.

The CPU overheads (i.e., 3.12%, 3.87%, 4.98%) are still quite small. This is because the same amounts of computations, for hashing and comparisons, need to be performed for both the join and outer join. Only copying dangling tuples to the output relation (in memory) is extra, which does not take much time. It is noted that the results could have been better or worse depending on the amounts of dangling tuples generated in the relations.

From the experiments, we observe that CPU overhead is much more acceptable than result size overhead. Therefore, if the memory is large enough to hold the trace at each stage, the trace gathering can be performed with query evaluation with too much of delay. On the other hand, if the memory is too small to hold the traces, the result size overhead could dramatically slow down the query processing. If that is the case, we may have to gather the trace off-line by running the query again in spare time.

## 5 Conclusions and Future Work

In this paper, we propose to collect information about joins, called traces, to re-estimate the selectivities of joins of repetitive queries. We have shown that the exact selectivities of joins in all execution orders of a query can be computed from its trace. In the future, we shall empirically study the overheads incurred in the the process of trace gathering more thoroughly.

## References

1. Christodoulakis, S.: Implications of certain assumptions in database performance evaluation. *ACM Trans. Database Syst.* 9, 163–186 (1984)
2. Markl, V., Raman, V., Simmen, D., Lohman, G., Pirahesh, H., Cilimdžić, M.: Robust query processing through progressive optimization. In: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, SIGMOD 2004*, pp. 659–670. ACM, New York (2004)
3. Muralikrishna, M., DeWitt, D.J.: Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In: *SIGMOD Conference*, pp. 28–36 (1988)
4. Piatetsky-Shapiro, G., Connell, C.: Accurate estimation of the number of tuples satisfying a condition. In: *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, SIGMOD 1984*, pp. 256–276. ACM, New York (1984)
5. Poosala, V., Ioannidis, Y.E.: Selectivity estimation without the attribute value independence assumption. In: *Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB 1997*, pp. 486–495. Morgan Kaufmann Publishers Inc., San Francisco (1997)
6. Ramakrishnan, R., Gehrke, J.: *Database Management Systems*, 3rd edn. McGraw-Hill, New York (2003)
7. Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G.: Access path selection in a relational database management system. In: *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, SIGMOD 1979*, pp. 23–34. ACM, New York (1979)
8. Yu, F., Hou, W.-C., Luo, C., Zhu, Q., Che, D.: Join selectivity re-estimation for repetitive queries in databases, <http://www2.cs.siu.edu/~fyu/main-trace.pdf>

# Matching Star Schemas

Dariusz Riazati and James A. Thom

School of Computer Science and Information Technology

RMIT University,

Melbourne, Australia, 3001

dariusz.riazati@student.rmit.edu.au, james.thom@rmit.edu.au

**Abstract.** Star schemas describe the structure and properties of multidimensional sources such as data marts and data warehouses. They have a simple structure and a predictable topology. We propose StarMod a representation of Star schema model described in UML and infer its instances from relational schemas. StarMod includes a comprehensive set of properties specific to multidimensional data with a view to its application in matching Star schemas. This paper demonstrates that in comparison to using the relational model, the quality of matching between Star schemas is improved if they are described using a more precise model such as StarMod, even if these Star properties are inferred from the relational schema. We demonstrate that StarMod can be also effective for matching arbitrary non-Star relational schemas.

## 1 Introduction

Star schema is a relational model for multidimensional sources such as data marts and data warehouses [12]. Despite attempts to provide industrial strength solutions to automate the matching of relational schemas [4, 9], most organizations prefer to employ experts to perform the matching and integration of their relational databases. This is tolerated as organizational changes that necessitate the integration do not occur frequently. However, in a business intelligence environment, there is always a need for reports to be generated based on integration of multiple sources given short notice.

A common application of a semi-automated Star schema matching is where a data analyst wishes to combine data marts for related subject areas. Other motivating scenarios are (i) where a business enterprise with disparate data warehouses for different lines of business such as banking, insurance and wealth needs to combine some of its data marts around customer and address details for marketing campaigns; (ii) integration of local data marts with externally sourced data marts such as those sourced from bureau of statistics.

Generic relational properties such as *table* and *column* fail to describe properties of Star schemas precisely. Star schemas designed correctly by conforming to principles specified by Kimball and Ross [12] have a certain topology and hence are more restricted than arbitrary relational schemas which makes them to be more predictable. This allows us to anticipate their model and identify their distinct properties.

We propose StarMod as a representation of Star schema model. We use UML (Unified Modeling Language) to provide a visual description of StarMod but implement it using RDFS (Resource Description Framework Specification) and OWL (Web Ontology Language) with their constructs corresponding to the UML representation. StarMod includes a predefined set of concepts, relationships and constraints for describing multidimensional data and in particular Star schemas, which can be used as a template to define Star schemas for any domain. We also present a set of heuristics to infer instances of StarMod (used in the matching process) from XML schemas corresponding to relational schemas.

The immediate application of StarMod is a more precise and consistent definition of Star based schemas, but more importantly we demonstrate that when compared to the relational model, using StarMod to describe Star schemas improves the accuracy of their match results. We also demonstrate that StarMod can be also effective for matching arbitrary relational schemas.

In our evaluation, we use Similarity Flooding (SF) [16] to perform matching between a number of Star and non-star schemas described using the relational model and StarMod, and compare their results. All results are then compared against those obtained from COMA++ [6] for the same schemas.

The remainder of this paper is as follows: Section 2 describes StarMod and its implementation; Section 3 describes our evaluation of using StarMod in matching Star and non-Star schemas; Section 4 offers a review of related works; Section 5 includes our concluding remarks.

## 2 StarMod

Star schema is a relational model specifically for relational implementation of multidimensional data. The topology of tables in this model resembles a star at the center of which is a (fact) table that contains additive measures. This table includes foreign keys referring to one or more (dimension) tables containing attributes used as categories of data to summarize measures. Figure 1 shows our running example of a Star schema. The fact table is MONTHLY\_SALES and dimension tables are MAKE, MODEL, DEALER, FISCAL\_CAL.

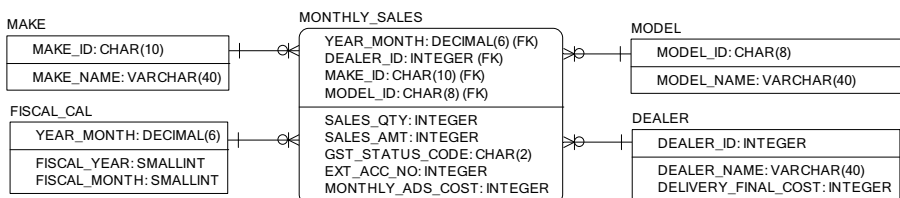


Fig. 1. An example of a Star schema

## 2.1 StarMod Properties and Their Application in Schema Matching

A **Star** represents a single Star schema and is defined as an aggregation of a number of dimension and fact tables. Each **DimensionTable** is a subtype of a *Table* and is an aggregate of one or more *Attributes* being subtypes of *Column*. Dimension and fact tables are related through *Keys*. Making distinction between dimension and fact tables improves the similarity between matching dimension and fact tables as well as their respective columns.

There are three types of *Attributes*: (i) A **SurrogateKeyAttribute** is a unique sequential number that is generated for every new row. It is important to separate surrogate keys from other types of attributes as they do not participate in any aggregation function, and their classification as such reduces their similarity with natural keys and other attributes. For example `customer_key` may be a customer number in one dimension and a surrogate key in another. (ii) A **DegenerateFact** is an additive attribute to which data functions such as *Sum* or *Average* can be applied. For example `no_of_claims` as an attribute of dimension table `Claimant` is classified as a *DegenerateFact* whereas `claim_no` as an attribute of dimension table `Claim` is not, even though they have identical data types and similar labels. Degenerate facts are matched with *measures* and degenerate facts. (iii) **DataAttributes** are generally used as categories by which measures are summarized. Their classification as such allows them to be matched with data attributes and degenerate dimensions (described later in this Section).

A **FactTable** is also a subtype of *Table* and is an aggregate of one or more *Elements* being subtypes of *Column*. There are four types of *Elements*: (i) **SurrogateKeyReference** is a foreign key that refers to a *SurrogateKeyAttribute* of a dimension table. By classifying surrogate key references as such, they are better distinguished from surrogate keys in dimension tables. (ii) The role of **SurrogateKeyElement** in a fact table is similar to that of *SurrogateKeyAttribute* in a dimension table. (iii) A **Measure** is an additive (or quantitative) element such as `number_of_claims` in fact tables [8]. (iv) A **DegenerateDimension** is a dimension attribute for which no dimension exists [10]. Use of degenerate dimensions is common in Star schemas. By classifying degenerate dimensions as such, we are able to separate them from measures and increase their similarity with data attributes. For example, a degenerate dimension called `claim_no` is distinguished from a measure called `number_of_claims`.

**Hierarchy and Level:** Each dimension *hierarchy* consists of a set of *levels* between which there exists a partial order. For example, given levels  $l_1$  and  $l_2$  where  $l_1 \preceq l_2$ , we say that  $l_1$  *rollsUpTo*  $l_2$ . Each *level* has one or more *DataAttributes* that uniquely identify it.

We represent hierarchies as part of StarMod for completeness of the representation but we do not use them in the schema matching process for several reasons. Matching attributes may belong to mismatching hierarchies. However, rejecting these matches as early as the schema matching can be restrictive. In many cases, it is possible to achieve coherence between hierarchies by applying some constraints against the data [18]. Moreover, even if hierarchies are found

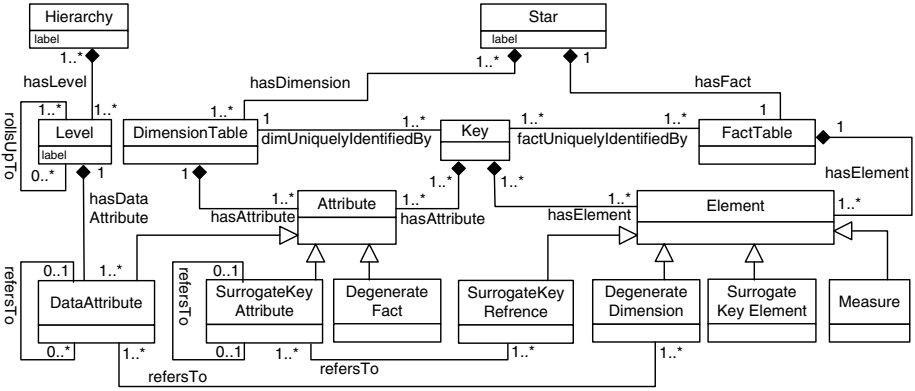


Fig. 2. UML representation of StarMod

to be matching, the data after integration may not conform to the original hierarchies [19]. Also, dimension hierarchies are not always readily available and where inferred are only estimates and subsets of their intended hierarchies [18].

## 2.2 Inferring Properties of StarMod from Relational Schemas

In this section, we describe a set of heuristics that we use to infer Star properties described in Section 2.1 to describe relational schemas using StarMod properties. These heuristics use a combination of data types, keys (which also include indexes) and foreign keys. The input to this process is an XML schema obtained from an existing relational database. A program written in XSLT language uses these heuristics to transform the XML schemas (corresponding to the relational schema) to instances of StarMod and described in OWL.

**FactTable and DimensionTable:** We say that if the primary key of a table does not appear as foreign key in any table, then we classify the table as a *FactTable*, otherwise it is classified as a *DimensionTable*.

**SurrogateKeyAttribute:** A column whose owning table is a *DimensionTable* is classified as a *SurrogateKeyAttribute* if it is defined as a primary key and has a constraint for having its value generated.

**DegenerateFact:** Identifying degenerate facts is a difficult task as it requires domain information on the meaning and purpose of the column. In absence of such information, we classify a column as a *DegenerateFact* only if its owning table is classified as a *DimensionTable* and its data type is *xsd:decimal* with a restriction defined as *xsd:fractionDigits*.

**DataAttribute:** A column whose owning table is a *DimensionTable* and is not a *SurrogateKeyAttribute* or *DegenerateFact* is classified as *DataAttribute*.

**SurrogateKeyElement:** A column whose owning table is a *FactTable* is classified as a *SurrogateKeyElement* if it is defined as part of a key and has a constraint for having its value generated (similar to the *SurrogateKeyAttribute*).



**SurrogateKeyReference:** A column whose owning table is a *FactTable* is classified as a *SurrogateKeyReference* if it refers to a *SurrogateKeyAttribute*.

**DegenerateDimension:** Similar to *DegenerateFacts*, accurate classification of degenerate dimensions requires additional semantic information. We classify a column as a *DegenerateDimension* if its owning table is a *FactTable*, and one of the following is true: i) its data type is *xsd:string*, ii) its data type is *xsd:integer* or *xsd:short* or *xsd:decimal*, and is defined as part of a key or a foreign key. A miss-classification is possible where for example a column such as `POSTCODE` defined as a *xsd:short* is not part of a key or foreign key. This can result in a mismatch if one column is included in the key or the foreign key but its matching column is not. Although degenerate dimensions are used frequently, it is considered to be a good practice to minimize their use and define them as string.

**Measure:** A column whose owning table is a *FactTable* is classified a *Measure* if one of the following is true: i) it has a data type *xsd:decimal* with the restriction *xsd:fractionDigits*, ii) it has one of the data types *xsd:short*, *xsd:integer* or *xsd:decimal* and is not defined as part of a key or a foreign key. Miss-classification of measures is possible but to a lesser extent, e.g. `no_of_cylinders` which does not appear in a key or foreign key and has a data type *xsd:short* is not a measure. This highlights a design shortcoming where non-metric data items are more accurately defined as string.

**Key:** Similar to [13], every *attribute* or *element* that appears in a key is also made an *attribute* or *element* of a *Key* for the respective dimension or fact table.

**Hierarchies:** Dimension hierarchies are not always available and as described in Section 4, they may need to be inferred. Where required, we add dimension hierarchies to the XML schemas manually but for reasons described earlier in Section 2.1 they are not used in the matching process.

**Inference of snowflaked dimensions** occurs where a column is classified as *DataAttribute* or *SurrogateKeyAttribute* and refers to a *DataAttribute* or *SurrogateKeyAttribute*. The relationship *refersTo* defines the hierarchy between the owning dimension tables.

### 3 Evaluation of StarMod in Schema Matching

Our objective is to establish that when compared to using a relational model, using a more precise representation of Star schemas can help improve their match results. We use two of the well known schema matching approaches for which there is readily available implementation.

Similarity Flooding (SF) is a well known versatile matching algorithm that calculates similarity between elements of two graphs. Properties of relational schemas can be represented as graphs in which each node represents a property and edges represent the relationship between properties. For example, tables have columns and columns have names, data types and lengths. The matching is based on the intuition that similarity between for example column names, data types and length increases the similarity between the owning columns and by the same token the similarity between two columns increases the similarity

between the owning tables. Graphs can be modified to include different set of properties and relationships and hence allows us to use the SF with a model such as StarMod. SF uses RDF statements which can be easily obtained from OWL instances of StarMod.

COMA++ is also a well known schema matching approach that takes advantage of a rich set of matchers [6]. Its latest version (2008c) which we used in our evaluation includes a library of 17 matchers ranging from element-level matchers to hybrid structural matchers. Its *MatchComposeOperation* combines match results to calculate a single new match result. We used COMA++ with its existing matchers and experimented with the same schemas we used in experimenting with SF. This helped us answer these questions: (i) How SF using both relational model and StarMod compares with COMA (i.e. COMA++)? and (ii) is there a potential for improving COMA results by using specialised relational properties such as those defined in StarMod?

### 3.1 Discussion of Match Results for Example Schemas

Using the two example schemas in Figures 1 and 3, we compare and discuss their match results suggested by SF, SF\* and COMA. For brevity we refer to SF\* when we use the StarMod, and SF when we use the relational model to describe the schemas. We calculate the accuracy of match results as follows [16]:  $Accuracy = Recall \times (2 - 1/Precision)$ . Table 1 shows the match results suggested by SF, SF\* and COMA.

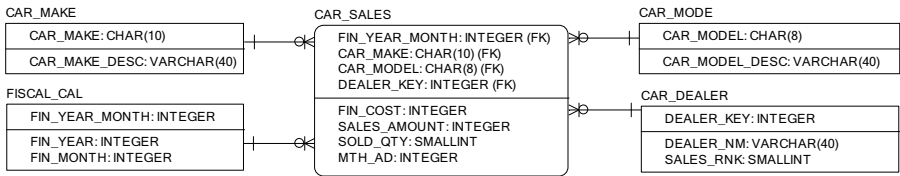


Fig. 3. The Star schema matched against the schema in Figure 1

The first two columns show the elements from the source and target schemas. Columns SF, SF\* and COMA indicate if they considered the pair to be a match. The column Experts shows the match results agreed by at least 2 of the 3 experts. Next, we compare the results of the three approaches.

**SF versus SF\*:** SF benefits from relatively limited information such as data type, column name and table name and hence it is not able to make sufficient distinction between columns with strong similarity between their names. This has led to significantly lower precision by SF and is particularly visible where columns with identical names appear in dimension and fact tables. On the other hand, stronger classification of tables, columns and relationships by SF\* has resulted in higher accuracy.

**COMA versus SF and SF\*:** COMA benefiting from a combination of matchers outperforms SF by a significant margin. When however comparing its

**Table 1.** Comparison of match results for the example schemas

Match Results - Formula 'C'		SF*	SF	COMA	Experts
Schema: pp1.xsd	Schema: pp2.xsd				
DEALER.DEALER_ID	CAR_DEALER.DEALER_KEY	✓		✓	✓
DEALER.DEALER_NAME	CAR_DEALER.DEALER_NM	✓		✓	✓
FISCAL_CAL.FISCAL_MONTH	FINANCIAL_CAL.FIN_MONTH	✓	✓	✓	✓
FISCAL_CAL.FISCAL_YEAR	FINANCIAL_CAL.FIN_YEAR	✓	✓	✓	✓
FISCAL_CAL.YEAR_MONTH	FINANCIAL_CAL.FIN_YEAR_MONTH	✓	✓	✓	✓
MAKE.MAKE_NAME	CAR_MAKE.CAR_MAKE_DESC	✓			✓
MAKE.MAKE_NAME	CAR_MAKE.CAR_MAKE			✓	
MAKE.MAKE_ID	CAR_MAKE.CAR_MAKE	✓		✓	✓
MODEL.MODEL_ID	CAR_MODEL.CAR_MODEL	✓		✓	✓
MODEL.MODEL_NAME	CAR_MODEL.CAR_MODEL_DESC	✓			✓
MONTHLY_SALES.DEALER_ID	CAR_SALES.DEALER_KEY	✓		✓	✓
MONTHLY_SALES.MAKE_ID	CAR_SALES.CAR_MAKE	✓		✓	✓
MONTHLY_SALES.MODEL_ID	CAR_SALES.CAR_MODEL	✓		✓	✓
MONTHLY_SALES.MONTHLY_ADS_COST	CAR_SALES.FIN_COST	✓		✓	
MONTHLY_SALES.SALES_AMT	CAR_SALES.SALES_AMOUNT	✓	✓	✓	✓
MONTHLY_SALES.SALES_QTY	CAR_SALES.SOLD_QTY	✓		✓	✓
MONTHLY_SALES.YEAR_MONTH	CAR_SALES.FIN_YEAR_MONTH	✓		✓	✓
DEALER	CAR_DEALER	✓	✓	✓	✓
FISCAL_CAL	FINANCIAL_CAL	✓	✓	✓	✓
MAKE	CAR_MAKE	✓	✓	✓	✓
MODEL	CAR_MODEL	✓	✓	✓	✓
MONTHLY_SALES	CAR_SALES	✓	✓	✓	✓
DEALER.DELIVERY_FINAL_COST	CAR_SALES.FIN_COST		✓	✓	
MONTHLY_SALES.SALES_QTY	CAR_DEALER.SALES_RNK		✓		
MONTHLY_SALES.MONTHLY_ADS_COST	CAR_SALES.MONTH_AD				✓
Number of accurate matches		20	9	18	
Total number of matches		21	11	20	21
Accuracy		0.90	0.33	0.76	

results with SF\*, we find that the improved SF\* competes well with COMA. The false positive match between MONTHLY\_SALES.MONTHLY\_ADS\_COST and CAR\_SALES.FIN\_COST returned by SF\* is explained by two factors. Their similarity is increased by the fact that they are both classified as *measures*, this prevented a false positive match between CAR\_SALES.FIN\_COST a *measure*, and DEALER.DELIVERY\_FINAL\_COST a *DataAttribute*. Both CAR\_SALES.FIN\_COST and CAR\_SALES.MTH\_AD competing to match against MONTHLY\_SALES.MONTHLY\_ADS\_COST are *measures* but the first has a much stronger similarity of name with the target.

We also observe that SF\* has better *recall* and *precision* than COMA, however, there are instances where COMA's result could be improved by using specialization, e.g. the false negative case between MONTHLY\_SALES.MONTHLY\_ADS\_COST and CAR\_SALES.FIN\_COST could be prevented if the former was classified as a *Measure* and the latter as a *DataAttribute*. In the next section, we validate our findings by using a larger collection of Star and non-Star schemas.

### 3.2 Evaluation of Using StarMod in Matching Schemas on a Larger Scale

We used 18 pairs of schemas, 14 of which were based on those collected from text books, industry and internet resources of which 8 pairs were Star schemas and 6 pairs were non-Star schemas. We also included our running example and the

3 less complex relational schemas used by Melnik et al. [16] in their evaluation. Please see <http://goanna.cs.rmit.edu.au/~kasheh/Star>.

Table 2 includes two sub-tables showing the results of our experiments against non-Star and Star schemas. In each sub-table, the first column shows the schema pair. The next three columns show the accuracy measures returned from SF, SF\* and COMA. Negative accuracy scores are due to the *precision* being below 50%.

**Table 2.** Accuracy measures for schemas used in the evaluation

Non-Star schemas				Star schemas			
Schema Pair	SF	SF*	COMA	Schema Pair	SF	SF*	COMA
M7L, M7R	1	1	0.80	PP1, PP2	0.33	<b>0.90</b>	0.76
M8L, M9R	0.30	0.30	<b>0.50</b>	T01A, T01B	0.63	<b>0.83</b>	<b>0.83</b>
R05, R05A	0.37	0.13	<b>0.62</b>	T07A, T09A	0	<b>0.33</b>	-0.17
M8L, M8R	0.53	<b>0.71</b>	0.50	A01, A02	0.35	<b>0.68</b>	0.61
R01, R02	0.24	<b>0.52</b>	<b>0.52</b>	T02A, T02B	-0.3	<b>0.08</b>	0
R03, R06	0.23	<b>0.46</b>	0.23	T10, T11	0.33	<b>0.43</b>	0.43
R06, R07	<b>0.28</b>	0.21	0.07	T05A, T05B	0	<b>0.10</b>	0
R07, R03	0.67	0.72	<b>0.88</b>	T06B, T04	-0.56	<b>-0.22</b>	-0.44
R08A, R08B	0.35	0.23	<b>0.52</b>	T07B, T11	0.31	<b>0.41</b>	0.22

We now compare the three approaches for Star and non-Star based schemas. **SF versus SF\***: In respect to the schema pairs involving Star schemas, the match results show consistent improvement for SF\* over the SF. The results are also consistent with our observations in the running example. In respect to the schema pairs involving non-Star schemas, the results are mixed with an overall similar performance for using SF or SF\*. A closer examination of the schemas shows that SF performs better than SF\* where the topology of tables between the two schemas are very different. This difference (as expected) results in a mismatch between how properties are classified. Conversely, SF\* performs better than SF where there are reference tables (resembling dimension tables) and where the structure of tables across the two schemas are similar.

**SF\* versus COMA**: In respect to the schema pairs involving Star schemas, we find that SF\* accuracy is higher (by a smaller margin) or the same as COMA. The results are consistent with our findings from the running example showing that COMA's relatively higher number of false positive cases can be reduced by using a more precise classification of relational properties. In respect to the schema pairs involving non-Star schemas, as with SF versus SF\*, the results are again mixed with an overall similar performance for both.

**SF versus COMA**: For Star and non-Star schemas, COMA has an overall better results than SF stemming from higher cases of true positives for COMA which is even stronger against Star schemas. This indicates that COMA's matchers appear to make good use of the similarity between table structures.

These results support our hypothesis that using a more precise description of the Star schemas improves their match results and can be also effective for arbitrary relational schemas.

## 4 Related Work

Representation of multidimensional data using ER diagrams has received considerable attention in the past decade [5], [7], [11]. These works do not suggest a meta model and leave out properties such as surrogate keys, degenerate facts, and degenerate dimensions.

Abelló et al. [2] present a meta model (YAM2) for multidimensional data by extending UML stereotypes. Their model captures facts, dimensions, levels, measures and summarizations. The Star schema described using this model is designed from the user requirements and omits representation of important properties such as snowflaked dimensions, degenerate dimensions and facts.

Common Warehouse Metamodel (CWM) provides a metamodel for a wide range of areas of data warehouses including multidimensional resources in several languages including UML [1]. StarMod at its highest level of abstraction can be compared with CWM's metamodel for multidimensional data.

Luján-Mora et al. [15] propose a conceptual model using UML with a wider coverage of properties of multidimensional data than YAM2. It covers a comprehensive set of properties including hierarchies, degenerate dimensions, and degenerate facts. It is however designed to be aligned with the logical or conceptual model and as such it excludes physical properties such as keys, foreign key relationships and snowflaked dimensions. StarMod covers these concepts but is more aligned with the physical implementation of Star schemas and is designed with a view to its application in schema matching.

Matching relational schemas is a well researched area, a summary of different approaches is provided by Pavel and Euzenat [17]. Many of the concepts used in matching relational schemas are also applicable to Star schemas, but we are concerned with the research that exploits Star schema properties.

Li and Yang [14] discuss matching Star schemas specifically. Their approach converts schemas to a binary schema tree with the fact table as the root of the tree and dimensions form the child paths to the root node. The proposed matching uses linguistic properties and fixed similarity values for different combinations of data types. It recognizes properties of multidimensional data but only as far as dimensions, dimension hierarchies and facts.

The matching algorithm used by Marko et al. [3] recognises the value in matching dimensions, facts, levels, measures and attributes. The authors do not describe the model they use and how the Star properties used in the matching process were obtained. Moreover, the matching process excludes properties such as keys, degenerate dimensions and degenerate facts. The distinction with their approach is said to be the treatment of aggregation hierarchies.

## 5 Conclusion and Future Work

We introduced StarMod as a more precise representation of Star schemas. We used UML to visualize StarMod and implemented it using OWL to assist in the schema matching process. We described the inference of Star properties from

relational schemas and their applications in matching Star schemas. Two well known schema matching algorithms were used to measure and compare improvement in accuracy of schema matching by using StarMod. We demonstrated that a more precise model such as StarMod can improve the accuracy of match results for Star schemas and that it could be also effective against arbitrary non-Star relational schemas. As part of our future work, we intend to use domain ontologies with OWL reasoning to improve the inference of Star properties such as degenerate dimensions and facts which we were unable to infer. As further work, it would be interesting to extend COMA to recognise Star properties to improve its accuracy for Star schemas.

## References

1. <http://www.omg.org/spec/CWM/> (2003)
2. Abelló, A., Samos, J., Saltor, F.: YAM2: a multidimensional conceptual model extending UML. *Information Systems* 31, 541–567 (2005)
3. Banek, M., Vrdoljak, B., Tjoa, M.: Automating the schema matching process for heterogeneous data warehouses. *Int. J. of Data Warehousing and Mining* 4(4), 1–21 (2008)
4. Bernstein, P., Melnik, S., Petropoulos, M., Quix, C.: Industrial-strength schema matching. *SIGMOD Record* 33(4), 38–43 (2004)
5. Chen, Y.T., Hsu, P.Y.: A grain preservation translation algorithm: From ER diagram to multidimensional model. *Inf. Sci.* 177, 3679–3695 (2007)
6. Do, H.H., Rahm, E.: COMA: a system for flexible combination of schema matching approaches. In: *Proceedings of the 28th VLDB*, pp. 610–621 (2002)
7. Franconi, E., Sattler, U.: A datawarehouse conceptual datamodel for multidimensional aggregation: a preliminary report. In: *Proceedings of the Workshop on Design and Management of Datawarehouses (DMDW 1999)*, Heidelberg, Germany (1999)
8. Giovinazzo, W.: *Object Oriented Data Warehouse Design, Building a Star Schema*. Prentice Hall, Inc., New Jersey (2000)
9. Haas, L., Hernández, M., Ho, H., Popa, L., Roth, M.: Clio grows up: from research prototype to industrial tool. In: *Proceedings of ACM SIGMOD*, pp. 805–810 (2005)
10. Imhoff, C., Galembo, N., Geiger, J.: *Mastering Data Warehouse Design*. Wiley Publishing, Inc., Indianapolis (2003)
11. Kamble, A.S.: A conceptual model for multidimensional data. In: *Proceedings of the Fifth Asia-Pacific conference on Conceptual Modelling*, pp. 29–38 (2008)
12. Kimball, R., Ross, M.: *The Data Warehouse Toolkit*. Wiley Publishing, Inc., Indianapolis (2000)
13. de Laborda, C.P., Conrad, S.: Relational.OWL: a data and schema representation format based on OWL. In: *Proceedings of the 2nd Asia-Pacific Conference on Conceptual Modelling*, pp. 89–96 (2005)
14. Li, L., Yang, L.: Automatic schema matching for data warehouses. In: *5th World Congress on Intelligent Control and Automation*, pp. 3939–3943 (2004)
15. Luján-Mora, S., Trujillo, J., Song, I.Y.: A UML profile for multidimensional modeling in data warehouses. *Data Knowl. Eng.* 59(3), 725–769 (2006)
16. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: *Proceedings of 18th International Conference on Data Engineering*, pp. 117–118 (2002)

17. Pavel, S., Euzenat, J.: A survey of schema-based matching approaches. Tech. rep., Informaticae Telecomunicazioni, University of Trento (2004)
18. Riazati, D., Thom, J.A., Zhang, X.: Inferring aggregation hierarchies for integration of data marts. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6262, pp. 96–110. Springer, Heidelberg (2010)
19. Torlone, R.: Two approaches to the integration of heterogeneous data warehouses. *Distrib. Parallel Databases* 23(1), 69–97 (2008)

# Automated Construction of Domain Ontology Taxonomies from Wikipedia

Damir Jurić, Marko Banek, and Zoran Skočir

University of Zagreb, Faculty of Electrical Engineering and Computing,  
Unska 3, HR-10000 Zagreb, Croatia  
{damir.juric,marko.banek,zoran.skocir}@fer.hr

**Abstract.** The key step for implementing the idea of the Semantic Web into a feasible system is providing a variety of domain ontologies that are constructed on demand, in an automated manner and in a very short time. In this paper we introduce an unsupervised method for constructing domain ontology taxonomies from Wikipedia. The benefit of using Wikipedia as the source is twofold: first, the Wikipedia articles are concise and have a particularly high “density” of domain knowledge; second, the articles represent a consensus of a large community, thus avoiding term disagreements and misinterpretations. The taxonomy construction algorithm, aimed at finding the subsumption relation, is based on two different techniques, which both apply linguistic parsing: analyzing the first sentence of each Wikipedia article and processing the categories associated with the article. The method has been evaluated against human judgment for two independent domains and the experimental results have proven its robustness and high precision.

## 1 Introduction

The key step for implementing the idea of the Semantic Web into a feasible system is the development of methods for automated, instantaneous construction of domain ontologies. Most of the existing techniques for ontology construction exploit Web documents as their source [3,4,12], which often leads to disagreements in understanding and interpreting the ontology content between the creators of the documents and the ontology users. These disagreements can be reduced to the lowest possible minimum only if the sources for the automated construction express a general consensus of many people.

Wikipedia [14], a publicly available online encyclopedia, with its 3,500,000 articles in English as of February 2011, is a consensus of the largest world community engaged in collecting and classifying knowledge. Since it is an encyclopedia, its articles are concise and have a high “density” of domain knowledge. This makes a contrast to Web documents in general, where only a minority of the automatically captured content (classes, instances, relations) is relevant for the domain of interest.

The goal of our research is to develop a set of unsupervised techniques that will take one term (concept) at the input and, using Wikipedia, WordNet [6]



and some additional publicly available linguistic tools, create the domain ontology consisting of concepts and properly named relations surrounding the input term. In our previous work [1] we have developed techniques for learning not only binary but also n-ary relations from Wikipedia articles in order to populate domain ontologies. In this paper we present an unsupervised method for constructing the taxonomy structure of domain ontologies automatically. We construct the taxonomy of a domain ontology by combining two different techniques. Firstly, we exploit the organization of Wikipedia articles into categories. Secondly, we analyze the text of the articles linguistically. To the best of our knowledge, none of the existing techniques for taxonomy construction based on Wikipedia has successfully merged these two approaches.

The paper is structured as follows. The related work is outlined in Section 2. Section 3 illustrates the mechanism for defining the coverage of the target domain ontology, given a single input definition term. Section 4 presents the algorithm for taxonomy construction, which consists of two phases. Its first phase, the anchor class computation algorithm, is explained and experimentally evaluated in Section 5. Section 6 describes the second phase of the algorithm, the taxonomy construction based on the anchor class. In Section 7 conclusions are drawn.

## 2 Related Work

Various machine learning methods have been developed to perform different tasks related to ontology construction [5,9]. However, learning by induction is not in accordance with the very notion of ontologies as sets of declarative logical statements [2]. Newer techniques overcome the stated contradiction by performing linguistic analysis of the source text in order to find relations in general (each sentence is a statement and natural language has a logic of expressing facts, although complicated and subtle) [1,3]. On the other hand, particular relations like the subsumption (*is-a*) relation can be found by pattern analysis [7,12].

Since the positioning of Wikipedia as the most important online encyclopedia several approaches were developed to exploit it for taxonomy construction. Yago [13] is a general ontology (hence not a domain ontology, which is our goal) where the class taxonomy is based on WordNet (each WordNet synset becomes a class of Yago), while the instances are collected titles of various Wikipedia articles. The names of the categories to which these articles belong are analyzed using a shallow linguistic parser and a stemmer in order to find a correspondent WordNet term. A taxonomy based exclusively on the category hierarchy of Wikipedia has been derived in [11,15] by exploiting six different techniques that include pattern analysis, linguistic parsing and string matching.

## 3 Domain Definition Term and Domain Coverage

Our approach for automated domain ontology construction from Wikipedia starts with taking a single term, the domain definition term *domainName*, as input. We can produce an output domain ontology if one or more categories

containing *domainName* in their names (including a category named exactly *domainName*) exist in Wikipedia. We initially create an empty list of articles. We add to the list all articles that are members of a category containing *domainName* in its name. If the category has any subcategories, the procedure is performed recursively for each of the subcategories.

Articles in the list are the *core terms* of the ontology and they become either its instances or classes (the distinction between classes and instances is described in [15]). For each of those articles we define a taxonomic structure. All taxonomic structures share a common node and thus are part of the single unique taxonomic tree of the domain ontology. The future domain ontology also contains *additional terms*, which are attached to the ontology by forming a relation with a core term. Additional terms may be Wikipedia titles or common words (WordNet synsets) and are possible connections with other domain ontologies. We do not provide taxonomic structures for the additional terms, respecting the decision of the authors of Wikipedia not to put them in the domain category.

## 4 Taxonomy Construction Algorithm

Each article in Wikipedia can belong to multiple categories that describe the domains related to the article. Categories themselves may have subcategories, or reversely, supercategories. However, the categories only form a thematically organized thesaurus, which may imply relations other than subsumption: ALEXANDER THE GREAT (Fig. 1) was one of MACEDONIAN MONARCHS, MONARCHS OF PERSIA and PHARAOHS OF THE ARGEAD DYNASTY, but not a HELLENISTIC RULER CULT (he was a divinity of such a cult). Ponzetto and Strube [1] developed a set of techniques which determine whether a subcategory-supercategory relation is a subsumption or not.

Categories: Alexander the Great | 356 BC births | 323 BC deaths | 4th-century BC Greek people | 4th-century BC Macedonians | 4th-century BC rulers | Ancient Macedonian generals | Ancient Pellaeans | City founders | Deified people | Greek historical hero cult | Hellenistic individuals | Hellenistic ruler cult | Macedonian monarchs | Monarchs of Persia | Pharaohs of the Argead dynasty

**Fig. 1.** Categories referenced by the article ALEXANDER THE GREAT

Since we construct domain ontologies, we believe that their taxonomies should be much less complicated than the taxonomy of WordNet or Yago. The taxonomy construction algorithm, applied for each core term (i.e. each Wikipedia article) consists of two basic steps:

1. finding a WordNet entry that subsumes the core term (the WordNet entry becomes the class of the future domain ontology and is called the *anchor* class of the core term),
2. associating the categories of the core term article with WordNet entries and exploring which of them form a taxonomic relation with the anchor class.

The first step of the algorithm will be explained in Section 5, and the second step in Section 6.

## 5 Anchor Class Computation

### 5.1 First Sentence Algorithm

A major novelty of our taxonomy construction algorithm is to perform a linguistic analysis of the first sentence in each Wikipedia article. By the firm conventions of Wikipedia, the first sentence contains the title term of the article or its synonym in bold letters (Fig. 2) and tries to put the title term into a broader context. In the largest majority of cases, the sentence contains the verb *be* and hence expresses subsumption: ALEXANDER THE GREAT is a *king* (Fig. 2). Thus, *king* is the anchor class for the core term (instance) ALEXANDER THE GREAT.

Alexander III of Macedon (356–323 BC), popularly known as Alexander the Great (Greek: Μέγας Αλέξανδρος, Mégas Aléxandros), was a Greek king (basileus) of Macedon. He is the most celebrated member of the Argead

Fig. 2. First sentence of the article ALEXANDER THE GREAT

The first sentence of each article contains a subsumption relation if its subject is the core term, if its main verb is *be* (in particular: *is*, *are*, *was*, *were*), and if the main verb links the subject to a noun that corresponds to a WordNet class (the verb *be* serving as the main verb of the sentence is called *copula*: the Stanford parser [10] identifies a *cop* dependency between the respective form of the verb *be* and the noun). The latter noun becomes the anchor class. If the noun corresponds to more than one meaning in WordNet, sense disambiguation must be performed (these issues will not be discussed due to page limitation). We call the presented algorithm the *First Sentence Algorithm* (FSA).

The first experimental evaluation of FSA showed that it works with a satisfactory precision. However, since in many cases either the proposed anchor class did not exist in WordNet or the first sentence did not contain the verb *be* at all, we had to develop a second algorithm, which, working jointly with FSA, would produce highly satisfactory results.

### 5.2 Category Head Algorithm

The second algorithm is based on analyzing the lexical heads of the categories related to the article and is called the *Category Head Algorithm* (CHA). The *lexical head* of a noun phrase (the category names are noun phrases) is the noun which all other words in the phrase describe (e.g. the head of the category ANCIENT MACEDONIAN GENERALS is *generals*, while the head of the category PHAROHS OF THE ARGEAD DYNASTY is *pharaohs*). The algorithm follows the notion that the category heads appearing only once for a particular article may not necessarily represent the *is-a* relation (as in case of *birth*, *death* or *cult* in the article ALEXANDER THE GREAT, see Fig. 1), but those appearing twice or more almost always should. *Monarch* is the only category head appearing twice

for Alexander the Great and is thus suggested as the anchor class; it is a direct WordNet hypernym of *king*, which is proposed by FSA.

In cases when more than a single head term appears twice or more frequently, we sort all the category heads in accordance with their frequency. The most frequent category head becomes the anchor class only if its frequency is at least twice as large as the second largest frequency and if it appears as a noun entry in WordNet (if there is more than one meaning for the entry, word sense disambiguation is performed).

In case when the category list of an input article consists of a single member CHA outputs the head of that single category (experiments in Section 5.3 prove that CHA works successfully even in that case).

### 5.3 Experimental Evaluation and Combined Algorithm as Solution

We tested in parallel the two algorithms for computing the anchor class on two different domains: *Alexander the Great* (the entire domain, 461 articles) and *Zagreb* (randomly selected 268 articles out of a total of 1461). The performance of the algorithms was compared with the human judgment. Two human evaluators first had to agree on the anchor class. The few cases when no agreement could be reached between the evaluators were cast away. An automatically produced anchor class was held as correct only when its corresponding WordNet synset was identical to the synset produced by the evaluators.

Let  $T$  be the number of correctly computed anchor classes. Let  $F$  be the number of falsely computed anchor classes. Let  $N$  be the number of cases when the algorithm can produce no anchor class as output (either if the first sentence of the article contains no copula, if none of the category heads appears twice more frequently than the others, or if the extracted noun is not in WordNet e.g. *somatophylax* or *phourarch*, which denote a bodyguard of the King of Macedon and a Greek military title, respectively).

Precision  $P$  is defined in the standard fashion, as the ratio between the number of correctly computed anchor classes and the total number of anchor classes computed by the algorithm i.e.  $P = T/(T + F)$ . Since our task is not a classification, we cannot calculate recall. Instead, we define two measures that take into account both the precision of the algorithm and its inability to produce an anchor class in certain cases. The first measure,  $A$ , *ability to produce an output*, is the ratio of the number of the input articles for which the algorithm can produce a result (either a correct or false one) and the total number of input articles:  $A = (T + F)/(T + F + N)$ . The second measure is called *pseudo-F-measure (PFM)*. While the real F-measure deals with the true positive, the false positive and the false negative incomes of a classification task, our measure replaces the false negatives with cases when the algorithm is unable to perform the computation ( $N$ ).

$$PFM_{\beta} = \frac{(1 + \beta^2) \cdot T}{(1 + \beta^2) \cdot T + \beta^2 \cdot N + F} \quad (1)$$

In case when  $\beta = 1$ , incorrectly computed anchor classes are considered of an equal importance as the cases when no anchor class can be computed at all:  $PFM_1 = 2T/(2T + N + F)$ . If precision is held more important (i.e. we prefer smaller  $F$  at the expense of larger  $N$ ),  $\beta$  is smaller than 1.

The experiments (Table 1, Table 2) show that CHA is more precise while FSA produces an output for a larger portion of the input articles (see the first two rows in Table 2). The highest values of  $A$  and  $PFM$  are achieved when both algorithms are applied sequentially. Technically, we always apply both algorithms, but in case when their output is different, the output of the algorithm declared as the “first” becomes the anchor class. The question which sequence of the algorithms works better can be decided by observing the values of  $PFM$  (the order of algorithms does not influence  $A$ ). For the *Zagreb* domain the value of  $PFM$  is almost equal for both orders. However, the combination when CHA is applied firstly and FSA secondly produces significantly better results for the domain *Alexander the Great*. Hence, we propose a combined algorithm where CHA prevails over FSA if both produce output terms that (1) are not identical, (2) are not a pair of WordNet synonyms, and (3) one is not a hyponym of the other in WordNet. If the output term of one algorithm is a hyponym of the other output term, the hyponym becomes the anchor class, (we prefer an anchor class to be as specific as possible). For instance, for the article ALEXANDER THE GREAT CHA outputs *monarch*, while FSA outputs *king*. Since *king* is a hyponym of *monarch*, it becomes the anchor class. We call the combined algorithm CHA-FSA.

**Table 1.** Parallel evaluation of FSA and CHA: measures  $T$ ,  $F$  and  $N$

<b>Alexander</b>	CHA T	CHA N	CHA F	<b>Zagreb</b>	CHA T	CHA N	CHA F
FSA T	131	55	36	FSA T	119	5	6
FSA N	120	82	30	FSA N	101	7	18
FSA F	4	2	1	FSA F	8	2	2

**Table 2.** Parallel evaluation of FSA and CHA: measures  $P$ ,  $A$  and  $PFM$

	<b>Alexander the Great</b>				<b>Zagreb</b>			
	<b>P</b>	<b>A</b>	<b>PFM<sub>1</sub></b>	<b>PFM<sub>0.5</sub></b>	<b>P</b>	<b>A</b>	<b>PFM<sub>1</sub></b>	<b>PFM<sub>0.5</sub></b>
<b>FSA only</b>	0.7919	0.6985	0.7123	0.7580	0.8976	0.9478	0.9194	0.9062
<b>CHA only</b>	0.9694	0.4967	0.6501	0.8102	0.9155	0.5299	0.6533	0.7888
<b>Both, FSA first</b>	0.8179	0.8221	0.8042	0.8124	0.8927	0.9739	0.9301	0.9073
<b>Both, CHA first</b>	0.9024	0.8221	0.8518	0.8814	0.8851	0.9739	0.9259	0.9009

## 6 Finding Subsumption Relations between Core Term Categories and Anchor Class

Once the CHA-FSA algorithm has computed the anchor class of an ontology core term and associated it with a WordNet synset, we arrange the remaining category

heads into a taxonomy based on the anchor class. The taxonomy is a small subset of WordNet: each of its subsumption relations exists in WordNet, either directly or transitively. We do not copy the entire WordNet taxonomy with the anchor class as the lowest node, since most of its vertices would be irrelevant for the domain. For instance, given the anchor class *general* in the domain *Alexander the Great*, its relevant hypernyms are *officer* and *person* (which appear as category heads for at least some of the generals), but certainly not *commissioned military officer*, *skilled worker* or *organism*.

We find the lexical heads of all categories referred by an article corresponding to nouns in WordNet. We calculate Lin’s linguistic similarity in WordNet [8] between each category head and the anchor class. If the similarity is greater than or equal to an empirically determined threshold  $\alpha$ , and if the category head is also a WordNet hypernym of the anchor class, we insert it as a hypernym in our domain ontology taxonomy.

With the threshold  $\alpha$  set to 0.45 (resulting from our experiments), 5 of the 11 category heads of the article ALEXANDER THE GREAT make the similarity higher than  $\alpha$  (Table 3): *general*, *individual*, *monarch*, *pharaoh*, *ruler*. *King* is a WordNet hyponym of *monarch*, *monarch* a hyponym of *ruler* and *ruler* a hyponym of *person/individual*. On the other hand, the nearest common ancestor for *king* and *pharaoh* is *ruler* while the one for *king* and *general* is *person/individual*. Hence, a class taxonomy  $king \rightarrow monarch \rightarrow ruler \rightarrow person/individual$  is created (Fig. 3, hypernymy is represented with solid lines), while classes *pharaoh* and *general* are not taxonomy members of the core term ALEXANDER THE GREAT.

**Table 3.** Similarity between the anchor class *king* and the category heads for the article ALEXANDER THE GREAT

birth	cult	death	founder	general	individual	Maced.	monarch	people	pharaoh	ruler
0.0	0.0	0.3593	0.2731	0.5178	0.4665	0.2274	0.9763	0.0	0.6888	0.9524



**Fig. 3.** Taxonomic structure for the article (individual) ALEXANDER THE GREAT

## 7 Conclusion

In this paper we have presented an unsupervised approach for constructing domain ontology taxonomies from Wikipedia. Each ontology instance (corresponding to the title of a Wikipedia article) is associated with a class by applying two

different algorithms. The *First Sentence Algorithm* linguistically analyzes the first sentence of the corresponding Wikipedia article, while the *Category Head Algorithm* processes the lexical heads of Wikipedia categories associated with the same article. The resulting anchor class is the combined output of both algorithms. Once having computed the anchor class, we enrich the taxonomy with WordNet synset classes corresponding to the lexical heads of the categories if the latter are semantically similar to the anchor class. The experimental evaluation against human judgment for two independent domains proves that the combination of the two algorithms is a robust, highly precise and applicable solution for the largest majority of Wikipedia articles.

## References

1. Banek, M., Jurić, D., Skočir, Z.: Learning semantic n-ary relations from Wikipedia. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6261, pp. 470–477. Springer, Heidelberg (2010)
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. Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. *J. Art. Int. Research* 24, 305–339 (2005)
6. Fellbaum, C. (ed.): WordNet. An electronic lexical database. MIT Press, Cambridge (1998)
7. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proc. COLING, pp. 539–545 (1992)
8. Lin, D.: An information-theoretic definition of similarity. In: Pr. ICML, pp. 296–304 (1998)
9. Maedche, A., Staab, S.: Discovering conceptual relations from text. In: Proc. ECAI, pp. 321–325 (2000)
10. de Marneffe, C.-M., MacCartney, B., Manning, C.D.: Generating typed dependency parses from phrase structure parses. In: Proc. LREC, pp. 449–454 (2006)
11. Ponzetto, S.P., Strube, M.: Deriving a large-scale taxonomy from Wikipedia. In: Proc. AACL, pp. 1440–1445 (2007)
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, G., Weikum, G.: YAGO - A large ontology from Wikipedia and WordNet. *J. Web Semantics* 6(3), 203–217 (2008)
14. Wikipedia, <http://en.wikipedia.org> (retrieved February 12, 2011)
15. Zirn, C., Nastase, V., Strube, M.: Distinguishing between instances and classes in the Wikipedia taxonomy. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 376–387. Springer, Heidelberg (2008)

# Storing Fuzzy Ontology in Fuzzy Relational Database

Fu Zhang<sup>1</sup>, Z.M. Ma<sup>1</sup>, Li Yan<sup>2</sup>, and Jingwei Cheng<sup>1</sup>

<sup>1</sup> College of Information Science & Engineering, Northeastern University,  
Shenyang, 110819, China

<sup>2</sup> School of Software, Northeastern University, Shenyang, 110819, China  
mazongmin@ise.neu.edu.cn

**Abstract.** Information imprecision and uncertainty exist in many real-world applications and for this reason fuzzy ontologies have been extensively investigated and increasingly created. Therefore, it is critical to develop scalable and efficient fuzzy ontology storage mechanism. The fuzzy relational database may be a good candidate for storing fuzzy ontologies because of the widespread use and mature techniques. In this paper, we propose an approach for storing fuzzy ontologies in fuzzy relational databases. The elements of fuzzy ontologies are introduced first, where most of constructors of fuzzy ontologies are considered. On this basis, we propose an approach for storing all of these elements of fuzzy ontologies in fuzzy relational databases, and an example is provided throughout the paper to well explain the approach.

**Keywords:** Fuzzy ontology, Fuzzy relational database, Storage.

## 1 Introduction

The Semantic Web aims at extending the current Web so that the content of web pages is described with rich semantics and will not exclusively be meaningful for humans, but also be machine processable [2], [3]. Ontologies, as meaning providers, which enable a shared, explicit and formal description of the domain knowledge, have been recognized to play an important role in the Semantic Web and many application domains [3]. Currently, the Semantic Web is increasingly gaining popularity and importance with an increasing number of people using ontologies to represent their information. Lots of ontologies have been created and real ontologies tend to become very large to huge (millions of items) [2]. Therefore, one problem is considered that has arisen from practical needs: namely, possibilities for storing ontology information. In particular, relational databases, which can provide maturity, reliability and availability, may be the widest model used in the world for storing the information of domains [20]. On this basis, some proposals have been developed to store ontologies in relational databases [1], [2], [7], [8], [20].

However, the classical ontologies and the relevant ontology storage techniques may not be sufficient to represent and store fuzzy information that is commonly found in many application domains. Since the early 1980's, the representation of fuzzy information with fuzzy set theory has been addressed by Zadeh [21]. Over the years, fuzzy information is apparent in many real life application domains, such as databases, information systems, the Semantic Web and many more [9], [17].



Therefore, it is not surprising that the classical ontologies need to be extended for handling fuzzy information in various Semantic Web applications. A possible solution to tackle this problem is to incorporate fuzzy logic into ontologies. For this purpose, lots of proposals have been developed to represent and handle fuzzy information in ontologies and a great number of fuzzy ontologies for different domains were created [4], [6], [10], [12], [14], [16], [18], [19], [22].

With the increasing use of fuzzy ontologies in the Semantic Web and other application domains, fuzzy ontology based systems are growing in scope and volume. Therefore, the efficient storage of fuzzy ontologies is of paramount importance. To this end, being similar to the most common proposals that use relational databases to store ontologies, the fuzzy relational database may be also a good candidate for storing fuzzy ontologies because of the widespread use and mature techniques. If fuzzy ontologies can be stored in the fuzzy relational database, the fuzzy relational database may solve the scalability issue raised by real fuzzy ontologies and also may facilitate for retrieving and manipulating the information of fuzzy ontologies within the existing fuzzy relational database techniques.

In this paper, as an attempt to resolve the storage problem of fuzzy ontologies, we propose an approach for storing fuzzy ontologies in fuzzy relational databases. In our approach, the elements in fuzzy ontologies such as fuzzy classes, fuzzy properties, characters and restrictions of properties (e.g., functional, allValuesFrom, and etc.), data instances, and fuzzy axioms are considered. An example is provided throughout the paper to well explain the approach.

The remainder of this paper is organized as follows. Section 2 introduces fuzzy ontologies. Section 3 investigates how to store fuzzy ontologies in fuzzy relational databases. Section 4 shows the conclusions and further work.

## 2 A Quick Look to Fuzzy Ontologies

The following briefly introduces some main elements of fuzzy ontologies. All of these elements will be stored in fuzzy relational databases as will be presented in Section 3.

Ontology, which enables a shared, formal, explicit and common description of domain knowledge, is represented by ontology definition languages such as RDFS, OIL, DAML+OIL, or OWL [3]. The current Semantic Web standard ontology language is OWL (Web Ontology Language), which consists of three sub-languages of increasing expressive power: OWL Lite, OWL DL and OWL Full [3].

A fuzzy ontology, which is simply an ontology that uses fuzzy logic to provide a natural representation of imprecise and uncertain knowledge, may be usually represented by fuzzy OWL language (a fuzzy extension of OWL) [4], [6]. In general, a fuzzy ontology  $FO$  consists of the fuzzy ontology structure  $FO_S$  and the fuzzy ontology instance  $FO_I$  associated with the fuzzy ontology structure.  $FO_S$  is a set of *fuzzy class descriptions* and *fuzzy class/property axioms*, and  $FO_I$  is a set of *fuzzy individual axioms*. Both of the  $FO_S$  and  $FO_I$  are defined over the classes, object properties, datatype properties, and individuals. Table 1 gives the elements of  $FO_S$  and  $FO_I$ , including *fuzzy class descriptions* and *fuzzy class/property/individual axioms*. A more detailed introduction about fuzzy ontologies can be found in [4], [18].

**Table 1.** The elements of fuzzy ontologies and the Description Logic (DL) syntax

Fuzzy OWL Abstract Syntax	DL Syntax	Fuzzy OWL Abstract Syntax	DL Syntax
<b>Fuzzy class and property descriptions</b>		<b>Fuzzy property axioms</b>	
Class ( $A$ )	$A$	DatatypeProperty ( $U$ )	
owl:Thing	$\top$	domain( $C_1$ )...domain( $C_m$ )	$\geq 1 U \sqsubseteq C_i$
owl:Nothing	$\perp$	range( $D_1$ )...range( $D_k$ )	$\top \sqsubseteq \forall U.D_i$
intersectionOf( $C_1 \dots C_n$ )	$C_1 \sqcap \dots \sqcap C_n$	[Functional]	$\top \sqsubseteq \leq 1 U$
unionOf( $C_1 \dots C_n$ )	$C_1 \sqcup \dots \sqcup C_n$	ObjectProperty ( $R$ )	
complementOf( $C$ )	$\neg C$	domain( $C_1$ )...domain( $C_m$ )	$\geq IR \sqsubseteq C_i$
oneOf( $\{o_1 \dots o_n\}$ )	$\{o_1\} \sqcup \dots \sqcup \{o_n\}$	range( $C_1$ )...range( $C_k$ )	$\top \sqsubseteq \forall R.C_i$
restriction ( $P$ someValuesFrom( $E$ ))	$\exists P.E \quad P \in \{R, U\}$	[Functional]	$\top \sqsubseteq \leq 1 R$
restriction ( $P$ allValuesFrom( $E$ ))	$\forall P.E \quad E \in \{C, D\}$	[InverseOf( $R_0$ )]	$R = (R_0)^-$
restriction ( $P$ hasValue( $a$ ))	$\exists P.\{a\}$	[Symmetric]	$R = R^-$
restriction ( $P$ minCardinality( $n$ ))	$\geq nP$	[InverseFunctional]	$\top \sqsubseteq \leq 1 R^+$
restriction ( $P$ maxCardinality( $n$ ))	$\leq nP$	[Transitive]	$\text{Trans}(R)$
restriction ( $P$ cardinality( $n$ ))	$= nP$	SubPropertyOf( $P_1, P_2$ )	$P_1 \sqsubseteq P_2$
		EquivalentProperties( $P_1, \dots, P_n$ )	$P_1 \equiv \dots \equiv P_n$
<b>Fuzzy class axioms</b>		<b>Fuzzy individual axioms</b>	
Class ( $A$ partial $C_1 \dots C_n$ )	$A \sqsubseteq C_1 \sqcap \dots \sqcap C_n$	Individual ( $o$ type( $C_i$ ) [ $\bowtie m_i$ ]...)	$o : C_i \bowtie m_i$
Class ( $A$ complete $C_1 \dots C_n$ )	$A \equiv C_1 \sqcap \dots \sqcap C_n$	value( $R_i, \alpha_i$ ) [ $\bowtie k_i$ ]...	$(\alpha_i, \alpha_j) : R_i \bowtie k_i$
SubClassOf( $C_1 C_2$ )	$C_1 \sqsubseteq C_2$	value( $U_i, v_i$ ) [ $\bowtie l_i$ ]...	$(\alpha_i, v_i) : U_i \bowtie l_i$
EquivalentClasses ( $C_1 \dots C_n$ )	$C_1 \equiv \dots \equiv C_n$	SameIndividual ( $o_1 \dots o_n$ )	$o_1 = \dots = o_n$
DisjointClasses ( $C_1 \dots C_n$ )	$C_i \sqcap C_j \sqsubseteq \perp$	DifferentIndividuals ( $o_1 \dots o_n$ )	$\alpha_i \neq \alpha_j$
EnumeratedClass ( $A$ $o_1 \dots o_n$ )	$A \equiv \{o_1 \dots o_n\}$		

### 3 Fuzzy Ontology Storage in Fuzzy Relational Database

In this section, we propose an approach for storing fuzzy ontologies in fuzzy relational databases, and an example is provided throughout the paper to well explain the approach.

Fig. 1 shows a fuzzy ontology modeling parts of the reality at a university, which includes the structure information and the instance information. Here, only parts of classes, properties and individuals are shown, and for ease of understanding the fuzzy ontology is represented as a graph instead of its formal representation in Section 2.

From Fig. 1, it is shown that the fuzzy ontology contains the following elements:

- six classes: {Department, Staff, AdminStaff, AcademicStaff, Student, Course};
- four object properties: {study\_in, choosecourse, work\_in, teach};
- four datatype properties: {staffname, title, email, age}, and the domains of these properties are as follows: {xsd:string, xsd:string, fuzzy:possibilitydistribution, fuzzy:label} (The fuzzy datatypes can be found in [5], [11], [15] in detail);
- three individuals: {staffid\_110001, depid\_0206, courid\_309};
- some axioms: {subClassOf (AdminStaff, Staff), subClassOf (AcademicStaff, Staff), ObjectProperty (work\_in domain(Staff) range(Department), ...)}.

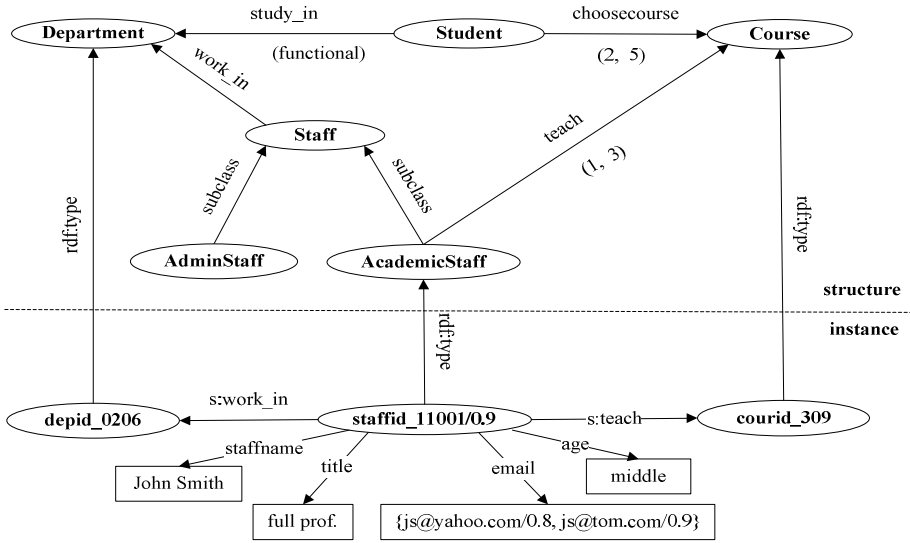


Fig. 1. A fuzzy ontology *Onto\_1* modeling parts of the reality at a university

The following several procedures will store the above fuzzy ontology structure information in the target fuzzy relational database. For simplicity, some constraints (e.g., primary key, foreign key, check, and unique) are not shown in tables of the target fuzzy relational database.

(1) Storing the resources:

Table 2 shows Resource\_Table in the target fuzzy relational database, which stores all the resources in Fig. 1, where: *OntologyName* is the fuzzy ontology name “Onto\_1”; *ID* uniquely identifies a resource; *namespace* and *localname* describe the URIref of a resource; and *type* describes the type of a resource.

Table 2. Resource\_Table

OntologyName	ID	namespace	localname	type
Onto_1	c_1	http://www...	Department	class
Onto_1	c_2	http://www...	Staff	class
Onto_1	c_3	http://www...	AdminStaff	class
Onto_1	c_4	http://www...	AcademicStaff	class
Onto_1	c_5	http://www...	Student	class
Onto_1	c_6	http://www...	Course	class
Onto_1	p_1	http://www...	study_in	property
Onto_1	p_2	http://www...	work_in	property
Onto_1	p_3	http://www...	teach	property
Onto_1	p_4	http://www...	choosecourse	property
Onto_1	p_5	http://www...	staffname	property
Onto_1	p_6	http://www...	title	property
Onto_1	p_7	http://www...	email	property
Onto_1	p_8	http://www...	age	property
Onto_1	i_1	http://www...	staffid_11001	individual
Onto_1	i_2	http://www...	depid_0206	individual
Onto_1	i_3	http://www...	courid_309	individual

**(2) Storing the relationships between classes:**

Table 3 shows Class\_Table, which stores the subclass/superclass (or unionOf, intersectionOf, complementOf, and so on, please see Table 1) relationships between two classes. Here,  $u$  denotes the membership degree of a class belonging to the subclass of another class.

For example, subClassOf (AdminStaff, Staff) is stored as the first tuple in Table 3.

**(3) Storing the domains and ranges of properties:**

Table 4 shows Property\_Field\_Table, storing domains and ranges of properties.

For example, for the object property study\_in ( $p_1$ ), its domain is Student ( $c_5$ ) and range is Department ( $c_1$ ).

**Table 3.** Class\_Table

Class <sub>1</sub> ID	Class <sub>2</sub> ID	relationship	$u$
c_3	c_2	SubClassOf	1.0
c_4	c_2	SubClassOf	1.0

**Table 4.** Property\_Field\_Table

ProID	domain	range
p_1	c_5	c_1
p_2	c_2	c_1
p_3	c_4	c_6
p_4	c_5	c_6
p_5	c_4	xsd:string
p_6	c_4	xsd:string
p_7	c_4	fuzzy: possdis
p_8	c_4	fuzzy: label

**(4) Storing the characters and restrictions of properties:**

Tables 5 and 6 show Property\_Character\_Table and Property\_Restriction\_Table, which store the characters and restrictions of properties.

For example, the functional object property study\_in can be stored in a tuple in Table 5; the cardinality constraints (2, 5) denotes that each student can choose at least 2 and at most 5 courses, which can be stored in Table 6.

**Table 5.** Property\_Character\_Table

ProID	type	character
p_1	object	Functional

**Table 6.** Property\_Restriction\_Table

ClassID	ProID	type	value
c_4	p_3	mincardinality	1
c_4	p_3	maxcardinality	3
c_5	p_4	mincardinality	2
c_5	p_4	maxcardinality	5

Since the fuzzy ontology in Fig. 1 does not contain all of the constructors in Table 1 (e.g., some class operations and property characters), their corresponding tables in the target fuzzy relational database are not shown here, and the other constructors in a fuzzy ontology (see Table 1) can be stored following the similar procedures above.

*The following several procedures will store the fuzzy ontology instance information.* From Fig. 1, there are three individuals in the fuzzy ontology as

mentioned in Resource\_Table (Table 2): staffid\_110001 ( $i_1$ ), depid\_0206 ( $i_2$ ), courid\_309 ( $i_3$ ).

**(5) Storing the relationships of individuals/classes:**

Table 7 shows Individual\_Class\_Relation\_Table in the target fuzzy relational database, which stores the relationships of individuals/classes in Fig. 1. Here,  $u$  denotes the membership degree of an individual to the class.

For example, staffid\_110001/0.9 in Fig. 1 denotes that the individual  $i_1$  is an instance of class AcademicStaff ( $c_4$ ) with degree 0.9, which is stored in Table 7.

**Table 7.** Individual\_Class\_Relation\_Table

IndID	ClassID	$u$
$i_1$	$c_4$	0.9
$i_2$	$c_1$	1.0
$i_3$	$c_6$	1.0

**Table 8.** Individual\_Crisp\_Property\_Table

IndID	ProID	value
$i_1$	$p_2$	$i_2$
$i_1$	$p_3$	$i_3$
$i_1$	$p_5$	John Smith
$i_1$	$p_6$	full prof.

**(6) Storing the values of crisp properties:**

Table 8 shows Individual\_Crisp\_Property\_Table, which stores the values of crisp properties of the individual  $i_1$ .

**(7) Storing the values of fuzzy properties:**

From Fig. 1, there are fuzzy datatypes, such as the label (*middle*) and the possibility distribution  $\{js@ya...\}$ , where the linguistic label *middle* can be represented by a function  $T[30, 40, 50, 60]$  defined on a numerical domain [15].

The following subprocedures introduce how to store these fuzzy property values.

**(7.1) Storing the fuzzy datatypes:**

Table 9 shows Fuzzy\_Types\_Table, storing the above fuzzy datatypes.

**Table 9.** Fuzzy\_Types\_Table

fuzzytypeID	fuzzytypeName
$f_1$	Label
$f_2$	Possibilitydistribution

**Table 10.** Individual\_Fuzzy\_Property\_Value\_Table

IndID	ProID	FuzzyPropertyValueID
$i_1$	$p_7$	$fpv_2$
$i_1$	$p_8$	$fpv_1$

**(7.2) Storing individuals with fuzzy properties and these fuzzy properties:**

Table 10 shows Individual\_Fuzzy\_Property\_Value\_Table, storing all those individuals with fuzzy properties and these fuzzy properties, where *FuzzyPropertyValueID* identifies the value of the fuzzy property *ProID* and it references the primary key attribute in Fuzzy\_Property\_Value\_Table (Table 11).

For example, the property age ( $p_8$ ) of the individual  $i_1$  is fuzzy and its value is identified by  $fpv_1$  which will be stored in the following Tables 11 and 12.

**(7.3) Storing the fuzzy values:**

Table 11 shows Fuzzy\_Property\_Value\_Table, storing *fuzzy typed literals* of the fuzzy property values, where *FuzzyPropertyValueID* uniquely identifies a fuzzy value, and *fuzzytypeID* denotes the datatype of a *fuzzytypeLiteral*.

Note that, since the current fuzzy database systems cannot support the fuzzy datatype storage, the field *fuzzytypeLiteral* stores the corresponding fuzzy typed literals of the fuzzy values in the form of *string* first, and then these fuzzy values will further be stored in the fuzzy database based on the approach in Table 12.

Table 12 shows Lable\_Table and PossibilityDistribution\_Table, storing all of the fuzzy property values in Fig. 1.

For example, the fuzzy typed literal *middle* is identified as *fpv\_1*, and its datatype is “*Label*” identified by *f\_1* and its value is stored in Lable\_Table.

**Table 11.** Fuzzy\_Property\_Value\_Table

FuzzyPropertyValueID	fuzzytypeLiteral	fuzzytypeID
fpv_1	middle	f_1
fpv_2	js@yahoo...	f_2

**Table 12.** Fuzzy\_Value\_Table

**Lable\_Table**

FuzzyPropertyValueID	alpha	beta	gamma	delta
fpv_1	30	40	50	60

**PossibilityDistribution\_Table**

FuzzyPropertyValueID	value <sub>1</sub>	poss <sub>1</sub>	value <sub>2</sub>	poss <sub>2</sub>
fpv_2	js@yahoo.com	0.8	js@tom.com	0.9

Until now, a fuzzy ontology, including classes, properties, individuals, axioms, datatypes, and characters and restrictions, can be stored in a fuzzy relational database. Moreover, our storage approach can be seen as a transformation. As mentioned in [13], when a transformation can preserve information capacities and it is considered as a correct transformation. Therefore, the correctness of the storage approach can be similarly proved based on the notion of information capacity [13].

**4 Conclusions and Future Work**

We have proposed an approach for storing fuzzy ontologies in fuzzy relational databases. The elements in fuzzy ontologies such as fuzzy classes, fuzzy properties, characters and restrictions of properties, data instances, and fuzzy axioms were considered, and an example was further provided throughout the paper to well explain the approach. There has not been much work in the fuzzy ontology storage and we

have tried to resolve the storage problem of fuzzy ontologies. All of these may solve the scalability issue raised by real fuzzy ontologies.

The future direction to work in this area would be to further extend the approach for storing fuzzy ontologies written in fuzzy OWL 2. Moreover, some more in-depth studies of the approach and an automated storage tool are also important directions.

**Acknowledgments.** The work is supported by the National Natural Science Foundation of China (60873010 and 61073139), the Fundamental Research Funds for the Central Universities (N090504005), and in part by the Program for New Century Excellent Talents in University (NCET-05-0288).

## References

1. Astrova, I., Korda, N., Kalja, A.: Storing OWL Ontologies in SQL Relational Databases. Proc. of World Academy of Science Engineering and Technology, 167–172 (2007)
2. Al-Jadir, L., Parent, C., Spaccapietra, S.: Reasoning with large ontologies stored in relational databases: The OntoMinD approach. *Data & Knowl. Eng.* 69(11) (2010)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American* 284(5), 34–43 (2001)
4. Bobillo, F.: Managing Vagueness in Ontologies. PhD Dissertation. University of Granada, Spain (2008)
5. Barranco, C.D., Campana, J.R., et al.: On Storing Ontologies including Fuzzy Datatypes in Relational Databases. In: Proc. of IEEE Int. Conf. on Fuzzy Systems, pp. 1–6 (2007)
6. 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)
7. Das, S., Inseok Chong, E., Eadon, G., Srinivasan, J.: Supporting ontology-based semantic matching in RDBMS. In: Proc. of the 30th VLDB Conference, pp. 1054–1065 (2004)
8. Gali, A., Chen, C.X., Claypool, K.T., Uceda-Sosa, R.: From ontology to relational databases. In: Wang, S., Tanaka, K., Zhou, S., Ling, T.-W., Guan, J., Yang, D.-q., Grandi, F., Mangina, E.E., Song, I.-Y., Mayr, H.C. (eds.) ER Workshops 2004. LNCS, vol. 3289, pp. 278–289. Springer, Heidelberg (2004)
9. Galindo, J. (ed.): Handbook of Research on Fuzzy Information Processing in Databases, pp. 55–95. Information Science Reference, Hershey (2008)
10. Inyaem, U., et al.: Construction of Fuzzy Ontology-Based Terrorism Event Extraction. In: Proc. of Knowledge Discovery & Data Mining (WKDD), pp. 391–394 (2010)
11. Lv, Y.H., Ma, Z.M., et al.: Fuzzy Ontology Storage in Fuzzy Relational Database. In: Proc. of Int. Conf. on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 242–246 (2009)
12. Lee, C.S., Jian, Z.W., et al.: A fuzzy ontology and its application to news summarization. *IEEE Transactions on Systems, Man and Cybernetics Part B* 35(5), 859–880 (2005)
13. Miller, R.J., Ioannidis, Y.E., et al.: The Use of Information Capacity in Schema Integration and Translation. In: Proc. of the 19th VLDB Conference, pp. 120–133 (1993)
14. Ma, Z.M., Yanhu, L., Yan, L.: A Fuzzy Ontology Generation Framework from Fuzzy Relational Databases. *Int. J. Semantic Web Information Systems* 4(3), 1–15 (2008)

15. Oliboni, B., Pozzani, G.: Representing Fuzzy Information by Using XML Schema. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 683–687. Springer, Heidelberg (2008)
16. Quan, T.T., Hui, S.C., et al.: Automatic fuzzy ontology generation for Semantic Web. *IEEE Transaction on Knowledge and Data Engineering* 18(6), 842–856 (2006)
17. Sanchez, E. (ed.): *Fuzzy Logic and the Semantic Web*. Elsevier, Amsterdam (2006)
18. Stoilos, G., Stamou, G., et al.: Fuzzy extensions of OWL: Logical properties and reduction to Fuzzy Description Logics. *Int. J. of Approximate Reasoning* 51, 656–679 (2010)
19. 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)
20. Vysniauskas, E., Nemuraite, L.: Transforming Ontology Representation from OWL to Relational Database. *Information Technology and Control* 35(3), 333–343 (2006)
21. Zadeh, L.A.: Fuzzy sets. *Information and Control* 8(3), 338–353 (1965)
22. Zhang, F., Ma, Z.M. et al.: Automatic Fuzzy Semantic Web Ontology Learning from Fuzzy Object-Oriented Database Model. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6261, pp. 16–30. Springer, Heidelberg (2010)



# Using an Ontology to Automatically Generate Questions for the Determination of Situations

Marten Teitsma, Jacobijn Sandberg, Marinus Maris, and Bob Wielinga

University of Amsterdam,  
Amsterdam, Netherlands

{m.teitsma, j.a.c.sandberg, m.maris, b.j.wielinga}@uva.nl

<http://www.uva.nl>

**Abstract.** We investigate whether the automatic generation of questions from an ontology leads to a trustworthy determination of a situation. With our Situation Awareness Question Generator (SAQG) we automatically generate questions from an ontology. The experiment shows that people with no previous experience can characterize hectic situations rather fast and trustworthy. When humans are participating as a sensor to gather information it is important to use basic concepts of perception and thought.

**Keywords:** Situation Awareness, Human-Centered Sensing, Ontology, Automatic Question Generation.

## 1 Introduction

The wide distribution of mobile devices makes it possible to use humans as a source of information. Especially in crisis situations, human observers may provide crucial information to evaluate the situation. Through the mobile device carried by the human observers, communication may be directly aimed at charting a particular situation as humans can answer questions about a situation. The combined answers of many observers help to refine the picture of the situation. It is in the new emerging field of Human-Centered Sensing (HCS) that the application we will propose makes use of humans as participatory sensors [5].

The study reported here focuses both on generation of relevant questions using Situation Theory [3] and on the validity of the answers provided by humans. Concerning the latter, we hypothesize that the more knowledge-intensive the task of the user, the more interpretation is needed, which will lead to variation among descriptions of the same situation.

Generally, in the field of information technology for crisis management, attention is given to support the emergency services in their crisis management task. Problems as how to synchronize information levels, facilitating decision making and communication between the services or experts have been the main focus of researchers [10]. These systems typically rely on traditional sensor systems which have stationary sensors at predefined places. HCS has several advantages when compared with such traditional sensor systems. As the sensing process makes

use of human intelligence, responses can become more versatile and adaptive than responses from a plain sensor system. In fact, the sensing process becomes an information gathering process in which the requester can lead the human "sensor" to provide the kind of information which is required. Other advantages include the power supply which is guaranteed because people use their mobile device on a daily bases for all sort of services; coverage is better because of the ubiquity and mobility of people and their devices and the adaptability is greater because humans are more flexible than traditional sensors. A problem which remains is that systems as envisioned in [5] use a central computer trying to make sense of all the data gathered from the sensors. Such a centralisation creates the problem of data or information overflow. This is a bottleneck for communication and makes the system very vulnerable [7]. The application we propose does not create this bottleneck because the information processing is distributed.

We envisage an application deployed on mobile devices which asks the user questions about a situation he finds himself in. In its core this application consists of our Situation Awareness Question Generator (SAQG) as described in section 2. The application uses an ontology to generate the questions. This ontology is sent by a server which also receives the (combined) answers. Communication between the server and application is thus spars and lightweight.

Just like the application proposed in [5] (*SafetyHeroes*), our application relies on the assumption that people are prepared to help other people. However, how many people are taking the effort to help is an open question and calls for research. Furthermore, the quality of the provided information by *SafetyHeroes* is not known. The experiment we conducted is precisely an answer to that question.

The next section introduces Situation Theory and SAQG. This is followed by a section introducing the experiment we conducted and a presentation of the results. Finally these results are evaluated in the Discussion and Conclusion section.

## 2 Situation Awareness Question Generator (SAQG)

To discuss information we use the terminology developed in Situation Theory by Devlin in [3]. In Situation Theory, a piece of information is called an infon which is formally described as a tuple of the form:

$$\langle\langle R, a_1, \dots, a_n, 0/1 \rangle\rangle \quad (1)$$

where  $R$  is a  $n$ -place relation, and  $a_1, \dots, a_n$  are variables representing objects appropriate for  $R$ . The last item is the polarity of the infon which is '1' when the infon is true given a particular situation and '0' when different from the description by the infon. A situation is defined by a minimal but sufficient set of infons.

Situation Theory is being used by Kokar [6] to create Situation Theory Ontology (STO) which is stated in Web Ontology Language (OWL). The central class in STO is *Situation*. Instances of *Situation* refer to specific situations which can become actual, i.e. correctly representing a real situation. *Situation* has therefore

various subclasses which restrict the general and abstract class further. Classes are related through OWL properties. Further specifications of situations are described in another file. In such a file, subclasses of *Situation* are specified. This division between a general ontology and a more specific, domain related, ontology makes it easy to extend the ontology.

The ontology consists of concepts which are as simple as possible. We don't want people to make use of a great volume of background knowledge to answer the questions because this knowledge differs among people. Furthermore, consideration about the right interpretation does take time and we want people to answer the questions as quickly as possible.

Visual information is categorized by Jaimes and Chang in [4] who make a distinction between percepts which refer to what human senses perceive in a physical causal relation and visual concepts which only exist due to knowledge and experience. The use of concepts at the lowest level requires very little knowledge and interpretation contrary to concepts at a higher level. The interpretation of these visual objects and their arrangement is solely based on everyday, common-sense knowledge. Moreover, the infons to describe visual objects and their constellation make use of basic concepts as defined in [9]. Concepts at the basic level of abstraction carry the most information and are most differentiated from one another. Such basic concepts can be found for objects and for events, e.g. 'car accident' which is more specific than an event with vehicles and more general than 'auto accident' which precludes trucks and other vehicles having an accident [8]. Basic level concepts are easy to understand because they are very common.

For the content of our ontology we used two sources. The first source is a traffic accident ontology proposed in [11] which is based on domain expert knowledge. Five core concepts are identified, i.e. 'time', 'location', 'weather', 'persons involved in the accident' and 'type of vehicle involved in the accident'. The second source for our ontology are two databases which gather all sorts of data about car accidents. These are the FARS (Fatality Analysis Reporting System) [1] and CARE (Community database on Accidents on the Roads in Europe) [2]. A lot of terms used as keywords in these database are basic concepts as meant in [9]. A category like 'Pavement' is immediately grasped by people and the subcategories 'Asphalt', 'Concrete' and 'Unpaved' are easily understood.

To determine the situation, SAQG is looking for subclasses of 'Situation' and their supporting infons. The infon which is most discriminative is being marked as most informative and used to generate a question. Because each infon is an informational entity, it is easy to formulate a question asking precisely for this piece of information. Each infon then has a question as a label. This label can be read when an infon is chosen as the most informative infon.

An answer to a question is a confirmation or refutation of an infon. When it is a confirmation a follow-up question is generated from the set of infons of which the confirmed infon is part. Is the infon refuted then a question is generated from the set which is the complement of the set of infons of which the refuted infon is part.

The goal of SAQG is to determine a situation which is defined with infons. When asked for the actuality of these infons a yes-no question results. In the car accident ontology a situation like 'With injury and one vehicle' is defined with two infons. In natural language these infons are stating 'people are injured' and 'one vehicle involved'. This definition generates in SAQG two questions: 'Are people injured?' and 'Is more than one vehicle involved?'. The answers to these two questions are used to determine whether this or another situation is actual according to the user of the application. When the situation is determined, additional information to obtain details about the situation may be required. Such detailed information is elicited through multiple choice questions with one or more possible answers. Not all questions are asked during each time the application is used. Because several rules are build in, some questions do not have to be asked because by implication the answer is already known. An example of such a rule is 'when the weathertype is rain then the street is wet'. When the participant has answered 'rain' to the question what kind of weather it is then the question after the condition of the street is not asked. These rules minimize the number of questions to be asked.

### 3 The Experiment

The participants of the experiment we conducted were undergraduate students in the Information Science courses at the Hogeschool van Amsterdam, University of Applied Sciences. The 89 participants are characterized by being mostly male (78.65%) and between 18 and 22 (76.41%). The participants were asked to watch one of four videos which lasted between 27 and 37 seconds. First a short introduction was given and the goal of the experiment was explained. When the participant had watched the video, questions generated by SAQG were presented.

The results were categorized in confusion matrices which were the basis for the analysis. Four measures were computed: the Matthews Correlation Coefficient (MCC) for correlation and the  $F_1$ -score for accuracy, recall and precision. The Matthews Correlation Coefficient is a measure of correlation between what is actual and what is predicted by a system or humans as in this case. It is a robust coefficient because it does not deviate when classes of different size are considered. MCC variates between -1 and +1 where -1 indicates a perfect negative correlation and +1 a perfect positive correlation, 0 indicates a random relation. The  $F_1$ -score is a measure of accuracy and varies between 0 and 1 where 0 indicates no accuracy at all and 1 a perfect accuracy. The  $F_1$ -score is the harmonic mean of the recall and precision. The recall (also called sensitivity or true positive rate) is a measure of how many of the actual situations are determined as such. Precision gives a measure of how many of the predicted situations are actually these situations.

In Table 1 the Matthews Correlation Coefficient,  $F_1$ -score, recall and precision, for the determination of situations and the yes-no questions used to determine these, is shown. The correlation is rather high, just as the accuracy. These measures are the highest when the situation 'With injury and multiple vehicles' was

**Table 1.** Matthews Correlation Coefficient,  $F_1$ -score, recall and precision for the situations

Confusion matrix	MCC	$F_1$	Recall	Precision
People are injured	0.76	0.88	0.93	0.84
Only one vehicle is involved	0.79	0.88	0.80	0.97
Mean of yes-no questions	0.78	0.88	0.87	0.91
With injury and multiple vehicles	0.85	0.89	0.83	0.95
With injury and one vehicle	0.64	0.73	0.86	0.63
Without injury and one vehicle	0.78	0.82	0.73	0.94
Without injury and multiple vehicles	0.64	0.73	0.73	0.73
Mean of situations	0.73	0.79	0.79	0.81

shown. This movie explicitly showed injured people and an accident with several cars involved. The situation 'With injury and one vehicle' was not so explicit, i.e. no one was seen to be injured and the participant had to infer by the severity of the accident that the driver and other people who might be in the car should be injured. This inference mechanism was also working the other way around when no one was injured and participants inferred the driver of the car having an accident should be injured because they interpreted the accident as being a severe accident. Another problem confronting the participants was how to interpret the concept of 'being involved' as in 'is there more than one car involved in the accident'. On the one side participants interpreted 'being involved' as 'being in the neighbourhood' or 'is part of what can be seen'. On the other side it was being interpreted as 'is the cause of the accident'. This broad interpretation and the inference from the severity of the accident to injury of the driver lead to a great number of false positives in the confusion matrix of the situation 'With injury and one vehicle'. Hence the relative low MCC value for the situations 'With injury and one vehicle' and 'Without injury and multiple vehicles'.

In Table 1 can be seen that there is a difference between the mean of all the measures for the questions and the mean of all the measures for the inferred situations. The inference of situations thus goes with a (small) loss of information.

The measures from the confusion matrices for the multiple choice questions are shown in Table 2. We have detected a group of questions for which the participants gave less trustworthy answers than the remaining group of answers. The recall value of the first group (marked in the table by \*) is 0.59 at the highest and the lowest value of the second group is 0.72. The first group is characterized as being information about numbers, clearly inferred or gathered during night and obscured vision.

When people have to count, their observation becomes less trustworthy. Two answers, one about the number of cars involved and another about the number of lanes, scored significantly less than the answers which not relied on counting, e.g. 'The road has three lanes' and 'More than four vehicles are involved' versus 'The road has one lane' and 'One vehicle is involved' (see Table 2). These answers were given to multiple choice questions where other answers than 'one lane' and 'three lanes' were possible except for the last answer which was an answer to a

**Table 2.** Summary of all the MCC's,  $F_1$ -scores, recall and precision

Confusion matrix	MCC	F1	Recall	Precision
More than four vehicles are involved	0.62	0.73	0.59*	0.96
The weather is dry	0.73	0.90	0.83	0.98
The weather is dry (without video 1)	0.84	0.94	0.91	0.98
The weather is snowy	0.88	0.91	0.91	0.91
The road is a highway	0.82	0.95	0.90	1.00
The road is in town	0.91	0.93	0.86	1.00
The road is dry	0.67	0.84	0.72	1.00
The road is dry (without video 1)	0.94	0.97	0.95	1.00
The road is covered with snow	0.88	0.91	0.88	0.96
The road has three lanes	0.44	0.69	0.52*	1.00
The road has one lane	0.91	0.93	0.96	0.91
The accident is during daylight	0.82	0.91	0.89	0.93
The accident is during the night in artificial light	0.85	0.70	0.59*	0.87
It is dawn or dusk	0.48	0.58	0.50*	0.69
Sight is unlimited	0.22	0.65	0.73	0.59
Sight is limited	0.32	0.55	0.48*	0.64
The pavement is asphalt	0.59	0.91	1.00	0.84
The pavement cannot be known	0.59	0.58	0.41*	1.00
Passenger cars and vans are involved	0.52	0.62	0.55*	0.71
Passenger cars only	0.57	0.89	0.87	0.91

yes-no question. A lot of participants used these other possible answers as their observation leading to a low recall (a lot of 'false negative'), high precision (a few 'false positive') and an overall low correlation between the actual and predicted values. The answers 'The road has one lane' and 'One vehicle is involved' both showed high recall and precision. This led to a high correlation for the 'one lane' answer indicating that such an observation was easy to make. Restrictions on the observation about 'One vehicle is involved' are given above.

When people had to infer information it became more difficult to give the right answers. Asked about the sort of pavement in video 4: *Without injury and multiple vehicles* people gave as answer 'asphalt' while the question clearly could not be answered because the road was covered with snow. The high recall for 'The pavement is asphalt' contrary to the low recall and high precision for 'The pavement cannot be known' even suggests a bias to 'The pavement is asphalt' as the answer when participants did not know what to answer.

An observation about the limits of sight was hard to make for the participants. Night or dawn as in video 1: *With injury and multiple vehicles* and video 4: *Without injury and multiple vehicles* gave participants problems to tell whether the sight was limited or not. A fence, as in video 3: *Without injury and one vehicle*, gave the same problems. Of all the MCC values the one for 'Sight is unlimited' is the lowest and for 'Sight is limited' the second lowest.

Doing observations during the night has proven to be difficult. In the Table 2 is also shown the impact of removing video 1: *With injury and multiple vehicles*

for the questions about the weathertype and whether the road was dry or not. In both cases the recall improves and precision stays the same indicating the value for 'false negative' decreases. Participants doing an observations of a nightly situation more often were wrong than participants doing an observation of a situation during daytime about whether it was dry weather and whether the road was dry.

During the experiment we also logged the time participants used to answer the questions. When the mean is taken of the different categories of questions a difference is seen for the yes-no questions for which the mean is 6.855 sec. and the multiple choice questions with one answer for which the mean is 9.34 sec. For the multiple choice questions with more possible answers the time is 17.55 sec.

## 4 Discussion and Conclusion

This experiment deviates from a real life situation in that the participants saw a video and were not really involved a car accident. But just like a real life situation only after the rapid course of events there was time for reflection and answering questions. Different from a real life situation was that the observation we asked about had to be remembered. When asked about the weather, in real life one looks (again) at the sky and answers the question. In our experiment the participants had to think back to what they had seen and try to remember. But this restriction is general and applies to all questions and so makes no difference for the relative trustworthiness.

The experiment showed the participants had more problems with questions which require some interpretation than questions which are more easy to answer. The counting of objects (vehicles or lanes) is difficult and leads to a drop in trustworthiness of answers. A question as 'How is the sight?' with possible answers 'limited' and 'unlimited' had to be interpreted. What we hoped to determine was whether mist or darkness limited sight but according to the participants a fence also limited sight. Which is true of course but says nothing about the overall sight which we were after. The concept of sight was thus not so basic and clear as we thought. For Human-Centered Sensing it is paramount to use basic concepts of perception and thought which are easy to understand, common in use and do need as little interpretation as possible.

First we conclude that it is possible to generate questions about a situation from an ontology. When used by our Situation Awareness Question Generator such an ontology has to comply to the Situation Theory Ontology as described. When it does, automatically generated questions about several situations become available.

The second conclusion we draw from this experiment is that it is possible to gather trustworthy information from people who saw a hectic situation like a car accident. It should be noted though that there are certain restrictions like when such an event is happening at night one should be more careful with this information than in broad daylight. All of the questions have a certain margin of truthfulness, i.e. the accuracy of the answers is not perfect.

Furthermore, it takes the participants less time to answer yes-no questions than multiple-choice questions. Multiple-choice questions where one has to choose more answers are the most time consuming. The difference shows that yes-no questions are preferable when a situation has to be determined very fast.

Further research shall be done with SAQG on a mobile platform. We do not expect a fundamental difference with respect to the trustworthiness of the answers. This future experiment will be in a real life situation, i.e. a participant uses a mobile device to determine a situation of which he or she is part. Of course this will not be a real crisis situation but a simulation because of ethical considerations.

## References

1. N. H. T. S. Administration. Fatal accident reporting system (2011)
2. E. U. Commission. Community database on accidents on the roads in europe (2011)
3. Devlin, K.: Logic and Information. Cambridge University Press, Cambridge (1991)
4. Jaimes, A., Chang, S.: A conceptual framework for indexing visual information at multiple levels. IS&T/SPIE Internet Imaging 3964, 2–15 (2000)
5. Jiang, M., McGill, W.: Participatory Risk Management: Managing Community Risk Through Games. In: 2010 IEEE Second International Conference on Social Computing (Social Com), pp. 25–32. IEEE, Los Alamitos (2010)
6. Kokar, M., Matheus, C., Baclawski, K.: Ontology-based situation awareness. Information Fusion (10), 83–98 (2009)
7. Ramachandran, U.: Situation awareness. NSF, High-Confidence Software Platforms for Cyber-Physical Systems (2006)
8. Rifkin, A.: Evidence for a basic level in event taxonomies. Memory & Cognition 13(6), 538 (1985)
9. Rosch, E., Mervis, C., Gray, W., Johnson, D., Boyes-Braem, P.: Basic objects in natural categories. Cognitive psychology 8(3), 382–439 (1976)
10. Turoff, M.: Past and future emergency response information systems. Commun. ACM 45(4), 29–32 (2002)
11. Yue, D., Wang, S., Zhao, A.: Traffic accidents knowledge management based on ontology. In: Proceedings of the 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery FSKD 2009, vol. 07, pp. 447–449. IEEE Computer Society, USA (2009)



# Indexing Frequently Updated Trajectories of Network-Constrained Moving Objects

Zhiming Ding

Institute of Software, Chinese Academy of Sciences  
South-Fourth-Street 4, Zhong-Guan-Cun, Beijing 100190, P.R. China  
zhiming@iscas.ac.cn

**Abstract.** Index is a key technique in improving the query processing performance of moving objects databases. However, current index methods for moving object trajectories take trajectory units as the basic index records and frequent index updates are needed when location updates occur, which greatly affects the overall performance of moving objects databases. To solve this problem, we propose a new index method, network-constrained Moving Object Sketched-Trajectory R-Tree (MOSTR-Tree) in this paper, which outperforms previously proposed methods under frequent location updates.

**Keywords:** Moving Objects, Database, Index, Spatial-Temporal, Trajectory.

## 1 Introduction

Moving Objects Database (MOD) is a database which can track and manage the dynamically changing locations of moving objects such as cars, ships, flights, and pedestrians. An MOD system can manage huge numbers of moving objects so that index is crucial to query them efficiently.

In recent years, the moving object index problem has been intensely studied with a lot of methods proposed, which can be roughly divided into two categories: indexing of current positions of moving objects and indexing of complete trajectories of moving objects. In indexing current positions of moving objects, representative methods include Time Parameterized R-Tree (TPR-Tree) [1] and its variants. However, these indices can only deal with current positions of moving objects so that the queries about the historical positions of moving objects can not be supported.

In indexing the trajectories of moving objects, earlier work is focused on Euclidean-based methods [2], which take Euclidean trajectory units as the basic index records with each unit corresponding to a straight line segments in the  $X \times Y \times T$  space. Since a trajectory can correspond to a complicated curve, a large number of Euclidean trajectory units are needed to represent a trajectory, which can lead to huge numbers of records in the index structures.

More recently, increasing research interests are focused on network-based trajectory index methods for moving objects. Network-based trajectory indices usually adopt two-layered structures [3]. A network-based trajectory unit describes an even-speed movement along a route so that it can actually describe a curve in the  $X \times Y \times T$  space. Compared with Euclidean-based trajectory indices, network-based trajectory indices can effectively reduce the number of index records. However, current trajectory index methods still have a lot of limitations:

(1) Nearly all existing trajectory indices take trajectory units as the basic index records, whose granularity is too fine. As a result, frequent index updates are needed when location updates occur so that the efficiency can be greatly affected.

(2) Most current network-based trajectory indices adopt two-layer structures, which are not easy to be implemented in general extensible DBMSs such as PostgreSQL, even though they could be implemented in specially designed systems.

(3) Currently available network-based trajectory indices can only deal with the situation when the positions of moving objects are completely matched to the network. When moving objects run outside the network occasionally, their positions can not be expressed and included in the indices.

To solve the above problems, we propose a new index method, *network-constrained Moving Object Sketched-Trajectory R-Tree (MOSTR-Tree)*, in this paper. The basic idea is as follows. First the  $X \times Y \times T$  space is divided into equal-sized grid cells so that each trajectory (called “original trajectory”) is mapped to a “sketched trajectory” which is composed of the line segments connecting the centers of the grid cells that the original trajectory travels through. The sketched trajectory units are then organized into an R-Tree. Since the sketched trajectory has much coarser granularity than the original trajectory, the index updating cost can be greatly reduced.

The rest part of this paper is organized as follows. Section 2 presents a general model for moving object trajectories, Section 3 describes the structure and algorithms of MOSTR-Tree, and Section 4 gives experimental results and conclusions.

## 2 General Data Model for Network-Constrained Moving Object Trajectories

In this section, we propose a general data model for moving object trajectories, which can accommodate both network-constrained movements and free movements.

**Definition 1 (Traffic Network).** A traffic network (or simply network)  $N$  is defined as:

$$N = (R, J)$$

where  $R$  is a set of routes and  $J$  is a set of junctions.

**Definition 2 (Route).** A route of network  $N$ , denoted by  $r$ , is defined as follows:

$$r = (rid, geo, len, ((jid_j, pos_j)_{j=1}^m))$$

where  $rid$  is the identifier of  $r$ ;  $geo$  is a polyline describing the geometry of  $r$  (the beginning point and the end point of  $geo$  are called “0-end” and “1-end” of  $r$  respectively);  $len$  is the length of  $r$ ,  $(jid_j, pos_j)$  ( $1 \leq j \leq m$ ) describes the  $j$ th junction in route  $r$  (see Definition 3) where  $jid_j$  is the identifier of the junction and  $pos_j$  is the relative position of the junction in the route (suppose the total length of each route is 1, then any position in the route can be presented by a real number  $pos \in [0, 1]$ ).

**Definition 3 (Junction).** A junction of the traffic network  $N$ , denoted by  $j$ , can correspond to an intersection, an exit/entrance, or a route’s beginning-point/endpoint in the real traffic network, which is defined as follows:

$$j = (jid, loc, ((rid_i, pos_i)_{i=1}^n, m))$$

where  $jid$  is the identifier of  $j$ ,  $loc$  is the location of  $j$  which is a point value in the  $X \times Y$  plane,  $(rid_i, pos_i)$  ( $1 \leq i \leq n$ ) describes the  $i$ th route connected by  $j$ , where  $rid_i$  is the route identifier,  $pos_i \in [0, 1]$  is the relative position of  $j$  inside the route; and  $m$  is the connectivity matrix [4] of  $j$  which describes the transferability of moving objects from one route to another through the junction.

**Definition 4 (Network Position).** A position inside the network  $N$ , denoted by  $npos$ , is defined as follows:

$$npos = \begin{cases} jid ; & \text{if } npos \text{ is located in a junction;} \\ (rid, pos) ; & \text{if } npos \text{ is located in a route} \end{cases}$$

The position of  $npos$  can have two possibilities. It can either locate in a junction (in this case the identifier of the junction,  $jid$ , is used to express the location), or in a route (in this case its location is expressed as  $(rid, pos)$  where  $rid$  is the identifier of the route, and  $pos \in [0, 1]$  is the relative position inside the route).

**Definition 5 (Motion Vector).** A motion vector  $mv$  is a snapshot of moving object’s movement at a certain time instant and is defined as follows:

$$mv = (t, (x, y), v, d, npos)$$

where  $t$  is a time instant,  $(x, y)$ ,  $v$ ,  $d$  are the location, the speed, and the direction of the moving object at time  $t$  respectively, and  $npos$  is the network position of the moving object at time  $t$ . Among the parameters,  $(x, y)$ ,  $v$ , and  $d$  come from GPS, and  $npos$  comes from network matching.

If  $npos \neq \perp$  ( $\perp$  means “undefined”),  $mv$  is called “network matched”. If  $npos = \perp$ , then  $mv$  is not network matched.

**Definition 6 (Trajectory of Moving Objects).** The trajectory of a moving object,  $traj$ , is a sequence of motion vectors sent by the moving object through location updates [5] during its journey and is defined as follows:

$$traj = (mv_i)_{i=1}^n = ((t_i, (x_i, y_i), v_i, d_i, npos_i))_{i=1}^n$$

Two neighboring motion vectors of the trajectory,  $mv_i$  and  $mv_{i+1}$  ( $1 \leq i \leq n-1$ ), can form a trajectory unit, denoted as  $\mu(mv_i, mv_{i+1})$ . Depending on whether  $mv_i$  and  $mv_{i+1}$  are network-matched,  $\mu(mv_i, mv_{i+1})$  can correspond to different shapes in the  $X \times Y \times T$  space. If  $mv_i$  and  $mv_{i+1}$  are both network matched, then  $\mu(mv_i, mv_{i+1})$  describes the movement from  $mv_i$  to  $mv_{i+1}$  along the shortest path between  $npos_i$  to  $npos_{i+1}$  which corresponds to a curve in the  $X \times Y \times T$  space. If one of or both  $mv_i$  and  $mv_{i+1}$  are not network matched, then  $\mu(mv_i, mv_{i+1})$  corresponds to a straight line segment.

Trajectories are generated through location updates [4-5] of moving objects. During a location update operation, the original parameters sampled from GPS at the moving object side include  $t$ ,  $(x, y)$ ,  $v$ ,  $d$ , while  $npos$  can be computed through the network-matching procedure either at the moving object side or at the server side. During its movement, the moving object repeatedly samples the motion vector and compares it with the motion vector sent at the last location update. Whenever certain location update condition is met, the moving object will send the new motion vector to the server. The server will then append the new motion vector to the trajectory of

the moving object and therefore, the trajectory at the server side is growing over time. At certain time interval (say once a month), the database server needs to transfer the trajectories from the database to the historical repository and to refresh the database.

Suppose that we have a trajectory  $traj = (mv_i)_{i=1}^n = ((t_i, (x_i, y_i), v_i, d_i, npos_i))_{i=1}^n$ .

Through  $traj$  we can derive the location of the moving object at any time  $t_q \in [t_1, t_n]$  [4]. Besides, we can also compute the locations of the moving object between  $t_n$  and the current time  $t_{now}$ . Let's consider  $t_q \in (t_n, t_{now}]$ , the location of the moving object at  $t_q$  can be computed in two different ways:

(1) Return  $(x_n, y_n)$  as the position of the moving object at  $t_q$  if  $t_q \leq t_n + \tau$  ( $\tau$  is a time threshold, for instance 3 minutes);

(2) Derive the “computed position” of the moving object at  $t_q$  through the parameters contained in  $mv_n$ , and then return the computed position as the result [4].

Both methods have their advantages and disadvantages, depending on whether the wireless network is reliable enough and whether the traffic network is up-to-date and precise. In this paper we adopt the first method. Through some modifications, the method proposed in this paper can be used when the second method is adopted.

### 3 The Structure and Related Algorithms of the MOSTR-Tree

#### 3.1 Transforming Trajectories to Sketched Trajectories

Before trajectories can be transformed to sketched trajectories, we first need to partition the  $X \times Y \times T$  space into equal-sized grid cells. Suppose that the spatial-temporal range managed by the moving object database is  $I_x \times I_y \times I_t$ , where  $I_x = [x_0, x_1]$ ,  $I_y = [y_0, y_1]$ , and  $I_t = [t_0, \perp]$  (the endpoint of  $I_t$  is  $\perp$  (“undefined”) since the current time is always growing).  $I_x$  can be divided into  $n$  equal-sized sub-intervals of size  $\zeta_x$  where  $\zeta_x = \frac{x_1 - x_0}{n}$ . Similarly,  $I_y$  can be divided into  $m$  equal-sized sub-intervals of size  $\zeta_y$ , where  $\zeta_y = \frac{y_1 - y_0}{m}$ . For  $I_t$ , since its endpoint is undefined, we can divide it into equal-sized sub-intervals of size  $\zeta_t = \Delta t$  where  $\Delta t$  is a predefined value.

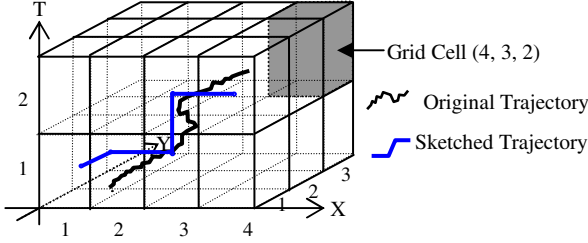
Through the above partitioning, the  $I_x \times I_y \times I_t$  space is divided into multiple equal-sized grid cells. Each grid cell can be identified by a triple  $(N_x, N_y, N_t)$ , where  $N_x$ ,  $N_y$ , and  $N_t$  are the cell's corresponding serial numbers along the X, Y, T axes. For instance, the gray-colored grid cell in Figure 1 is identified as (4, 3, 2).

After the  $I_x \times I_y \times I_t$  space is divided into equal-sized grid cells, we can then transform every trajectory into its corresponding sketched trajectory. To differentiate two kinds of trajectories, we call the trajectories discussed in Definition 6 as “original trajectory”.

**Definition 7 (Sketched Trajectory of Moving Object).** Suppose that the original trajectory of a moving object is  $traj = ((t_i, (x_i, y_i), v_i, d_i, npos_i))_{i=1}^n$ .  $traj$ 's sketched trajectory, denoted as  $sketch(traj)$ , is defined as follows:

$$sketch(traj) = (c_j)_{j=1}^k = ((t_j, x_j, y_j))_{j=1}^k$$

where  $c_j = (t_j, x_j, y_j)$  ( $1 \leq j \leq k$ ) is the center's coordinate of the  $j$ th grid cell that  $traj$  travels through. Two neighboring coordinates  $c_j$  and  $c_{j+1}$  ( $1 \leq j \leq k-1$ ) of  $sketch(traj)$  form a *Sketched-Trajectory Unit (STU)*, denoted as  $\hat{\mu}(c_j, c_{j+1})$ , which corresponds to a straight line segment connecting  $c_j$  and  $c_{j+1}$  in the  $X \times Y \times T$  space. A sketched trajectory can be seen as a sequence of sketched trajectory units so that it forms a polyline in the  $X \times Y \times T$  space, as shown in Figure 1.



**Fig. 1.** Partition of Grid Cells and the Resulted Sketched Trajectory

---

**Algorithm 1. Transforming Original Trajectory into Sketched Trajectory**

---

Global Arguments:

$I_x \times I_y \times I_t$ ; //Spatial-temporal Range of the database;  
 $\zeta_x, \zeta_y, \zeta_t$ ; // Parameters describing the size of grid cells;

INPUT:  $traj = (mv_i)_{i=1}^n = ((t_i, (x_i, y_i), v_i, d_i, npos_i))_{i=1}^n$ ;

OUTPUT:  $sketchTraj = ((t_j, x_j, y_j))_{j=1}^k$ ;

```

1. sketchTraj=NULL;
2. startingCell = getCellLocated(mv1);
3. append(sketchTraj, getCenter(startingCell));
4. IF  $n = 1$  THEN
5. |   Return(sketchTraj);
6. ELSE
7. |   currentCell = startingCell;
8. |   FOR  $i = 2$  to  $n$  DO
9. |   |   cellsTravelled = getCellsTravelled( $\mu(mv_{i-1}, mv_i)$ );
10. |   |   IF ( $|cellsTravelled|=1$ ) AND ( $extractCell(cellsTravelled, 1)=currentCell$ ) THEN
11. |   |   |   doNothing();
12. |   |   ELSE //  $\mu(mv_{i-1}, mv_i)$  travels through multiple grid cells
13. |   |   |   FOR  $j = 2$  to  $|cellsTravelled|$  DO
14. |   |   |   |   append(sketchTraj, getCenter(extractCell(cellsTravelled, j)));
15. |   |   |   ENDFOR;
16. |   |   |   currentCell = extractCell(cellsTravelled,  $|cellsTravelled|$ );
17. |   |   ENDIF;
18. |   ENDFOR;
19. |   Return(sketchTraj);
20. ENDIF.

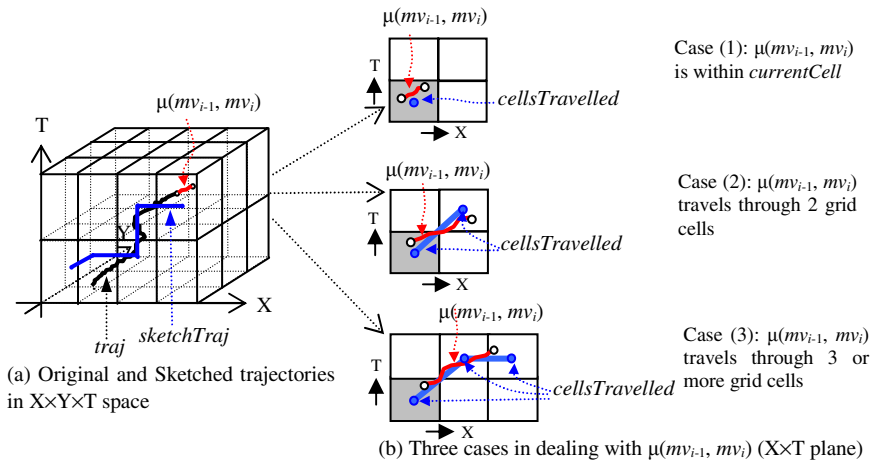
```

---

As depicted in Figure 1, the sketched trajectory approximates to the shape of the original trajectory with much less trajectory units. For a given original trajectory  $traj$ , the number of the sketched trajectory units in  $sketch(traj)$  is decided by the size of the grid cells. The bigger is the grid cell size, the less is the number of the sketched trajectory units contained in  $sketch(traj)$ , and vice versa.

Algorithm 1 describes the procedure of transforming an original trajectory to its corresponding sketched trajectory. In the algorithm, the function  $getCellLocated(mv)$  returns the grid cell within which a motion vector  $mv$  is located; the function  $getCellsTravelled(\mu)$  returns the grid cell sequence which a trajectory unit  $\mu$  travels through; the function  $extractCell(cellseq, i)$  extracts the  $i$ th grid cell from a grid cell sequence  $cellseq$ ; the function  $getCenter(cell)$  returns the center's coordinate of a grid cell  $cell$ , the function  $lcellseql$  returns the number of cells in a grid cell sequence  $cellseq$ , and the function  $doNothing()$  simply returns without doing anything.

In Algorithm 1, the trajectory units of  $traj$  are processed one by one. In dealing with a new trajectory unit  $\mu(mv_{i-1}, mv_i)$ , the algorithm first computes the grids cell(s) that the unit travels through by calling the  $getCellsTravelled()$  function (the result of the function can contain one or more grid cells), and then append the grid cell center(s) to  $sketchTraj$ .



**Fig. 2.** Transforming Original Trajectory to Sketched Trajectory

Figure 2 depicts three typical cases in dealing with a new trajectory unit  $\mu(mv_{i-1}, mv_i)$ , corresponding to lines 8~18 of Algorithm 1. The gray-colored grid cells in Figure 2(b) are the cells in which  $mv_{i-1}$  is located (*ie. currentCell*). Among the three cases, cases (1) and (2) happen more often than case (3). Since the granularity of grid cells is much coarser than that of original trajectory units, case (3) seldom occurs.

As shown in Figure 2(b), in case (1),  $\mu(mv_{i-1}, mv_i)$  is still inside  $currentCell$  and nothing will be done in this case (see lines 10~11). In case (2) and case (3),  $\mu(mv_{i-1}, mv_i)$  travels through 2 or more grid cells and therefore, the center's coordinates of the cells (except  $currentCell$ ) are appended to  $sketchtraj$  (see lines 13~15 of Algorithm 1).

### 3.2 Structure and Construction of the MOSTR-Tree

After the original trajectories are transformed to sketched trajectories, we can organize the sketched trajectory units into an R-Tree so that the MOSTR-Tree can be constructed. Figure 3 depicts the structure of the MOSTR-Tree.

The leaf nodes of the MOSTR-Tree contains records of the form  $\langle MBR, PT_{mo}, stu \rangle$ , where  $stu$  is a sketched trajectory unit,  $MBR$  is the MBR of  $stu$ , and  $PT_{mo}$  is the pointer or identifier leading to the complete database record of the corresponding moving object. The internal nodes of the MOSTR-Tree contain records of the form  $\langle MBR, PT_{node} \rangle$ , where  $MBR$  is the MBR bounding all the MBRs of the records in its child node, and  $PT_{node}$  is a pointer leading to the child node.

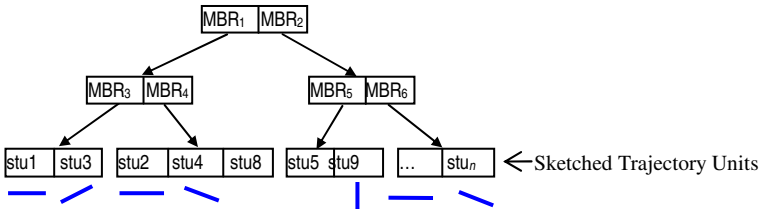


Fig. 3. Structure of the MOSTR-Tree

When constructing the MOSTR-Tree, the database server will transform every trajectory to its corresponding sketched trajectory and insert the sketched trajectory units into the MOSTR-Tree. The constructing procedure of the MOSTR-Tree is described in Algorithm 2. For simplicity, in Algorithm 2 we assume that no location update occurs, and more complicated cases will be discussed In Subsection 3.3.

---

**Algorithm 2. Constructing the MOSTR-Tree (without ongoing location updates)**

---

```

INPUT: trajSet; //the set of original trajectories to be indexed;
OUTPUT: mostrTree; //the MOSTR-Tree;
1. mostrTree = NULL;
2. FOR EACH traj ∈ trajSet DO
3. | sketchTraj = sketch(traj); //Computing sketched trajectory by calling Algorithm 1;
4. | FOR EACH sketchUnit IN getUnits(sketchTraj) DO
5. | | insert(mostrTree, sketchUnit);
6. | ENDFOR;
7. ENDFOR;
8. Return(mostrTree).

```

---

In Algorithm 2, the function  $getUnits(sketchTraj)$  extracts all the trajectory units from a sketched trajectory  $sketchTraj$  and returns them as a set. The function  $insert(mostrTree, sketchUnit)$  insert a sketched trajectory unit  $sketchUnit$  into the tree. The sketched trajectories and their units only appear when constructing and maintaining the MOSTR-Tree, and are not stored in the database permanently.

### 3.3 Maintaining and Constructing MOSTR-Tree with Ongoing Location Updates

During location updates, new trajectory units are appended to the trajectories so that the index also needs to be maintained. Let's consider a certain moving object  $mo$ . Suppose that its original trajectory is  $traj=(mv_i)_{i=1}^n = ((t_i, (x_i, y_i), v_i, d_i, npos_i))_{i=1}^n$  and  $sketch(traj) = ((t_j, x_j, y_j))_{j=1}^k$ . When  $mo$  launches a location update, it will send to the database server a new motion vector  $mv_u$ , which will be appended to  $traj$  by the server. The server first needs to check whether  $\mu(mv_n, mv_u)$  travels across the boundary of the grid cell where  $mv_n$  is located (*i.e.*  $getCellLocated(mv_n)$ ). If not, then the appending of  $mv_u$  to  $traj$  does not change  $sketch(traj)$  and the MOSTR-Tree does not need to be updated either. Otherwise, the new sketched trajectory unit(s) corresponding to  $\mu(mv_n, mv_u)$  need to be inserted to MOSTR-Tree.

Since the granularity of the sketched trajectory is much coarser than that of the original trajectory, the updating cost of the MOSTR-Tree can be greatly reduced. Algorithm 3 describes how the MOSTR-Tree is maintained during a location update.

---

#### Algorithm 3. Maintaining the MOSTR-Tree when Receiving a Location Update Message

---

```

INPUT: LUMsg=(moid, t, x, y, v, d, npos); //the loc. update message;
      mostrTree; // the MOSTR-Tree;
1.  $mv_u = (t, (x, y), v, d, npos)$ ;
2.  $mv_n = \text{final}(\text{getTrajectory}(moid))$ ;
3.  $currentCell = \text{getCellLocated}(mv_n)$ ;
4.  $cellsTravelled = \text{getCellsTravelled}(\mu(mv_n, mv_u))$ ;
5. IF ( $|cellsTravelled|=1$ ) AND ( $\text{extractCell}(cellsTravelled, 1) = currentCell$ ) THEN
6. | doNothing();
7. ELSE //  $\mu(mv_n, mv_u)$  travels through multiple grid cells
8. | FOR  $j = 2$  to  $|cellsTravelled|$  DO
9. | |  $sketchUnit = \hat{\mu}(\text{getCenter}(currentCell), \text{getCenter}(\text{extractCell}(cellsTravelled, j)))$ ;
10. | |  $\text{insert}(mostrTree, sketchUnit)$ ;
11. | |  $currentCell = \text{extractCell}(cellsTravelled, j)$ ;
12. | ENDFOR;
13. ENDIF.

```

---

In Algorithm 3, the function  $\text{getTrajectory}(moid)$  retrieves the original trajectory of the moving object whose identifier is  $moid$ , and the function  $\text{final}(traj)$  extracts the last motion vector from the trajectory  $traj$ .

For a running MOD system, the MOSTR-Tree can be constructed and maintained with ongoing location updates as described in Algorithm 4. In Algorithm 4, the location update messages received during the construction of the MOSTR-Tree are temporarily saved as a set in  $buffer$ . The function  $\text{fetch\_deleteMSG}(buffer)$  fetches and deletes a location update message from  $buffer$ . After the MOSTR-Tree is constructed, all the buffered location update messages are then processed with through Algorithm 3 until the buffer is empty, and then the server accepts new location update messages directly and maintain the MOSTR-Tree accordingly.



---

**Algorithm 4. Constructing&Maintaining the MOSTR-Tree with Ongoing Location Updates**

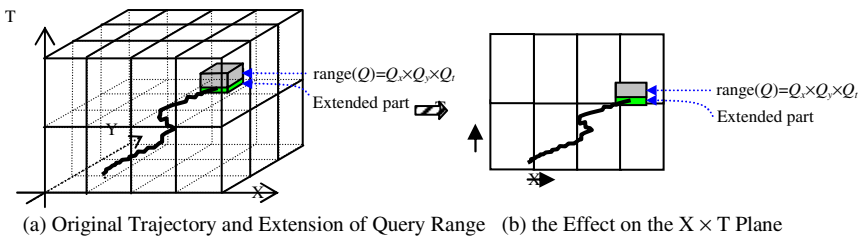
---

1. Call Algorithm 2 to construct the MOSTR-Tree, meanwhile save location update messages received during the construction procedure into *buffer*;
  2. **WHILE** *buffer*  $\neq \emptyset$  **DO** //new location update messages during this period are still saved to *buffer*
  3.     *LUMsg* = fetch\_deleteMSG(*buffer*);
  4.     Call Algorithm 3 to maintain the MOSTR-Tree according to *LUMsg*;
  5. **ENDWHILE**;
  6. **WHILE** MOD is running **DO**
  7.     Accept new location update message *LUMsg*;
  8.     Call Algorithm 3 to maintain the MOSTR-Tree according to *LUMsg*;
  9. **ENDWHILE**.
- 

### 3.4 Query Processing Based on the MOSTR-Tree

In this subsection, we discuss the query processing mechanism based on the MOSTR-Tree. Suppose that  $Q$  is an arbitrary query on moving object trajectories, whose query range is  $\text{range}(Q) = Q_x \times Q_y \times Q_t$  where  $Q_x = [q_x^0, q_x^1]$ ,  $Q_y = [q_y^0, q_y^1]$ , and  $Q_t = [q_t^0, q_t^1]$ . Query range describes X, Y, T ranges the query concerns and corresponds to a cube in the  $X \times Y \times T$  space.

In dealing with such a query, we first need to extend  $Q_t$  by replacing it with  $Q_t = [q_t^0 - \tau, q_t^1]$ . As stated in Section 2, the last reported location of the moving object continues to take effect for  $\tau$  time. We can imagine that the original trajectory has a vertical line segment of length  $\tau$  following the last trajectory unit. However, this vertical line segment is missing (not expressed as a record) in the index so that we should extend the query's time range to make the query range cover the vertical line. Otherwise, the corresponding moving object could be lost from the query result. After the extension, the new query range is  $\text{range}(Q) = Q_x \times Q_y \times Q_t$ , as shown in Figure 4.



**Fig. 4.** Extension of the Query Time Range

As depicted in Figure 4, the original trajectory  $traj = ((t_i, (x_i, y_i), v_i, d_i, npos_i))_{i=1}^n$  does not intersect with  $\text{range}(Q)$ . Assume that  $Q = \text{“query all moving objects which are inside geographical area } \alpha \text{ (suppose that } (x_n, y_n) \in \alpha \text{) at time } t_n + \epsilon \text{ (where } \epsilon \leq \tau \text{)”}$ . If we do not extend the query time range, then the moving object will not be contained in the query result, which is inconsistent with the statement in Section 2. By

extending the query’s time range, the moving object’s trajectory intersects the new query range and the moving object can be included in the query result.

After the query range is extended, the system needs to align the query range to grid cell centers, as shown in Figure 5.

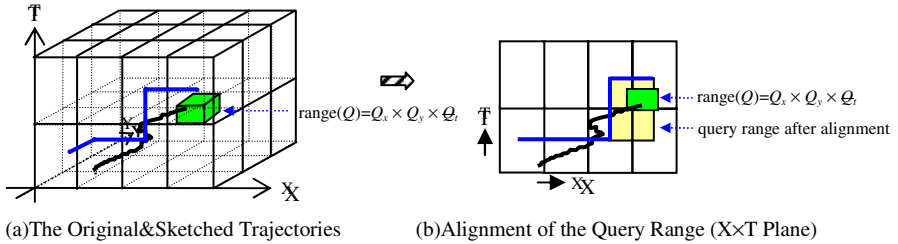


Fig. 5. Alignment of the Query Range to Grid Cell Centers

In Figure 5, the original trajectory of the moving object intersects  $range(Q) = Q_x \times Q_y \times Q_z$ , and should be included in the query result. However, the sketched trajectory does not intersect with  $range(Q)$ . If we search in the MOSTR-Tree with  $range(Q)$  directly, the moving object will be lost from the query result. To solve this problem, we should make necessary adjustment to the query range. From analysis we can see that, if the query range is aligned to the corresponding grid cell centers, we can be assured that for all moving objects whose original trajectories intersect the query range, their sketched trajectories will intersect the aligned query range.

In the following, we discuss how to align the query range  $range(Q) = Q_x \times Q_y \times Q_z$  to the corresponding grid cell centers. Let’s first consider  $Q_x = [q_x^0, q_x^1]$ . We can make the following transformation:

$$q_x^0 = x_0 + \left( \left\lfloor \frac{q_x^0 - x_0}{\xi_x} \right\rfloor + 0.5 \right) * \xi_x \quad q_x^1 = x_0 + \left( \left\lfloor \frac{q_x^1 - x_0}{\xi_x} \right\rfloor + 0.5 \right) * \xi_x$$

We can make similar transformations with  $Q_y = [q_y^0, q_y^1]$  and  $Q_z = [q_z^0 - \tau, q_z^1]$ , so that the query range is aligned to the corresponding grid cell centers.

After the alignment transformation is conducted, we can get a new query range. We can then query the MOSTR-Tree with the new query range and get a set of candidate moving objects, which will then be further evaluated according to the query conditions. The final result will be output to the querying user.

### 4 Performance Evaluation and Conclusion

To evaluate the performance of MOSTR-Tree, we conducted a series of experiments and the results show that MOSTR-Tree provides better performance than TB-Tree, a representative trajectory-unit-based index method, under frequent location updates. For those MOD systems that only manage historical (static) trajectories without ongoing location updates, we can certainly use TB-Tree and other trajectory-unit-based index methods [1-3, 6-8] which have better selectivity than MOSTR-Tree.

However, heavy location updates are inevitable for most running MOD systems since MOD's main purpose is to track the dynamically changing locations of moving objects, and in this case MOSTR-Tree shows its superiority.

**Acknowledgements.** The work was partially supported by NSFC under grand number 60970030.

## References

- [1] Saltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the Position of Continuously Moving Objects. In: Proc. of ACM SIGMOD 2000, TX, USA (2000)
- [2] Pfoser, D., Jensen, C.S., Theodoridis, Y.: Novel Approach to the Indexing of Moving Object Trajectories. In: Proc. of the 26th VLDB, Cairo, Egypt (2000)
- [3] Almeida, V.T., Güting, R.H.: Indexing the Trajectories of Moving Objects in Networks. *GeoInformatica* 9, 1 (2005)
- [4] Ding, Z., Güting, R.H.: Managing Moving Objects on Dynamic Transportation Networks. In: Proc. of SSDBM 2004, Santorini, Greece (June 2004)
- [5] Ding, Z., Zhou, X.: Location update strategies for network-constrained moving objects. In: Haritsa, J.R., Kotagiri, R., Pudi, V. (eds.) DASFAA 2008. LNCS, vol. 4947, pp. 644–652. Springer, Heidelberg (2008)
- [6] Cudré-Mauroux, P., Wu, E., Madden, S.: TrajStore: An adaptive storage system for very large trajectory data sets. In: Proc. of ICDE, pp. 109–120 (2010)
- [7] Rasetic, S., Sander, J., Elding, J., M.: Nascimento: A Trajectory Splitting Model for Efficient Spatio-Temporal Indexing. In: VLDB 2005, pp. 934–945 (2005)
- [8] Botea, V., Mallett, D., Nascimento, M., Sander, J.: PIST: An Efficient and Practical Indexing Technique for Historical Spatio-Temporal Point Data. *GeoInformatica* 12(2), 143–168 (2008)

# Online Index Selection in RDBMS by Evolutionary Approach\*

Piotr Kołaczkowski and Henryk Rybiński

Institute of Computer Science, Warsaw University of Technology  
{pkolaczk,hrb}@ii.pw.edu.pl

**Abstract.** In recent years, many algorithms for automatic physical database tuning have been proposed and successfully used in tools for administration of relational database management systems. The novel method described in this paper uses a steady-state evolutionary approach to continuously give index recommendations so that the database management system can adapt to changing workload and data distribution. Contrary to online algorithms offering recommendations on a per-query basis, our solution takes into account index reuse across different queries. The experiments show that the quality of the recommendations obtained by the proposed method matches the quality of recommendations given by the best offline index selection algorithms. Moreover, high performance and low memory footprint of the method make it suitable for autonomic database tuning systems.

## 1 Introduction

Relational Database Management Systems (RDBMS) have been continuously developed for more than three decades now and became very complex. To administer them, a lot of experience and knowledge is required. The costs of employing professional database administrators are often much higher than the costs of database software licensing [4]. Recently, we have been observing a high demand on solutions reducing these costs. Especially intelligent, automatic tools for solving complex administration problems are very helpful. One of such complex problems is performance tuning. In this work we consider the aspect of proper index selection (IS), which often significantly affects the overall database application performance. The importance of proper IS increases with the size of a database. A perfect tool for automatic IS should not only provide good index recommendations, but also be able to continuously monitor the database system, and adapt the selected index set appropriately whenever the database structure, content or workload change. This self-tuning activity should not pose a significant load on the database system or consume large amounts of resources, because otherwise the main purpose of using it, i.e. increasing the overall performance of the system, would be defeated.

---

\* The work has been supported by the Ministry of Science and Higher Education grant No N N 516 375936.

There has been a lot of research into automatic physical database tuning recently. Some of these methods, successfully employed in commercial RDBMS, e.g. [2] provide very high quality of index recommendations. However, high recommendation quality usually comes at a price of high computational cost and high memory usage, so these methods are not well suited for self-tuning database systems. On the other hand, some methods (e.g. [10,11]) propose continuous on-line tuning algorithms that reduce the tuning overheads by carefully choosing a small subset of indexes that need to be concerned as candidates for materialization. However, they make simplifying assumptions about the underlying cost model, e.g. they assign benefits to single indexes, as if these benefits were independent on the existence of other indexes. Methods based on that approach are fast, but may give results far from the optimum.

Our aim was to develop a method that would give near-optimal results and simultaneously be capable of running in real-time mode. The method is based on an evolutionary approach. Evolutionary algorithms tend to find good solutions fast for large and complex combinatorial problems, and can adapt the solution to the changing external conditions. This approach has been successfully used to tackle the offline ISP [8].

One of the most costly operation when finding optimal index set is evaluating the quality of the various candidate index configurations. This often requires several query planner invocations to find the optimal plans for the tasks that might use some of the candidate indexes. Finding the optimal plan for the given task is itself a hard optimisation problem. The number of query planner invocations can be reduced by caching parts of the plans that are not affected by the existence and usage of indexes [9], however this increases memory usage for storing many equivalent query plans for each query. In [7] we have proposed a method to explore the space of index configurations without invoking the query planner at all. The algorithm treats the IS problem (ISP) as a multi-query optimisation problem, with objective to find the set of query plans that minimise the summary execution costs, assuming the optimiser is allowed to introduce whatever indexes it wishes. Then, the set of optimal indexes is retrieved a posteriori from the query plans, by linear scanning the plan nodes.

In the following section of this paper we describe a variant of algorithm [7], capable of real-time index selection. In Section 3 we present results of experiments comparing efficiency and performance of our method to the greedy per-query index selection [12] and relaxation based IS [2]. Finally we discuss the strong and weak points of the proposed method and sketch plans for future research.

## 2 Index Selection Algorithm

The method consists of three main components: (1) the incremental workload compression module responsible for reducing the number of input queries and adjusting their weights according to how frequent they appear in the workload, (2) the evolutionary IS module responsible for continuously providing the best index configurations, and (3) the index materialization service responsible for

making decisions, when to physically instantiate the advised indexes. The components work simultaneously, forming a pipeline.

### Workload Compression

The aim of workload compression is to reduce the number of tasks input to the IS module, in order to increase efficiency of the IS tool. The reduction of the workload size must not cause significant deterioration of IS results. As shown in [3], the workload compression problem can be treated as a special case of a clustering problem, where each data point is a task in the workload and the distance between two points is a function of predicted ISP results quality loss if replacing one task with another one. The incoming tasks are clustered, and afterwards one task in each cluster is retained. The retained tasks form the compressed workload. The number of tasks in the compressed workload equals to the number of the clusters, so the clustering algorithm should create as few clusters as possible to achieve good compression ratio. On the other hand, the tasks in each cluster should be similar to each other, especially by means of using the same indexes in having similar query execution plans. The more similar are the tasks in each cluster, the less degradation of the ISP results is expected.

In [6] we proposed a novel algorithm for workload compression. The method is based on a simple and fast incremental clustering. The basic idea behind our method is to maintain a set of clusters and for each incoming task assign the task to appropriate cluster, or form a new cluster from this task. A *seed task* is the first task added to the cluster. The first incoming task becomes the seed of the first cluster. For each subsequent incoming task, the nearest seed task  $q^*$  is found. If the distance between these two tasks exceeds a user-defined limit, then the incoming task forms a new cluster and becomes its seed. Otherwise, the task is added to the cluster having seed  $q^*$ . The compressed workload is formed from the cluster seeds. When a task leaves the time window, it is also removed from the previously assigned cluster. Each cluster seed  $q^*$  is assigned a weight  $\lambda(q^*)$  equal to the number of all queries in the cluster. In practice, weights are adjusted incrementally, whenever a task is added to or removed from a cluster.

### Index Selection

For the IS, an evolutionary search strategy described in [7], adapted to continuous, online operation is used. The IS module maintains a population of vectors of query execution plans. A single vector, called further *individual*, consists of plans created for every seed task supplied by the workload compression model. One plan for each of the seed tasks is kept in an individual. The population size is limited by a small, user defined constant  $m$ . The initial population is formed as a single individual initialized with query plans optimal for the current database state, generated by the query planner. Starting from this vector of optimal query plans guarantees, that the index recommendations given by the index advisor will never be worse than the current materialized index set. Then, the individuals in the population are reproduced until the population reaches the target

size  $m$ . From this point on, the size of the population remains constant and new individuals replace the old ones. To reduce the overhead of copying large query execution plans and index metadata on each individual reproduction phase, the plans and indexes are lazily copied. The actual copying is done only for small parts of the vector that is mutated. Whenever a new seed task  $q$  appears in the output of the workload compression module, all the individuals in the population are extended by the optimal plan of  $q$ , generated by the query planner for the current state of the database. Similarly, if a seed task is removed from the output of the workload compression module, the individuals are modified appropriately. After each iteration an individual with the highest fitness is remembered and kept as the current solution of the index selection problem. The set of result indexes is read as a sum of sets of indexes used by each of the plans in the best plan vector.

The fitness of an individual  $\mathbf{v}$  is evaluated from the formula:

$$f(\mathbf{v}) = -(\mathcal{C}_{\text{exe}}(\mathbf{v}) + \beta \mathcal{C}_{\text{mat}}(\mathcal{D}(\mathbf{v}^*))) \left(1 + \alpha \frac{\mathcal{S}(\mathcal{D}(\mathbf{v}^*))}{s_{\text{db}}}\right) \quad (1)$$

where:

- $\mathcal{C}_{\text{exe}}(\mathbf{v})$  is the cost of executing all the plans in the plan vector  $\mathbf{v}$ ,
- $\mathcal{C}_{\text{mat}}(D_x)$  is the cost of materializing the indexes  $D_x$ ,
- $\mathcal{C}_{\text{sto}}(D_x)$  is the cost of storing the indexes  $D_x$ ,
- $\mathcal{D}(\mathbf{v})$  is the set of indexes used by the plans in  $\mathbf{v}$ ,
- $s_{\text{db}}$  is the total size of the database,
- $\mathcal{S}(D_x)$  is the size of the indexes  $D_x$ ,
- $\alpha$  is a non-negative constant set by the user, determining how much is the fitness affected by the size of selected indexes,
- $\beta$  is a non-negative constant set by the user determining how “aggressively” the optimizer reacts to workload changes.

The selection and replacement operators are responsible for applying selective pressure to the individuals in the population, so that the better fitted ones have higher chance to survive. There are many different ways of implementing these operators [1]. Because no single selection and replacement scheme is best suited for every problem, we have implemented three different schemes: tournament, proportional and fitness-uniform [5]. Experiments that we have made have shown that the tournament selection and replacement give the fastest convergence. In tournament selection, a small number of individuals are chosen randomly using uniform distribution and the best individual among them is returned. The procedure of the tournament replacement is the same, except that instead of the best individual, the worst one is chosen to be replaced. Thanks to tournament selection, the best-fitted individuals have more chances to reproduce. Thanks to tournament replacement, the best-fitted individuals have more chances to survive. Therefore, those parts of the solution space are explored more intensively, where some good solutions have already been found.

The mutation operator is responsible for creating new individuals from the selected by the selection operator ones. The basic mutation of an individual  $\mathbf{v}$

consists of applying small *atomic transformations* to a randomly chosen subset of plans in  $\mathbf{v}$ . An atomic transformation is the smallest indivisible modification that transforms a query plan into a different equivalent query plan. In our solution we use the following transformation classes: (1) join reordering, (2) choosing a different join algorithm, and (3) choosing a different table access method, including introduction of a new index. For details of these transformations see [7].

A *mutation of a query plan* consists of at least one successful atomic transformation. All the transformation classes have assigned a certain constant probability. Probabilities of triggering transformations of different classes may be set independently. The number of successful single plan transformations  $k$  performed per mutation is a random number, generated with such a probability distribution, that numbers near 1 are the most probable. This makes most of the mutations small and enables the algorithm to perform *hill climbing*. It is essential that sometimes the mutation is large, in order to avoid premature convergence to a locally optimal solution and to ensure the whole search space is explored.

### Index Materialization

The output index set recommendation  $D^*$  given by the index selectio module is available at every moment from the start of the optimization process and it changes over time. There are two causes of the changes of  $D^*$ : (1)  $D^*$  has not converged to the stable optimum yet, and (2) the environment changes and makes the previously optimal  $D^*$  obsolete. This behaviour must be taken into consideration in the index materialization strategy. We predict that materializing the recommendation  $D^*$  whenever it changes can cause unnecessary work spent for creating indexes that would be otherwise not recommended if the materialization was postponed until the  $D^*$  converges. However, because of even slight but constant environment changes (e.g. workload characteristics changes), the output recommendation  $D^*$  may never converge to the optimum. Thus, a simple strategy of materializing  $D^*$  whenever it is stable for some user-defined period of time would generally not work. Our proposition is illustrated in Alg. 1, executed after every iteration of the main loop of the evolution. Whenever  $D^*$  changes, indexes added to and removed from it are recorded together with the number of current iteration  $i_{\text{cur}}$  of the main loop of the evolution. An index  $d$  can be materialized in the database only if was present in the  $D^*$  continuously for at least  $\mu$  iterations. Consequently, an index  $d$  can be removed from the database only if it was *not* present in the  $D^*$  continuously for at least  $\mu$  iterations. The  $\mu$  constant must be determined experimentally and should be larger than the average number of iterations of the main loop required for  $D^*$  to converge for a stationary ISP.

## 3 Experimental Results

In this section we describe the experiments performed to test how the implementation of the proposed algorithm performs at solving both synthetic and



**Algorithm 1.** Index materialization strategy

---

```

1: loop
2:   if  $D^*$  changed then
3:     let  $D_{\text{old}}^*$  be the previous value of  $D^*$ 
4:     for all  $d \in D^* \setminus D_{\text{old}}^*$  do
5:        $i_{\text{add}}(d) \leftarrow i_{\text{cur}}$ 
6:     end for
7:     for all  $d \in D_{\text{old}}^* \setminus D^*$  do
8:        $i_{\text{rm}}(d) \leftarrow i_{\text{cur}}$ 
9:     end for
10:  end if
11:  for all  $d \in D_{\text{mat}}$  do
12:    if  $d \notin D^* \wedge i_{\text{cur}} - i_{\text{rm}}(d) > \mu$  then
13:      remove  $d$  from the database
14:    end if
15:  end for
16:  for all  $d \in D^*$  do
17:    if  $d \notin D_{\text{mat}} \wedge i_{\text{cur}} - i_{\text{add}}(d) > \mu$  then
18:      materialize  $d$  in the database
19:    end if
20:  end for
21: end loop

```

---

real-world, stationary and non-stationary ISP instances of various sizes. We compare our implementation (EVO) to the best-effort implementations of the two state-of-the-art IS algorithms: (1) the greedy one (GR) used in the IBM DB/2 RDBMS [12] and the relaxation based one (RB) used in the Microsoft SQL Server RDBMS [2]. The implementation of the reference algorithms has been consulted with their original authors i.e. Daniel C. Zilio from IBM Toronto Lab and Nicolas Bruno from Microsoft Research.

RB and EVO advisors provided index recommendations of superior quality for synthetic and real workloads consisting of many queries. Fig. 1 shows that all the advisors provide recommendations of the same cost if there is enough space for indexes for the TPC-H workload. A similar behaviour was observed for the other workloads. When the space limit is reduced, the estimated execution cost gets higher, because less indexes can be included in the result recommendation. The GR index advisor can only exclude indexes from the final recommendation and cannot replace them with smaller ones, so the cost increase is much faster than the cost increase observed for the EVO and RB advisors. For example, when the storage space constraint  $s_{\text{max}}$  is set to  $3 \times 10^8$ , the estimated execution cost of workload is over 2 times higher for the GR recommendation than for the EVO and RB recommendations. Thus, ability of merging indexes can provide large cost benefits for some kinds of workloads.

It is not sufficient for an index advisor to give good index recommendations, if the time required for getting them is unacceptable, e.g. grows exponentially with

the number of input queries. Such an advisor would not be practically useful. Performance of EVO index advisor is superior for all tested workloads.

Due to the lack of the initial candidate selection phase, first results are given in a few seconds and also not much more time is required to obtain recommendations of acceptable quality. RB index advisor is the slowest one. In the case of the largest workload MMORPG, 10 minutes was not enough to give any index recommendation. This index advisor approaches a solution from the upper side of the size limit, so when there are lots of initial index candidates, it requires a lot of time till the superfluous index candidates are removed. The initial phase of the candidate index set generation performed by RB and GR advisors also adds up to the total startup time, but it is only a fraction of time required for the relaxation performed by the RB advisor. This delay is seen for the TPC-H workload but it is negligible for the other simpler workloads.

To check the ability to adapt to workload changes by our algorithm, we compared quality of index recommendations given by the evolutionary index advisor working in the online mode to the index advisor in the offline mode. Both synthetic and real workloads were used. To check what how fast the online index advisor reacts to workload characteristic changes, we created mixed read/read and read/write workloads by using two different random workload generators, denoted as RAND-1 and RAND-2. As real-world workloads we used first  $n$  queries from the MMORPG workload for two values of  $n$ :  $n = 2000$  and  $n = 10000$ .

The workloads were fed to two evolutionary index advisors, online and offline. The advisors had a storage limit set to 1000% of the initial size of the database. The workload compression module was turned on for each index advisor. For each workload we measured: (1) total estimated cost of execution of the workload in the database without any indexes, (2) total estimated cost of execution of the workload in the database with indexes previously recommended for the whole workload by the offline index advisor and (3) sum of total estimated cost of execution of the workload and estimated cost of materialization of indexes when running the workload in the database with an index advisor running in the online mode (online). Results are illustrated in Table II

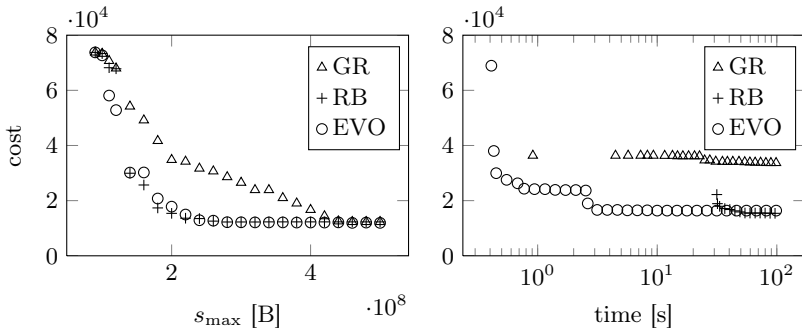


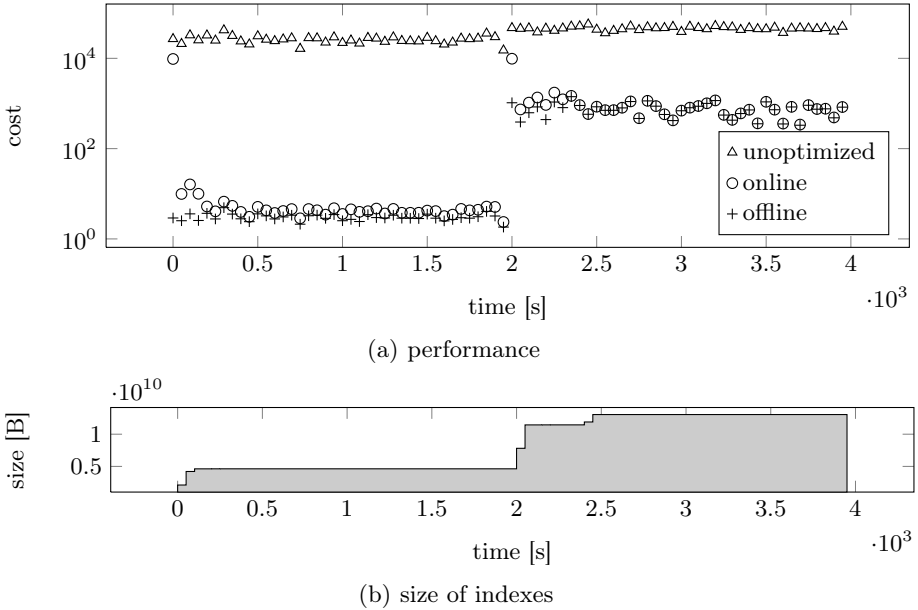
Fig. 1. Comparison of quality and performance of index recommendations given by various index advisors for the TPC-H workload.

The estimated cost of execution of the synthetic workloads were about twice as high for the unoptimized database as for the database with indexes previously recommended by the offline index advisor. The estimated execution costs obtained from running the online index advisor, were higher than the costs obtained for the preoptimized database, but significantly lower than the costs obtained for the unoptimized database. The online index advisor has more difficult task than the offline one, because it does not know the queries in advance. Some of the queries are executed before appropriate indexes are available, thus increasing the overall execution cost. Also the costs of creating indexes are not negligible, although these costs can be reduced by creating deferred indexes [10]. Deferred indexes were not used in our experiments. Both advisors were able to give much better improvement of the estimated workload execution cost for the real-world workload than for the synthetic ones. However, the total cost of execution of the workload consisting of 2000 queries is about three times higher when the indexes were recommended by the online index advisor, than for the offline one. This was primarily caused by the high cost of executing the queries before indexes were materialized. For the workload consisting of 10000 queries, the workload execution costs obtained by using both advisors were almost equal. In all of the experiments, estimated costs of the index materialization did not exceed 5% of the final execution cost of executing the workload with the online index advisor active, so we conclude the algorithm is resilient to noise and does not overreact to temporary variations in query distribution.

The detailed log of a sample run of the online index advisor is presented in Fig. 2. Each point in the figure presents an average cost of executing 50 consecutive queries. The run demonstrates the ability of the online index advisor to immediately create indexes for new queries, both at the beginning of the workload, as well as just after the moment when the workload characteristic changes. The statements sent before  $t = 2 \times 10^3 s$  were not present in the further part of the workload. Unused indexes are not dropped immediately after changing the workload characteristic, in order to avoid recreating them in case the statements they were generated for appear back again in the workload. After running the experiment for a much longer time than shown in the figure, we observed systematic dropping of unused indexes. By another experiment we assured that the indexes are also dropped when the workload changes from read-mostly to write-mostly so that the index maintenance cost becomes higher.

**Table 1.** Estimated synthetic workload execution costs for various database configurations. Averages from 10 runs are given.

		RAND-1 uniform	RAND-1 mixed	MMORPG $n = 2000$	MMORPG $n = 10000$
Unopt.	(1)	$4.6 \times 10^6$	$6.2 \times 10^6$	$63 \times 10^4$	$28 \times 10^5$
Offline	(2)	$2.3 \pm 0.1 \times 10^6$	$2.9 \pm 0.1 \times 10^6$	$1.4 \pm 0.1 \times 10^4$	$2.0 \pm 0.1 \times 10^5$
Online	(3)	$3.0 \pm 0.1 \times 10^6$	$3.5 \pm 0.1 \times 10^6$	$4.5 \pm 0.2 \times 10^4$	$2.1 \pm 0.1 \times 10^5$



**Fig. 2.** Sample log of autonomic database tuning session involving a mixed read/read workload composed of streams having two different RAND-1 profiles

## 4 Summary

We have shown that the algorithm based on evolutionary query plan transformations can be efficient at solving stationary and non-stationary IS problems. Unlike many theoretical approaches, the presented solution is complete and ready to be implemented in the relational database system. Our research covered all important aspects of building an autonomic index advisor, from the algorithm for creating and updating a terse workload representation, through the algorithm for calculating good index configurations, up to the algorithm for deciding which indexes to build and when. We have also implemented a working prototype of the index advisor and tested it on synthetic and real-world IS problem instances.

Our research focused on ISP only. However, proper index selection is only a part of physical database tuning process. Another important areas of interest, not covered by this work are: selection of materialized views, recommendation of horizontal and vertical data partitioning, multidimensional clustering. These aspects of database tuning should not be treated separately because strong inter-connections between various physical database structures, e.g. between indexes and materialized views have been observed [13]. We believe that the approach based on direct query plan manipulation can be also successfully applied to these areas and further research in this direction should be continued.

## References

1. Bäck, T.: Evolutionary algorithms in theory and practice. Oxford University Press, Oxford (1996)
2. Bruno, N., Chaudhuri, S.: Automatic physical database tuning: a relaxation-based approach. In: SIGMOD 2005: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 227–238. ACM, New York (2005)
3. Chaudhuri, S., Gupta, A.K., Narasayya, V.: Compressing SQL workloads. In: SIGMOD 2002: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 488–499. ACM, New York (2002)
4. Ganek, A.G., Corbi, T.A.: The dawning of the autonomic computing era. IBM Syst. J. 42(1), 5–18 (2003)
5. Hutter, M.: Fitness uniform selection to preserve genetic diversity. In: IEEE International Conference on E-Commerce Technology, vol. 1, pp. 783–788 (2002)
6. Kołaczkowski, P.: Compressing very large database workloads for continuous online index selection. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 791–799. Springer, Heidelberg (2008)
7. Kołaczkowski, P., Rybiński, H.: Automatic Index Selection in RDBMS by Exploring Query Execution Plan Space. SCI, vol. 223, pp. 3–24. Springer, Heidelberg (2009)
8. Kratica, J., Ljubić, I., Tošić, D.: A genetic algorithm for the index selection problem (2003), <http://citeseer.ist.psu.edu/568873.html>
9. Papadomanolakis, S., Dash, D., Ailamaki, A.: Efficient use of the query optimizer for automated physical design. In: VLDB 2007: Proceedings of the 33rd International conference on Very Large Data Bases, pp. 1093–1104. VLDB Endowment (2007)
10. Sattler, K.U., Schallehn, E., Geist, I.: Autonomous query-driven index tuning. In: IDEAS 2004: Proceedings of the International Database Engineering and Applications Symposium (IDEAS 2004), pp. 439–448. IEEE Computer Society, USA (2004)
11. Schnaitter, K., Abiteboul, S., Milo, T., Polyzotis, N.: Colt: continuous on-line tuning. In: SIGMOD 2006: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 793–795. ACM Press, New York (2006)
12. Skelley, A.: DB2 advisor: An optimizer smart enough to recommend its own indexes. In: ICDE 2000: Proceedings of the 16th International Conference on Data Engineering, p. 101. IEEE Computer Society Press, Washington, DC, USA (2000)
13. Zilio, D.C.: Physical Database Design Decision Algorithms and Concurrent Reorganization for Parallel Database Systems. Ph.D. thesis (1998)

# Towards Balanced Allocations for DHTs

George Tsatsanifos<sup>1</sup> and Vasilis Samoladas<sup>2</sup>

<sup>1</sup> National Technical University of Athens

gtsat@dblab.ece.ntua.gr

<sup>2</sup> Technical University of Crete

vsam@softnet.ece.tuc.gr

**Abstract.** We consider the problem of load-balancing structured peer-to-peer networks. Load-balancing is of major significance for large-scale decentralized networks in terms of enhanced scalability and performance. Our methods focus mainly on task-skew. Specifically, we address the problem with general rigorous algorithms on the basis of migration. In particular, the cornerstones of our methods are the notions of *virtual nodes*, *replication* and *multiple realities*. Finally, our work is complemented with extensive experiments.

## 1 Introduction

We revisit the problem of load-balancing structured p2p networks [8]. Our intention is to develop a realistic balancing paradigm, which can be used on top of any DHT overlay, mitigating load imbalance effects, in order to enhance performance and therewith scalability. Our methods are focused mainly on task-skew. We address the problem with general algorithms based on migration, reducing the problem to a balls-and-bins game by facing hosts as bins and virtual nodes as balls. More specifically, we study rudimentary balancing techniques, namely *virtual nodes*, *replication*, and *multiple realities*, over specific performance evaluation metrics. By scrutinizing each method in isolation, we draw valuable conclusions on how they can interoperate with each other, and augment them by applying each a balls-in-bins model.

The remainder of this paper is organized as follows. Section 2 reviews relevant literature. Section 3 discusses augmented rudimentary balancing techniques. Section 4 evaluates our work, and Section 5 concludes.

## 2 Related Work

The common balancing paradigm for DHTs consists of randomized hashed functions that are supposed to distribute data load among peers in a uniform fashion. However, this approach is far inadequate, especially in cases where certain parts of key space receive disproportional portions of popularity. Specifically, if node and item identifiers are randomly chosen, there is a  $\Theta(\log n)$  imbalance factor in the number of items stored at a node. Mercury [2] supports explicit load balancing using random sampling. *The threshold algorithm* [4] consists of a load-balancing protocol whereby each tuple-insert or delete is followed by an execution of the load-balancing algorithm, which may involve moving sequential data across peers. Their rationale that a node attempts to shed

its load whenever it increases by a factor  $\delta$ , and attempts to gain load when it drops by the same factor. Nevertheless, the threshold algorithm cannot be applied to systems indexing tuples along many dimensions. Karger and Ruhl propose a family of caching protocols to decrease the occurrence of hotspots. *Address-space balancing* [6] improves consistent hashing in that every node is responsible for a fraction of the address space with high probability. The protocol is dynamic, with an insertion or deletion causing other nodes to change their positions. On the other hand, *Item balancing* [6] targets at the distribution of items to nodes. Rao et al. in [7] introduce the notion of the *virtual server*, a single node of the underlying DHT contrary to the physical host being responsible for more than one virtual servers. Most importantly, the topology of the virtual servers should be able to adapt to dynamic changes of the actual network accordingly.

### 3 Balanced Allocations

In this section, we augment existing rudimentary balancing techniques with algorithms that enhance performance. More specifically, we reduce the problem to a balls-in-bins game by facing peers as bins and virtual nodes as balls. We now delineate some terminology conventions we have made for distinguishing certain notions we use. We define as a *node* the entity responsible for some part of the key space of the overlay, which was assigned according to the overlay protocols. As far as this work is concerned, nodes and peers are two distinct notions. Henceforth, *peers* serve as node containers. A peer's task is to deliver incoming messages to the appropriate comprised nodes and forward all their outgoing messages to their destination.

Most of the prior works in balancing use a straightforward approach for random and arbitrary assignment of balls to bins, requiring no knowledge about balls, bins, their structure and their loads. On the other hand, our methods are iterative allocation schemes based exclusively on local knowledge that exploit balls-in-bins games with minimum makespan scheduling approximate solutions [1]. To elaborate, assume that there has already been made some sort of assignment of nodes to peers. Balancing takes place among the nodes of a neighborhood, where a peer's neighborhood corresponds to the union peer-set of all peers containing nodes from the routing tables of the peer's comprised nodes. For each iteration a random peer from an unbalanced peer-set is picked, and all nodes from linked peers are reassigned successively to the bins of that specific peer-set, starting from the heaviest node and assigning it to the lightest peer of the peer-set. In essence, our methods constitute infinite processes, as they are repeated sequentially, and we stop when no significant changes take place in the structure and a convergence criterion has been met.

The principle of the *multiple realities* technique is to maintain multiple, independent coordinate spaces, with each node in the system being assigned a different zone in each coordinate space. Data tuples are being replicated in every reality, enhancing this way, data availability and fault-tolerance. We consider the case where  $n$  peers participate in the network in all  $g$  realities. We also impose a limitation for a peer to participate in each reality with one node only. In order to balance, we reassign the nodes from all realities of a selected peer's nodes' vicinities, in such a way that all peers acquire nodes with approximately the same summing loads. When a peer "broadcasts" its request to

all realities simultaneously, the fastest answer is returned to the user, and thus latency ameliorates. Thus, when a lookup query is enacted in all  $g$  realities simultaneously requires  $g$  times more messages to be resolved.

For the *parallel universes* technique we aim at reducing latency by invoking queries simultaneously in all realities and get the fastest answers. Since balancing is our main concern, we force peers to enact their queries only in one of the realities, selected randomly and independently. Nonetheless, data insertions and deletions apply to all realities; otherwise the system is inconsistent, in that identical requests in different realities would yield different results. However, no other special consideration has been made, other than creating and maintaining redundant overlays. Hence, this comes at a cost of a fixed replication factor. In effect, along with overweight areas of space, the underweight ones are replicated as well. Our *local allocation* method assigns the nodes of the currently heaviest peer and its associated peers to the lightest peer among that peer-set, with respect to their summing load in all previous realities. The redundancy introduced in both schemes is fixed and equal to the number of realities  $g$ , and we expect it is kept in  $O(\log n)$ . However, message overhead is rendered obsolete.

---

**Algorithm 1.** Pseudo-Inflationary balancing algorithm

---

```

1: repeat
2:   state = State()
3:    $h = \text{heaviest}()$ 
4:   for  $u$  in  $h.\text{universes}$  do
5:     hosts = minHeap()
6:     nodes = maxHeap()
7:     for  $j$  in  $h.\text{nodes}$  do
8:       nodes.push( $(l_j, j)$ )
9:     end for
10:    hosts.push( $(L_i^{<u}, i)$ )
11:    cache(state,  $h$ )
12:    while not nodes.empty() do
13:       $(L_i, i) = \text{hosts.pop}()$ 
14:       $(l_j, j) = \text{nodes.pop}()$ 
15:      assign( $i, j, u$ )
16:    end while
17:  end for
18: until allocationChange(state)

```

---

The purpose of the *virtual nodes* technique is the even allocation of keys to nodes, by associating keys with virtual nodes, and mapping multiple virtual nodes (with unrelated identifiers) to each peer. This method results in assigning many nodes to a single peer. Intuitively, this will provide a more uniform coverage of the identifier space. For example, if we allocate  $\log n$  randomly chosen virtual nodes to each peer, then each of the  $n$  bins will contain  $O(\log n)$  nodes, with high probability. This does not affect the worst-case path length. On the other hand, the maintenance cost for a peer congregating many nodes (eg., updating routing tables) increases readily. The virtual nodes technique can be applied to any load function and type of skewness, enhancing this way flexibility and functionality. Besides, what load consists of is problem specific. In



principle,  $n$  hosts allocate the  $m = g \times n$  nodes of the overlay, where  $g > 1$ . Hence, the maximum load of the busiest peer equals to the sum of the loads of its comprised nodes, and thus, can never be less than the load of the busiest node. This method is unsuitable for very large networks, since the fact that max process throughput  $\Lambda_{\max}$  has a concave behavior. As a result, it would diminish instead of improve after some specific value of the overlay size. Therefore, given an overlay of specific size, there are certain values of  $g$  that ameliorate the maximum throughput. Thus, this technique provides a network of size  $n$  with the throughput of an overlay of size  $m$ , a property that can be graphically interpreted as a transposition in the  $\Lambda_{\max}$  graph, as messages are routed throughout the larger overlay. More importantly, the virtual nodes technique is devoid of redundancy. A peer may contain many nodes but each node can be hosted in one peer exclusively (many-to-one scheme). Our *local allocation* scheme uses a peer's limited knowledge. We select a peer  $i$  that invokes a balancing procedure among all peers in  $i$ 's routing table. At first, all peers participating in the process deploy their nodes. Then, we convey all nodes one-to-one and we assign the heaviest node to the lightest peer. In particular, we maintain a priority queue with all peers and their loads, and for each node  $u$  we successively pop the lightest peer  $p$  and we re-insert it with its new load-value  $\text{Load}(p) + \text{Load}(u)$ , due to the node assignment of  $u$  to  $p$ . Whether we select  $i$  at random, or due to its heavy load, affects only the number of iterations that the process will need in order to converge.

---

**Algorithm 2.** Inflationary balancing algorithm
 

---

```

1: repeat
2:   state = State()
3:   hosts = minHeap()
4:   nodes = maxHeap()
5:    $h = \text{heaviest}()$ 
6:   for  $i$  in  $h.\text{routingTable}$  do
7:     for  $j$  in  $i.\text{nodes}$  do
8:       nodes.push( $(l_j, j)$ )
9:     end for
10:    hosts.push( $(0, i)$ )
11:    cache(state,  $i$ )
12:  end for
13:  for  $j$  in  $h.\text{nodes}$  do
14:    nodes.push( $(l_j, j)$ )
15:    hosts.push( $(0, h)$ )
16:  cache(state,  $h$ )
17:  end for
18:  while not nodes.empty() do
19:     $(L_i, i) = \text{hosts.pop}()$ 
20:     $(l_j, j) = \text{nodes.pop}()$ 
21:    hosts.push( $(L_i + l_j, i)$ )
22:    assign( $j, i$ )
23:  end while
24: until allocationChange(state)

```

---

The *replication* technique aims at alleviating bottlenecks, by imposing additional redundancy, and distributing load of hotspots among more than one hosts, enhancing high availability and fault tolerance. Obviously, this method aims exclusively at task-skew. Our approach, instead of replicating single popular data tuples, focuses on how we can replicate nodes, manage them efficiently, and preserve consistency among all replicas of an overlay node. When replicating hotspots, only a portion of the original node’s traffic reaches each copy, and as a result heavy nodes are alleviated. According to our paradigm, we replicate nodes responsible for popular areas to as many hosts is needed, so that there is no task-skew among hosts. Hence, replication factor varies from node to node, with respect to their load. Our *local allocation* is based exclusively on a peer’s limited knowledge about the network. At each iteration we select a random peer and we redistribute all linked peers to the associated nodes. This is an extremely flexible and effective solution as we assign hosts to nodes in such a way that overlay load imbalances render obsolete (one-to-many).

---

**Algorithm 3.** Deflationary balancing algorithm
 

---

```

1: repeat
2:   state = State()
3:   hosts = Vector()
4:   nodes = minHeap()
5:   h = heaviest()
6:   for j in h.routingTable do
7:     hosts.expand(replicas(j))
8:     nodes.push((0, j))
9:     cache(state, j)
10:  end for
11:  hosts.expand(replicas(h))
12:  nodes.push((0, h))
13:  cache(state, h)
14:
15:  while not hosts.empty() do
16:    (Lj, j) = nodes.pop()
17:    assign(j, hosts.pop())
18:    nodes.push(( $\frac{l_j}{len(j.hosts)}$ , j))
19:  end while
20: until allocationChange(state)

```

---

## 4 Experimental Evaluation

In order to assess our methods and evaluate their performance, we performed extensive experiments with workloads of varying dimensionality and skewness.

### 4.1 Setting

Our experimental evaluation consists of two major parts. A static part, whereby different allocations for rigorous balancing techniques are compared, while the latter part

consists of dynamic simulations. In the former part, we are especially interested on how our paradigm performs compared to other types of allocation. Most of the prior works in balancing use a straightforward *naive* approach for random and arbitrary assignment of balls to bins. Consequently, no special consideration is made on how peers allocate nodes. Nevertheless, we are also interested to compare our methods with an *ideal* allocation for each method. More specifically, we make use of a heuristic, greedy allocation based on global knowledge, according to which each time the heaviest ball is assigned to the globally lightest bin; whereas our methods rely exclusively on peer knowledge. Albeit on-line, the aforementioned allocation type is not realistic as each peer has only partial knowledge of the network, substantially small but functional.

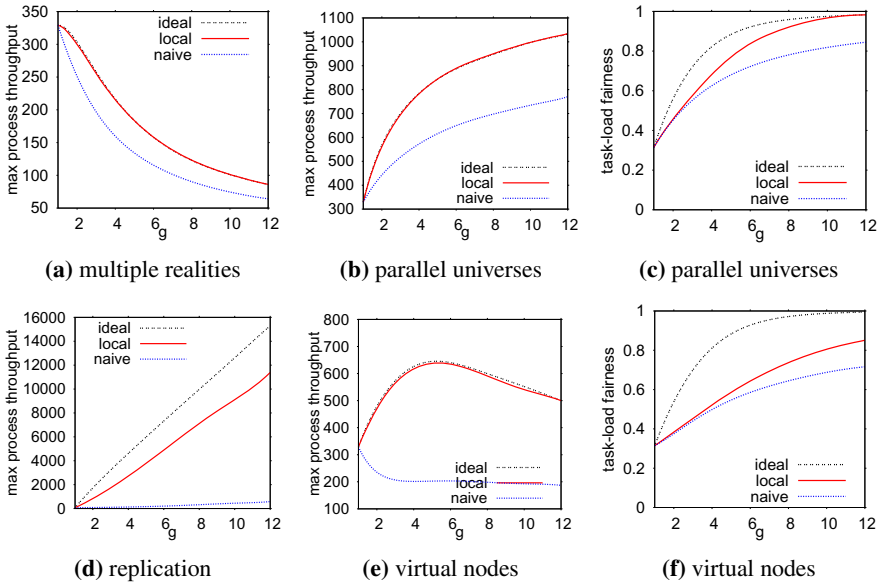
In all experiments, we make use of the PGrid-Z [3] overlay, which incorporates a Z-curve into P-Grid to support multidimensional range search, and we evaluate our methods by various metrics. Latency is the maximum distance in terms of hops from the initial peer to any peer reached during search. Albeit, maximum process throughput  $\Lambda_{\max}$  [3] may be criticized as being too pessimistic, as it only depends on the single most loaded peer, it can be argued that just one overloaded peer will indeed cause trouble for the rest of the network in practice, as being a bottleneck will affect network delay and drop requests in a struggling effort to cope with overwhelming traffic. We are also interested in storage and task load fairness. In particular, we use Jain’s fairness index [5], to measure data- and task-skew. Furthermore, we consider successful a method that is capable of exploiting redundancy to enhance performance. Therefore, we measure redundancy from the most replicated node. In addition, the maximum number of comprised nodes in a single peer is also a significant metric, as a peer that contains many nodes has to deal with high data load and maintenance cost. We also present the communication cost in terms of exchanged data tuples.

We use of datasets describing roads and rivers of Greece. The final dataset consist of 100K tuples. Each queryset consists of 10K range queries, as this type of search is among the most resource consuming. In particular, we define three cluster-centers corresponding to the largest cities of Greece: Athens, Thessaloniki and Patras. All clusters follow normal distributions and querysets consist of queries generated from a cluster chosen with probability proportional to the population of each town. Last, we also created synthetic datasets of variable dimensionality to study the impact of dimensionality.

## 4.2 Results

On the whole there is a significant benefit from using our methods instead of using the straightforward naive approach. Figure 1 illustrates the efficiency of our methods, as ideal allocations, based on global overlay knowledge, perform only slightly better compared to our schemes that rely exclusively on a peer’s partial knowledge about the overlay. The *multiple realities* technique is unsuitable for complex operations where result is returned in fragments over time. Specifically, as shown in Figure 1a, this is the only method where max process throughput  $\Lambda_{\max}$  diminishes with  $g$  for all allocation types. Evidently, enacting a request in all realities in order to improve latency impairs  $\Lambda_{\max}$  dramatically. Most importantly, it fails to improve latency enough. Contrary to what expected, latency was only slightly affected by the number of multiple realities because this technique was designed for lookup queries. In Figure 1b, our *parallel universes*

technique outperforms the former technique in terms of  $\Lambda_{\max}$  and shows an increasing behavior with respect to the number of realities. Clearly, this method has a significant impact on max process throughput. Moreover, we had to compare our paradigm with different allocation methods. Regarding naive allocation, we assume for each reality, nodes are mapped to arbitrary peers. For the ideal allocation we consider again a centralized method that is run for each reality. Specifically, we assign the heaviest node to the lightest peer with respect to a peer's summing load in all realities preceding the one being examined. Then, each peer comprises exactly  $g$  nodes, each participating in a different reality. In addition, task-load fairness ameliorates with  $g$  (Fig. 1c).



**Fig. 1.** Max process throughput and task-load fairness

Regarding the *virtual nodes* technique,  $\Lambda_{\max}$  shows a concave behavior in Figure 1e, due to a combination of two phenomena. For low  $g$  values, where fragmentation is insignificant, fairness beneficial effects are dominant, and thus,  $\Lambda_{\max}$  ameliorates. However, it diminishes as routes become longer for higher  $g$  values. However, congregating numerous nodes increases maintenance costs. Apart from keeping multiple routing tables, peers are burdened with maintenance tasks, such as detecting failures. In addition, latency and precision are affected due to the method's direct interaction with the overlay size. Latency increases with  $g$  as larger overlays result in longer message routes. On the other hand, skewness helps precision because the query clusters were imported in dense areas of the key-space. Hence, when peers join the network by selecting a key with equal probability (data balanced), they populate densely those areas as well. In addition, precision increases with  $g$ , as the overlay nodes become responsible for smaller areas, while our queries have fixed size, and thus, less irrelevant nodes become reached. In effect,

this method infuses the overlay with  $\lambda_{\max}$ , precision and latency from larger overlays with respect to  $g$ . Concerning the naive assignment of nodes to peers, we adopt the randomized strategy for each node to be assigned to any peer with equal probability. For, the ideal centralized allocation scheme the heaviest node is assigned greedily to the globally lightest peer at the time. Clearly, as Figure 1e depicts, our approach outperforms naive assignments as it provides 3.5 times better  $\lambda_{\max}$ . More importantly, ideal allocations are less than 5% more efficient than our scheme for all configurations and  $g$  values.

For *replication*, the greater the redundancy, the less traffic bottlenecks intake. Thereby, imbalances are blunted as overloaded peers are alleviated. Naturally, maximum redundancy increases linearly with  $g$  until fairness is achieved, and withal, it reaches an upper bound beyond which there is no further benefit in load fairness index. Hence, we limit  $g$  to take values smaller than this value. In addition, the invariant latency can be explained as routing takes place along the overlay and not the actual network. Concerning competitor schemes, for the naive allocation, each host replicates a randomly selected node with equal probability. The ideal allocation is a centralized algorithm that probes in sequence all nodes and each time copies the heaviest node to an available host. Figure 1d shows that there are immense benefits of using our method instead of the corresponding naive allocation. It also reveals that this is the most efficient technique in terms of  $\lambda_{\max}$ , as it selectively replicates hotspots.

## 5 Conclusions

To recapitulate, the virtual nodes technique improves performance for certain configurations due to its combined effects, without imposing additional redundancy. Nonetheless, this method accumulates data load from all comprised nodes. The multiple realities technique is considered appropriate exclusively for lookup queries as complex queries impair performance dramatically; whereas our parallel universes paradigm enhances performance significantly. Replication is a flexible method that does not impose fixed additional redundancy, and allows the network to adapt to any load distribution.

## References

1. Avidor, A., Azar, Y., Sgall, J.: Ancient and new algorithms for load balancing in the p norm. *Algorithmica* 29(3), 422–441 (2001)
2. Bharambe, A.R., Agrawal, M., Seshan, S.: Mercury: supporting scalable multi-attribute range queries. In: *SIGCOMM*, pp. 353–366 (2004)
3. Blanas, S., Samoladas, V.: Contention-based performance evaluation of multidimensional range search in p2p networks. In: *InfoScale 2007*, pp. 1–8 (2007)
4. Ganesan, P., Bawa, M., Garcia-molina, H.: Online balancing of range-partitioned data with applications to peer-to-peer systems. In: *VLDB*, pp. 444–455 (2004)
5. Jain, R., Chiu, D., Hawe, W.: A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *DEC TR-301* (1984)
6. Karger, D.R.: Simple efficient load balancing algorithms for peer-to-peer systems. In: *ACM SPAA*, pp. 36–43 (2004)
7. Rao, A., Lakshminarayanan, K., Surana, S., Karp, R.M., Stoica, I.: Load balancing in structured p2p systems. In: *IPTPS*, pp. 68–79 (2003)
8. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: *SIGCOMM 2001*, pp. 161–172 (2001)

# Caching Stars in the Sky: A Semantic Caching Approach to Accelerate Skyline Queries

Arnab Bhattacharya<sup>1</sup>, B. Palvali Teja<sup>2</sup>, and Sourav Dutta<sup>3</sup>

<sup>1</sup> Computer Science and Engineering, Indian Institute of Technology, Kanpur, India  
arnabb@iitk.ac.in

<sup>2</sup> Amazon Development Limited, Hyderabad, India  
palvali.teja@gmail.com

<sup>3</sup> IBM Research Laboratory, New Delhi, India  
sodutta3@in.ibm.com

**Abstract.** Although multi-criteria decision making has emerged with the advent of skyline queries, processing such queries for high dimensional datasets remains a time consuming task. Real-time applications are thus infeasible, especially for non-indexed skyline techniques where the datasets arrive online. In this paper, we propose a caching mechanism that uses the semantics of previous skyline queries to improve the processing time of a new query. In addition to exact queries, such special semantics allow accelerating related queries. We achieve this by generating partial results guaranteed to be in the skyline sets. We also propose an index structure for efficient organization of the cached queries that improve the efficiency. Experiments show the efficiency and scalability of our proposed methods.

## 1 Introduction

To address the problem of multi-criteria decision making and user preference queries over attributes in relations having no clear preference function, Börzsönyi et al. [2] introduced skyline queries. The classic example of a skyline query involves choosing hotels that are good in terms of two attributes, price and distance to beach. The query discards hotels that are both dearer and farther than a skyline hotel. Formally, for every attribute, there is a preference function that states which values dominate.

Efficient INDICES are difficult to build on relations available only at run-time on-the-fly [12]. Hence, skyline queries suffer from large processing time and I/O bottleneck. Caching techniques improve the situation to some extent. However, the use of traditional caching techniques do not promise significant improvement for skyline queries as user interests are unpredictable and an inexact query with even a slight modification where preferences are over a different subset of attributes, results in a cache miss. For example, consider the following skyline queries:

```
select * from Airlines skyline of Duration min, Cost min, Services max  
select * from Airlines skyline of Duration min, Cost min
```

The second query can be answered completely from the cache if the results of the first are stored and intelligent semantic caching techniques are applied.

The special *semantics* of skyline queries allow such similar or related queries to be processed mostly from the cache using the results of the previous queries, *without accessing the database*. Although not all skyline queries can be handled so efficiently, the use of intelligent semantic caching significantly accelerates them by producing partial results.

Our contributions in this paper are as follows:

1. We introduce the concept of semantic caching for skyline queries (Section 3).
2. We categorize a new skyline query into four types according to the content in the cache and design efficient algorithms to process each of them (Section 3).
3. We design an index structure to organize past skyline queries in the cache and show how this helps in searching the cache for processing the new query (Section 4).

## 2 Background and Related Work

Consider a relation  $R$  with preferences specified for  $k$  attributes of  $R$ . A tuple  $r_i = (r_{i1}, r_{i2}, \dots, r_{ik})$  *dominates* another tuple  $r_j = (r_{j1}, r_{j2}, \dots, r_{jk})$  (denoted by  $r_i \succ r_j$ ) if for all  $k$  attributes,  $r_{ic}$  is preferred or equal to  $r_{jc}$ , and for at least one attribute  $d$ ,  $r_{id}$  is *strictly* preferred to  $r_{jd}$ . The *preference functions* for each attribute are specified as part of the skyline query. A tuple  $r$  is said to be in the *skyline* set of  $R$  if there does not exist any tuple  $s \in R$  that dominates  $r$ .

Skyline queries have been imported to databases from the maximum vector problem or Pareto curve [8] in the field of computational geometry. BNL [2] uses a nested loop approach by repeatedly reading the set of tuples. SFS [3] improves it by sorting the data based on a monotone function. LESS [6] combines the best features of these external algorithms; however, its performance depends on the availability of pre-sorted data. Using index structures, divide-and-conquer (NN [7]) and branch-and-bound (BBS [9]) approaches have been proposed.

The idea of caching query results to optimize subsequent query processing was first studied in [5]. General purpose algorithms for semantic caching [4][10] and dynamic caching policies [11] have also been proposed.

Several intelligent structures, e.g., SkyCube [14] and compressed skycubes [13], have been proposed to efficiently compute related skyline queries. However, complete construction of these structures are inefficient for real-time applications. Moreover, the entire cube may not fit in the limited cache size. In this paper, we propose novel and intelligent semantic caching algorithms using an indexing scheme.

## 3 Capturing Semantics of Skyline Queries

In this section, we characterize a skyline query in terms of previous skyline queries, which help relate the new query to those in the cache.

### 3.1 Characterization of Queries

We assume that the skyline queries are for a single relation that maintains the *distinct value condition* [14], which states that if no two data points have the same values for

**Table 1.** Characterization of queries

Cache	$S_1 = \{1, 2, 3\}, S_2 = \{1, 2\}, S_3 = \{3, 4\}, S_4 = \{5, 6\}$			
Query	Exact	Subset	Partial	Type
$Q_1 = \{1, 2\}$	$S_2$	$S_1$	$S_1, S_2$	Exact
$Q_2 = \{2, 3\}$	-	$S_1$	$S_1, S_2, S_3$	Subset
$Q_3 = \{4, 5\}$	-	-	$S_3, S_4$	Partial
$Q_4 = \{6, 7\}$	-	-	$S_4$	Partial
$Q_5 = \{7, 8\}$	-	-	-	Novel

all the dimensions, then the skylines for dimension set  $A$  is a subset of skylines for dimension set  $B$  when  $A \subset B$ . Each query is represented as the set of attributes of skyline preferences, which we assume is not altered for a particular dimension. This assumption holds since user preferences are generally the same.

Given a cache  $C$  modeled as a set of queries  $\{S_1, S_2, \dots, S_n\}$ , where each  $S_j$  is again a set of attributes, a new query  $Q = \{a_1, a_2, \dots, a_q\}$  can be characterized into *at least* one of the following groups:

1. **Exact Query:**  $Q$  is an *exact* query if it matches exactly with a cached query, i.e.,  $\exists S_j, Q = S_j$ , indicating the re-occurrence of a previous query.
2. **Subset Query:**  $Q$  is a *subset* query if all its attributes are completely contained in a cached query, i.e.,  $\exists S_j, Q \subset S_j$ .
3. **Partial Query:**  $Q$  is a *partial* query if some of its attributes are subsets of a cached query, i.e.,  $\exists Q' \subset Q, \exists S_j, Q' \subseteq S_j$ .
4. **Novel Query:**  $Q$  is a *novel* query if none of its attributes are cached, i.e., if  $\forall a_i \in Q, \forall S_j, a_i \notin S_j$ .

Section 3.3 depicts the importance of hierarchy of categorization in query processing. The *most* restrictive category determines the type of the query. For example, if a query is both an exact and a subset query, it is treated as an exact query and a query is categorized as a novel query if and only if it cannot be characterized as an exact, subset or partial query. Table 1 describes an example in detail. When a new skyline is queried, the cached queries are scanned to determine its type.

### 3.2 Semantic Segments

While each cached semantic query is a set of attributes, certain other descriptors for the query are also encapsulated in a data structure called the *semantic segment*:

- (i) *Attributes and preferences:* Attributes on which the skyline preferences are applied,
- (ii) *Result:* A link to a table of records that constitute the answer to this query, and
- (iii) *Replacement value:* It is used for cache replacement methods (see Section 4.5).

### 3.3 Query Processing Algorithms

Based on the type of the new query, different query processing strategies are followed as described in this section.



**Exact Queries:** If the query is an exact query, the result set of the cached query is directly returned as the result set of the new query.

**Subset Queries:** If the new query  $Q$  is a subset of a cached query  $S_j$ , then the following lemma<sup>1</sup> shows that the result set of  $Q$  is a subset of the result set of  $S_j$ .

**Lemma 1.** *If a skyline query  $Q$  is a subset of another skyline query  $S$ , then the result set of  $Q$  is completely contained in the result set of  $S$ .*

The next lemma shows that to determine whether a tuple from the result set of  $S_j \supset Q$  is in the result set of  $Q$ , only the tuples in  $S_j$  need to be checked for dominance.

**Lemma 2.** *If a tuple  $v$  in the result set of  $S$  is not a skyline for  $Q \subset S$ , then there must exist  $u \in \text{result}(S)$  such that  $u \succ v$ .*

Hence,  $u$  will be in the result set of  $Q$  only if none of the tuples in the result set of  $S_j$  dominate  $u$ .

If a new query  $Q$  is a subset of many cached queries  $S_i, S_j$ , etc., any tuple in the result set of  $Q$  must also be in the result set of all of  $S_i, S_j$ , etc. Thus, only the tuples in the *intersection* of the result sets of these subset queries need to be examined.

Thus, subset and exact queries can be processed from the cache itself. The advantage, however, cannot be retained for the other two types of queries as explained next.

**Partial Queries:** Suppose the new query  $Q$  is partial to a cached query  $S_j$ . The attribute set  $Q' \subset Q$  is equal to  $S'_j \subseteq S_j$ . Using Lemma 1, the skyline set corresponding to the attributes  $Q' = S'_j$  is a subset of the skyline maintained for  $S_j$ . This serves as the *base set*. When  $Q$  is a *superset* of  $S_j$ , the *entire* skyline set of  $S_j$  serves as the base set.

However, the computation of the base set does not complete the processing of  $Q$ . The following lemma shows there may exist a tuple not in the base set (i.e., the skyline set for  $Q'$ ), but is part of the skyline set of  $Q$ .

**Lemma 3.** *A tuple in the skyline set of  $Q$  need not be in the skyline set of its subset  $Q'$ .*

Thus, the base set alone is not sufficient and it is necessary to query from the database. However, the base set helps in two important ways.

First, since the tuples in the base set are guaranteed to be in the skyline set of  $Q$ , they can be output immediately. For real-time applications, the implications of this concept of *incremental* results are enormous. The other skyline tuples can be simultaneously computed from the database.

The second important advantage is the fact that the base set fits in the cache and can serve as the initial skyline window, thereby speeding up generic skyline algorithms, such as BNL [2], SFS [3], and LESS [6]. For other non-indexed algorithms, the base set may or may not help, but will never deteriorate the performance.

For two or more queries  $S_i, S_j$ , etc. partial to  $Q$ , the set computed from the *union* of their base sets serves as the combined base set. Since this set is larger, the advantages are more pronounced.

<sup>1</sup> The proofs of the lemmas are omitted due to space constraints; please refer to [1].

**Novel Queries:** Since these queries contain no attributes common to previous skyline queries, the cache does not contain any information to expedite the processing, and are, thus, completely processed from the database.

### 3.4 Need for an Index Structure

Processing a new query first involves searching the semantic segments in the cache to determine its type. However, the number of semantic segments is exponential in number of dimensions, and therefore, can be time-consuming for high-dimensional datasets.

An even bigger concern when the semantic segments are unorganized is the redundancy of storage. Consider queries  $S_i$  and  $S_j$  where  $S_j \subset S_i$ . The skyline tuples of  $S_j$  are already stored as skyline of  $S_i$  and, hence, wastes precious memory. Efficient organization of the semantic segments in the cache is thus required.

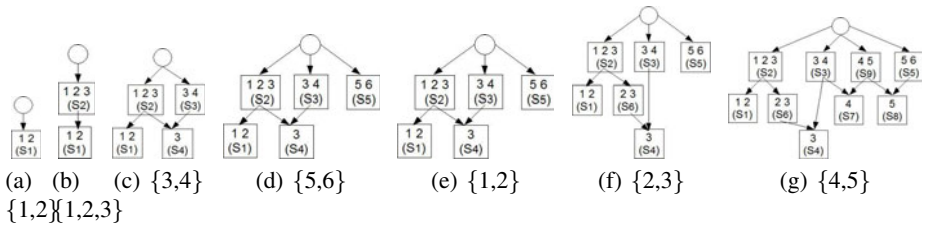


Fig. 1. Querying and insertion of semantic segments in the index

## 4 Index Structure

The index structure we propose is a directed acyclic graph (DAG) linking the different semantic segments. The semantic segment for a query  $S_1$  is made a child of query  $S_2$  if  $S_1 \subset S_2$ . Clearly, there cannot be any cycle, although a segment can have multiple parents. Since the graph may be disconnected, a pseudo root node is added that acts as the parent of all root nodes. Compared to SkyCube based structures, this does not store the entire user query space and is based only on the queries previously encountered.

### 4.1 Modified Semantic Segments

The index structure, in addition to the fields described in Section 3.2, store two more fields in each semantic segment for efficient management: (iv) *Child pointers*, and (v) *Bit vectors*.

The child pointers link a semantic segment to its children. For efficient retrieval, each attribute of the query maintains a bit vector of the size of the number of children. The children are ordered according to their arrival. The  $i^{\text{th}}$  bit in the  $j^{\text{th}}$  bit vector is set to 1 if and only if the  $i^{\text{th}}$  child contains the  $j^{\text{th}}$  attribute.

## 4.2 Eliminating Redundancy of Result Sets

Our skyline query processing algorithm uses the index structure to eliminate the redundancy of cached query result sets. If a query has a child (i.e., a subset), all the skyline tuples are not stored in the result set; rather, they are distributed between itself and the child. For example, suppose query  $S_1$  has a child  $S_2$ , which is a leaf node. The skyline tuples for  $S_2$  are stored in its result set, i.e.,  $r(S_2) = s(S_2)$ . However, since these records are a subset of the skyline tuples for  $S_1$ , redundancy is removed by not storing them again in  $S_1$ . Instead, only the difference of the skyline set for  $S_1$  with  $S_2$  are stored, i.e.,  $r(S_1) = s(S_1) - s(S_2)$ . The complete skyline records for  $S_1$  can thus be retrieved by combining the result set of all its children.

## 4.3 Query Processing and Insertion Using Index

We illustrate the index search operation for query processing and subsequent insertion using the series of query examples as shown in Fig. 11 (only attributes are shown).

Initially, the cache is empty and the index contains the pseudo root node. When the first query  $\{1, 2\}$  arrives, it is classified as a novel query, and is inserted as semantic segment  $S_1$  (Fig. 11a).

The next query is  $\{1, 2, 3\}$ . All the root nodes are searched to find that it is a partial query (superset of  $S_1$ ), and the entire skyline set of  $S_1$  is used as the base set. The new query now becomes the root and the old root its child (Fig. 11b).

Then, query  $\{3, 4\}$  arrives. Scanning the root nodes, it is found to be partial to  $S_2$  and the base set consisting of the skyline tuples of the common attributes,  $\{3\}$  is computed. This semantic segment ( $S_4$ ), being a subset of both  $S_2$  and the new query  $S_3$ , is maintained as a child of both (Fig. 11c).

The next query  $\{5, 6\}$  is a novel query as it does not match with any of the root nodes. Consequently, it is processed from the database and is inserted as a new root node in the index (Fig. 11d).

The next query  $\{1, 2\}$  is first categorized as a subset query of  $S_2$ . The children of  $S_2$  are then searched to improve the categorization to exact. The skyline set of  $S_1$  is reported and the index remains unmodified (Fig. 11e).

Query  $\{2, 3\}$  then arrives. Being a subset of  $S_2$ , its children are searched, but no exact match is found. The skyline set of  $\{2, 3\}$  is computed from that of  $\{1, 2, 3\}$  and inserted as a child of  $S_2$ . Since the skyline set of  $\{3\}$  is already maintained as a semantic segment ( $S_4$ ), and is a subset of this new query as well, the child pointers and bit vectors are modified in  $S_2$  and  $S_6$ , making  $S_4$  a descendant of  $S_2$  (Fig. 11f).

Query  $\{4, 5\}$  is similarly handled (Fig. 11g).

**Table 2.** Experimental parameters and their default values (in bold)

Parameter	Values
Cardinality ( $N$ )	$1 \times 10^4, 3 \times 10^4, \mathbf{1 \times 10^5}, 3 \times 10^5, 1 \times 10^6$
Dimensionality ( $d$ )	3, 4, 5, <b>6</b> , 7
Cache size ( $ C $ )	0.1%, 1%, 3%, <b>5%</b> , 7%, 10%
Number of queries ( $ Q $ )	1, 5, 10, 25, 50, <b>100</b>

### 4.4 Deletion from Index

When the cache is full and a new query arrives, an effective replacement policy must be chosen to select the replacement candidate. Further, since the cache is very dynamic, efficient update operations on the index need to be designed.

The skyline set of a parent in the index is shared among itself and its children. Therefore, if a child is deleted from the cache, for correctness, its skyline set needs to be merged back with that of its parent, and hence deleting a child does not produce much space advantage. Thus, for our index structure, only root nodes are deleted and the children having no parents become roots.

### 4.5 Cache Replacement

Due to limited cache size, not all semantic segments may be stored. This is the main drawback of SkyCube-based techniques. Thus, for efficient use of cache, the most useful semantic segments need to be preserved during replacement.

The first important parameter is the *usage factor* ( $\alpha$ ). When the semantic segment is first inserted into the index, its replacement factor is set to 1. Every time its result set is used, the value is incremented. The one with a lower replacement factor should be replaced, as it is used less.

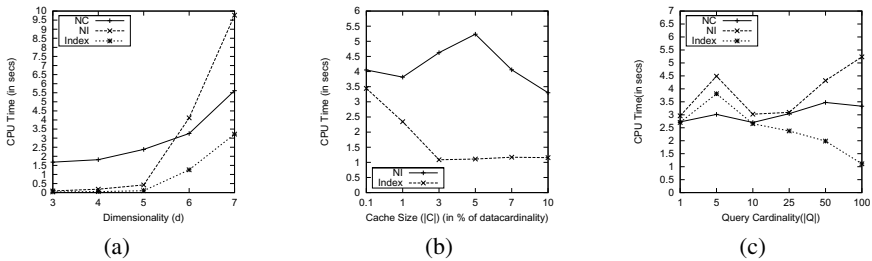


Fig. 2. Effect of (a) dimensionality, (b) cache size, and (c) query cardinality

The second important factor is the *size* ( $\beta$ ) of the skyline set, i.e., the number of tuples in it. A semantic segment that stores a large number of tuples as its skyline set does not allow other semantic segments to be stored. Hence, it should be removed.

The third parameter is the *dimensionality* ( $d$ ). When the number of dimensions is more, there is more chance of a new query to become partial to or subset of it, and hence, should not be replaced.

A *replacement value* ( $\delta$ ) for each semantic segment is computed by combining the three, i.e.,  $\delta = f(\alpha, \beta, d)$ . The semantic segment with the *lowest*  $\delta$  is the *least useful* and should be chosen for replacement. The function  $f$ , therefore, should be monotonic with  $\alpha$  and  $d$  and anti-monotonic with  $\beta$ . While different functions fit the condition, we use the following simple function:  $\delta = (\alpha \times d) / \beta$ .

## 5 Experimental Results

In this section, we evaluate the performance of the caching techniques. The techniques were implemented using Java on an Intel Core 2 Duo 2GHz machine with 2GB RAM in Ubuntu Linux environment. For skyline computation, we used the non-indexed sort-filter-skyline (SFS) [3] algorithm. We analyzed and compared the execution times of three different skyline processing techniques: (i) without using cache (NC), (ii) using cache without using the index (NI), and (iii) using cache with index (Index).

We used the standard data generator for skyline queries from <http://www.pgfoundry.org/projects/randdataset> to generate synthetic datasets; the dimensions were chosen to be independent. The scalability and performance of the techniques on synthetically generated data were measured against four different parameters: (i) cardinality of the dataset ( $N$ ), (ii) dimensionality of the dataset ( $d$ ), (iii) size of the cache ( $C$ ), and (iv) number of queries ( $Q$ ). The values of these parameters were varied according to Table 2. To study the effects, each parameter was varied keeping the others constant.

The first experiment is on varying dimensionality. The indexed method performs the best. For large dimensional datasets, the caching method without indexing performs worse than the no caching method as the number of semantic segments is too large (Fig. 2a). The next experiment on cardinality of datasets shows that the caching method without indexing suffers for small sized datasets due to the large overhead of searching through unorganized semantic segments in the cache (figure omitted due to space constraints; please see [1]).

When the cache size increases, more semantic segments are stored, thereby reducing the query time. The non-indexing method initially suffers due to unorganized segments, but improves as more queries get classified as exact or subset queries (Fig. 2b).

When there is enough space in the cache to store all possible skyline queries, any new query should be answered very fast. The final set of experiments (Fig. 2c) measures the average query time as more queries arrive. When no caching is used, there is little effect. When no indexing is used in the cache, performance suffers due to unorganized semantic segments. The indexing method initially suffers from index construction overhead, but progressively improves due to the superior arrangement of semantic segments in the cache. The performance on real datasets showed similar trends (please see [1]).

## 6 Conclusions

In this paper, we introduced the concept of semantic caching to accelerate a skyline query by classifying it as one of the four types—exact, subset, partial and novel. While the exact and subset queries are processed directly from the cache, partial results for partial queries can be output from the cache before resorting to the database for the full skyline set. We also proposed an index structure to effectively organize the past queries in the cache, thereby improving the efficiency. In future, we plan to handle update-intensive databases.

## References

1. Bhattacharya, A., Teja, B.P., Dutta, S.: Caching stars in the sky: A semantic caching approach to accelerate skyline queries. In: arXiv:1106.1811 [cs.DB] (2011)
2. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430 (2001)
3. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: ICDE, pp. 717–719 (2003)
4. Dar, S., Franklin, M.J., Jónsson, B.T., Srivastava, D., Tan, M.: Semantic data caching and replacement. In: VLDB, pp. 330–341 (1996)
5. Finkelstein, S.: Common expression analysis in database applications. In: SIGMOD, pp. 235–245 (1982)
6. Godfrey, P., Shipley, R., Gryz, J.: Maximal vector computation in large data sets. In: VLDB, pp. 229–240 (2005)
7. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: an online algorithm for skyline queries. In: VLDB, pp. 275–286 (2002)
8. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. *J. ACM* 22(4), 469–476 (1975)
9. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: SIGMOD, pp. 467–478 (2003)
10. Ren, Q., Kumar, V.: Semantic caching and query processing. *IEEE Trans. on Knowledge and Data Engineering* 15, 192–210 (2003)
11. Sacharidis, D., Bouros, P., Sellis, T.K.: Caching dynamic skyline queries. In: Ludäscher, B., Mamoulis, N. (eds.) *SSDBM 2008*. LNCS, vol. 5069, pp. 455–472. Springer, Heidelberg (2008)
12. Sun, D., Wu, S., Li, J., Tung, A.K.H.: Skyline-join in distributed databases. In: *ICDE Workshops*, pp. 176–181 (2008)
13. Xia, T., Zhang, D.: Refreshing the sky: The compressed skycube with efficient support for frequent updates. In: SIGMOD, pp. 491–502 (2006)
14. Yuan, Y., Lin, X., Liu, Q., Wang, W., Xu Yu, J., Zhang, Q.: Efficient computation of the skyline cube. In: VLDB, pp. 241–252 (2005)

# Generating Synthetic Database Schemas for Simulation Purposes

Carlos Eduardo Pires<sup>1</sup>, Priscilla Vieira<sup>1</sup>, Márcio Saraiva<sup>1</sup>,  
and Denilson Barbosa<sup>2</sup>

<sup>1</sup> Federal University of Campina Grande, Computer Science Depart.,  
Campina Grande, PB, Brazil

{cesp,vieira,marcio}@dsc.ufcg.edu.br

<sup>2</sup> University of Alberta, Depart. of Computing Science, Edmonton, Alberta, Canada  
denilson@cs.ualberta.ca

**Abstract.** To simulate query answering in Peer Data Management System (PDMSs), simulators need to associate a database schema to each peer in the overlay network. Finding or creating a high number of database schemas can be a time consuming and tendentious task. This work proposes an automatic process to generate multiple synthetic database schemas with semantically coherent variations of a given base schema. The schemas are obtained through applying different types of modifications to subsets of the base schema. Our experimental validation has shown that the proposed method is able to produce random schemas that can be used in realistic simulations.

**Keywords:** Synthetic Database schema, Ontology, PDMS, Simulation, Random.

## 1 Introduction

Peer Data Management Systems (PDMSs) [1, 2, 3] are advanced P2P applications in which each peer is an autonomous data source that makes available a local schema. Peers manage their data locally, revealing part of their internal schemas to other peers. Schema mappings are generated to allow information exchange between peers. PDMS overlay networks tend to be large and allow complex interactions between the physical machines, underlying network, application, and user [4]. The testing of a PDMS overlay network or protocol in a realistic environment is often a complex and costly undertaking. Hence, simulation is the most popular technique for investigating overlay networks and PDMS applications [5].

Researchers interested in simulating a PDMS overlay network tend to avoid the development of a complex simulator and focus on some specific issue, e.g. query answering. Concerning that issue, PDMS simulators need to associate a local database schema to each peer in the simulated network. To make sense, the local schemas should belong to the same application domain. Depending on the domain that is considered, it can be easy to find a small number of related schemas. For instance, after a quick navigation in the *Database Answers* web site [6], we have found 13 database schemas related to the *education* domain.

However, to simulate a realistic and large-scale PDMS environment, one often needs a large number of schemas. Moreover, these synthetic schemas must belong to

a common domain and exhibit sufficiently large overlap (so as to allow the definition of mappings between them). On the other hand, there must also be some kind of heterogeneity among the synthetic schemas in order to better test the strengths and weaknesses of the simulated approaches. One possible solution is to generate these schemas manually, which is time consuming and error-prone, and thus does not scale. Instead, this work proposes an automatic process to generate a large number of database schemas to be used in PDMS simulations.

As shown in Figure 1, the proposed process consists in, given a base schema  $S$ , automatically generating multiple synthetic database schemas  $S_1, S_2, \dots, S_j$ , where each synthetic schema  $S_j$  corresponds to a modified subschema of  $S$ . We use two kinds of operations to modify the new schemas: *structural changes* and *element renaming*. Structural changes ensure that: (i) synthetic schemas will have a different number of elements; (ii) elements will differ in the number of properties; and (iii) properties will have different data types. Element renaming guarantees that elements in synthetic schemas will have a different label which is semantically similar to the original one. Each operation is performed by a different schema modifier algorithm. The process is extensible in the sense that new schema modifiers can be added. Some schema modifiers use a domain ontology as an external resource. Parameters are used to influence the characteristics of the synthetic schemas to be produced. The following sections offer a detailed description of how synthetic schemas are produced and modified.

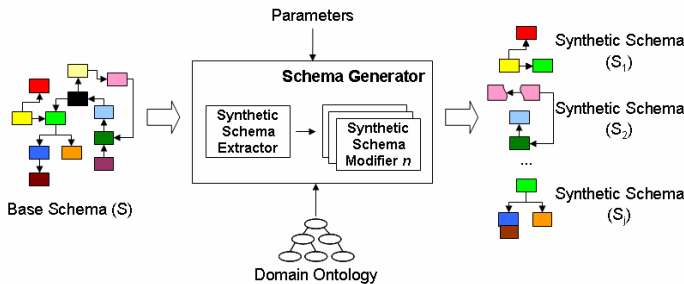


Fig. 1. Generating and modifying synthetic database schemas

## 2 Building Synthetic Schemas

To meet diverse application requirements, the schema generation process accepts several types of parameters. Depending on the parameters values that are provided, different synthetic schemas  $S_1, S_2, \dots, S_j$  can be produced for the same base schema  $S$ . The input parameters include: (i) *Base schema*: reference schema used as a basis to produce synthetic database schemas; (ii) *Number of schemas*: quantity of synthetic schemas to be generated as output; (iii) *Schema size*: number of elements that each schema should contain; (iv) *Schema format*: to simplify matters, the synthetic schemas are represented in the relational format; and (v) *Modifying operations*: different types of operations that can be used to modify the schemas.

Figure 2 depicts the proposed algorithm to generate multiple synthetic schemas. It accepts the general parameters (line 1) described in the previous subsection and



returns a collection of synthetic schemas (line 14). Each synthetic schema  $S_j$  is created incrementally: elements are selected from the base schema and added to  $S_j$  one at a time (lines 6-9). To guarantee the generation of ad-hoc schemas we use a random function to pick up elements in the base schema. Each selected element must maintain a relationship with at least one of the elements that were already included in the current synthetic schema. Such requirement is explained because, during the generation of a synthetic schema, if a new element is simply selected and added to the synthetic schema (i.e. ignoring its relationships with the other elements in the base schema), then manual intervention would be need to link the new element with the elements that were already added to the synthetic schema. Once a synthetic schema is built it is modified through the operations provided as input (line 10). These operations are detailed in Section 3.

```

1  GenerateSyntheticDatabaseSchemas (input: Parameters; output: CollectionSyntheticSchemas) {
2  Counter ← 0;
3  While Counter ≤ Parameters.NumberSchemas Do
4  SyntheticSchema[Counter].Size ← 0;
5  SyntheticSchema[Counter].Elements ← GetElement(Parameters.BaseSchema.Elements, Random);
6  While SyntheticSchema[Counter].Size ≤ Parameters.SchemaSize Do
7  SyntheticSchema[Counter].Elements ← SyntheticSchema[Counter].Elements +
   GetElement(SyntheticSchema[Counter].Elements.ReferencedElements, Random) -
   SyntheticSchema[Counter].Elements;
8  SyntheticSchema[Counter].Size++;
9  End While;
10 ModifySyntheticSchema(SyntheticSchema[Counter], Parameters.ModifyingOperations);
11 SyntheticSchemaCollection ← SyntheticSchemaCollection + SyntheticSchema[Counter];
12 Counter++;
13 End While;
14 Return(SyntheticSchemaCollection); }

```

Fig. 2. Algorithm for generating and modifying synthetic database schemas

We use the Internet Movie Database (IMDb) [7] both to illustrate our ideas and for experimental purposes. The IMDb schema is used as the base schema and corresponds to a graph consisting of 60 relations. An excerpt from the IMDb schema is shown in Figure 3.

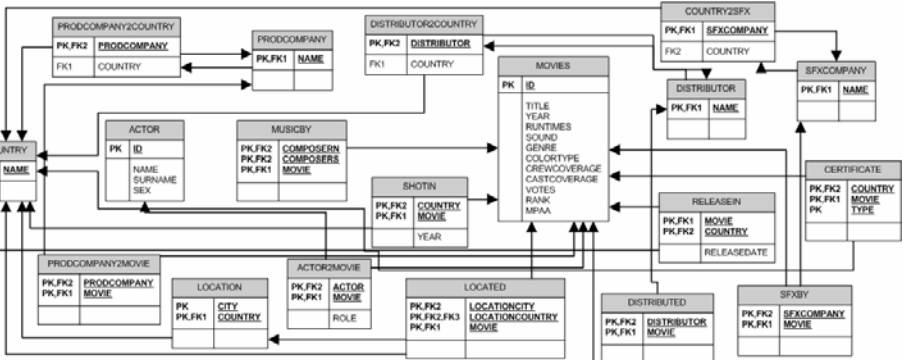


Fig. 3. An excerpt from the IMDb schema

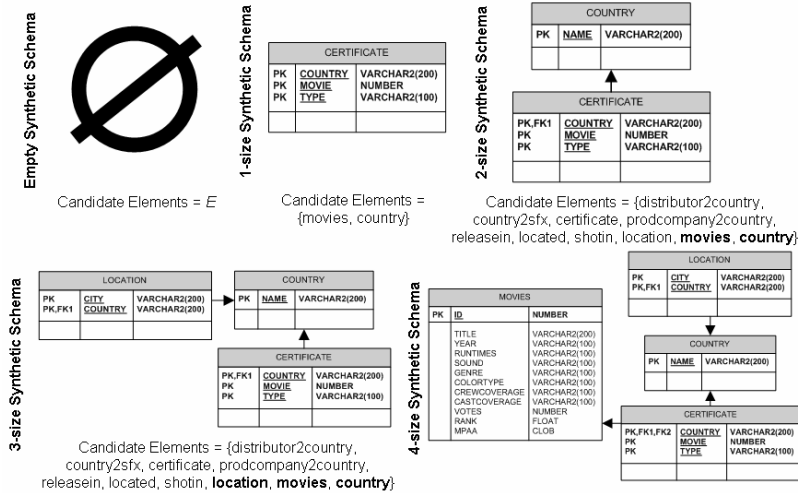


Fig. 4. A step-by-step example illustrating the generation of a 4-size synthetic schema

Figure 4 illustrates an example in which a 4-size synthetic schema has been requested. The first element selected by the random function is *certificate*. The next element to be added to the synthetic schema is necessarily one of the elements that maintain a relationship with *certificate* in the base schema. According to the IMDB schema, the candidate elements are *movies* and *country*. Suppose that *country* is selected by the random function. The other element to be added to the synthetic schema must maintain a relationship with *certificate* and/or *country* in the base schema. Besides *movies*, the candidate elements are *location*, *shotin*, *releasein*, *prodcompany2country*, *country2sfx*, *located*, and *distributor2country*. *Certificate* and *country* are also candidate elements since they maintain a relationship with each other. However, since these elements are already included in the synthetic schema they are discarded. Assuming that *location* is selected, the synthetic schema contains three elements: *certificate*, *country*, and *location*. The current candidate elements are now *movies*, *shotin*, *releasein*, *prodcompany2country*, *country2sfx*, *located*, and *distributor2country*. Finally, consider that the fourth element selected by the random function is *movies*.

### 3 Modifying Synthetic Database Schemas

Assuming that a synthetic schema  $S_j$  has been generated, the possible operations to modify  $S_j$  are described as follows:

**Removal of Properties** - consists in eliminating properties from elements of a synthetic schema  $S_j$ . At each iteration an element is selected from  $S_j$  and one of its properties is removed. In both cases a random function (*Random*) is invoked. To guarantee that each element will keep a minimum percentage of its original properties a parameter (*MinimumPct*) is used. Each element must keep at least one property. The parameter *MaxModifications* determines the number of properties to be removed from the entire synthetic schema. The variable

*Modifications* is incremented whether or not a property is removed. For instance, a property cannot be removed if it is the only remaining property of an element. Figure 5 illustrates the algorithm to remove properties from elements in a synthetic schema.

```

01 RemoveProperties (input: SyntheticSchema, MaxModifications, MinimumPct; output: SyntheticSchema) {
02 Modifications ← 0;
03 While Modifications <= MaxModifications Do
04 Element ← GetElement(SyntheticSchema.Elements, Random);
05 If Element.Properties.Count >= MinimumPct Then
06 Property ← GetProperty(Element.Properties, Random);
07 Element.Properties ← Element.Properties - Property;
08 SyntheticSchema.Elements(Element).Properties ← Element.Properties;
09 End If;
10 Modifications++;
11 End While;
12 Return(SyntheticSchema); }
    
```

Fig. 5. Algorithm to remove properties from elements of a synthetic schema

**Insertion of Properties** - consists in adding semantically related properties to the elements of a synthetic schema  $S_j$ . To this end, a domain ontology  $O$  on the same topic of the base schema must be available to provide the related properties. Figure 6 shows an excerpt from the Movie Ontology [8]. At each iteration an element  $e_i$  is randomly selected from the synthetic schema  $S_j$ . Its label is used as input to identify a corresponding term  $t_i$  in  $O$  as well as terms that are semantically equivalent to  $t_i$ , i.e. the synonyms of  $t_i$ . All properties of  $t_i$  and the ones of its synonyms are considered candidate properties. Among them one is randomly selected and added to  $e_i$ . *String* is used as the default data type for the new property. If a corresponding term  $t_i$  is not found in  $O$ , a new property cannot be inserted.

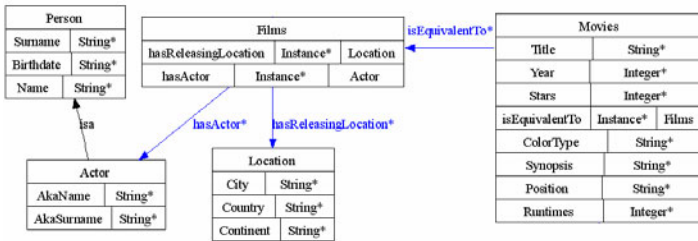


Fig. 6. An excerpt from the Movie Ontology [8]

Figure 7 illustrates the algorithm that allows the addition of properties to the elements of a synthetic schema. The parameter *MaxModifications* determines the number of properties to be added to the elements of the synthetic schema  $S_j$ . To avoid all properties from being added to the same element, the parameter *MaximumPct* is used. It controls the maximum number of properties that can be added to each element. The variable *Modifications* is incremented whether or not a property is added to an element. For instance, a property cannot be added to an element  $e_i$  when  $e_i$  already contains a property with the same label.

**Replacement of Element Label** - consists in replacing the label of an element  $e_i$  in the synthetic schema  $S_j$  by a semantically related label. Again, a domain ontology  $O$  is used as an external resource. At each iteration an element  $e_i$  is randomly selected from  $S_j$ . Its label is

```

01 AddProperties (input: SyntheticSchema, MaxModifications, MaximumPct, DomainOntology; output: SyntheticSchema) {
02   Modifications ← 0;
03   While Modifications <= MaxModifications Do
04     Element ← GetElement(SyntheticSchema.Elements, Random);
05     If Element.Properties.Count <= MaximumPct Then
06       NewProperty.Name ← GetNewProperty(Element.Label, DomainOntology, Random);
07       NewProperty.DataType ← STRING;
08       If NewProperty NOT IN Element.Properties And NewProperty IS NOT null Then
09         Element.Properties ← Element.Properties + NewProperty;
10         SyntheticSchema.Elements(Element).Properties ← Element.Properties;
11       End If;
12     End If;
13     Modifications++;
14   End While;
15   Return(SyntheticSchema); }

```

Fig. 7. Algorithm that adds properties to the elements of a synthetic schema

used as input to search for corresponding term  $t_i$  in  $O$ . All terms having a semantic relationship with  $t_i$  are obtained (i.e. the synonyms, superclasses and subclasses of  $t_i$ ). These terms are candidates to replace the element label. One of them is chosen by a random function (*Random*). Figure 8 depicts the algorithm to replace the label of elements in a synthetic schema. The parameter *MaxModifications* determines the number of labels to be replaced in the schema. The variable *Modifications* is incremented whether or not a label is replaced. For instance, a label cannot be replaced if a corresponding term  $t_i$  is not found in  $O$ .

```

1  ReplaceElementLabel (input: SyntheticSchema, MaxModifications, DomainOntology; output: SyntheticSchema) {
2    Modifications ← 0;
3    While Modifications <= MaxModifications Do
4      Element ← GetElement(SyntheticSchema.Elements, Random);
5      OldLabel ← Element.Label;
6      NewLabel ← GetNewLabel(OldLabel, DomainOntology, {Synonyms, Hypernyms, Hyponyms}, Random);
7      If NewLabel IS NOT null Then
8        SyntheticSchema.Elements(Element).Label ← NewLabel;
9      End If;
10     Modifications++;
11   End While;
12   Return(SyntheticSchema); }

```

Fig. 8. Algorithm to replace an element label

**Replacement of Property Data Type** - consists in replacing the data type of properties by another one which belongs to the same family of the original data type. A data type classification must be available. The classification must be the one supported by the DBMS storing the base schema. Figure 9 shows an excerpt from the data type classification of the Oracle DBMS.

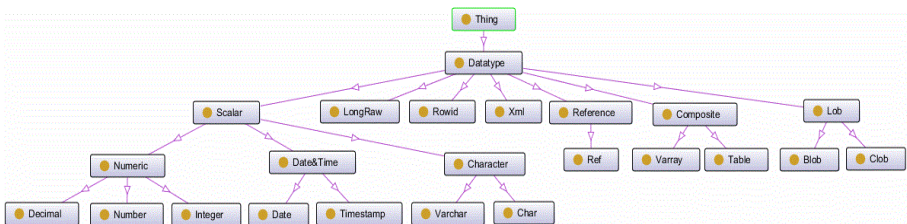


Fig. 9. Data type classification of Oracle

At each iteration an element  $e_i$  is randomly selected from the synthetic schema  $S_j$ . Afterwards, one of the properties  $p_k$  of  $e_i$  is randomly chosen. The data type  $d_k$  of  $p_k$  is used as input to find a corresponding term  $t_i$  in the data type classification. All terms in the same level of  $t_i$  (i.e. the terms with the same superclass of  $t_i$ ) are candidates to replace the original data type. Among them one is randomly chosen. Figure 10 shows the algorithm to modify the data type of the properties in a synthetic schema. The parameter *MaxModifications* determines the number of data types to be replaced in the synthetic schema. The variable *Modifications* is incremented whether or not a data type is replaced. For instance, a data type cannot be replaced if a corresponding term  $t_i$  is not found in the data type classification.

```

1 ReplaceDataType (input: SyntheticSchema, MaxModifications, DataTypeClassification; output: SyntheticSchema) {
2   Modifications ← 0;
3   While Modifications <= MaxModifications Do
4     Element ← GetElement(SyntheticSchema.Elements, Random);
5     Property ← GetProperty(Element.Properties, Random);
6     OldDataType ← Property.DataType;
7     NewDataType ← GetNewDataType(OldDataType, DataTypeClassification, Random);
8     SyntheticSchema.Elements(Element).Properties(Property).DataType ← NewDataType;
9     Modifications++;
10  End While;
11 Return(SyntheticSchema); }

```

Fig. 10. Algorithm to replace the data type of a property

To exemplify, consider the synthetic schema of Figure 4. Assume that the modifying operations will be executed in the following order: (i) removal of properties; (ii) insertion of properties; (iii) replacement of element label; and (iv) replacement of property data type. The Movie Ontology (Figure 6) and the Oracle data type classification (Figure 9) are used as the domain ontology and the data type classification, respectively. Table 1 illustrates the successive steps to modify the synthetic schema. Firstly, the properties *Castcoverage*, *Crewcoverage* and *Runtimes* are removed from *Movies*. The properties *Continent* and *Stars* are added to *Location* and *Movies*, respectively. The element label *Movies* is replaced by *Films*. Finally, the data types of the properties *Type* and *Rank* are replaced by *Char* and *Decimal*, respectively. The modified synthetic schema is shown in Figure 11.

Table 1. The successive steps applied to modify the synthetic schema of Figure 4.

Algorithm	Selected Relation / Relationship	Selected Property	Selected Term(s) / Data Types	Selected Property / Label / Datatype
REMOVE	Movies	Castcoverage	-	-
REMOVE	Movies	Crewcoverage	-	-
REMOVE	Movies	Runtimes	-	-
INSERTION	Location	-	Location, Nation, Place, Country	Continent
INSERTION	Movies	-	Movies, Films	Stars
LABEL	Movies	-	Films	-
DATATYPE	Certificate	Type	Varchar2	Char
DATATYPE	Films	Rank	Number, Integer	Decimal

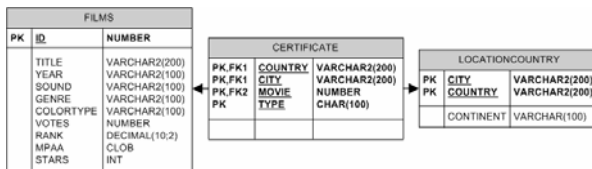


Fig. 11. The resulting synthetic schema after successive applications of modifying operations

## 4 Experiments and Results

The Schema Generator tool (<http://code.google.com/p/schemagenerator2/>) has been developed in Java. Elements, properties, and relationships are selected using the class *Random* available in the package `java.util`. Jena [9] has been used to provide ontology manipulation and reasoning. In this version, we have used the Movie Ontology [8] as the domain ontology. The goal of our experiments was to verify the possibility of obtaining synthetic database schemas exhibiting not only a sufficiently large overlap but also some kind of heterogeneity. This was achieved by measuring the similarity between synthetic schemas and their corresponding modified version. Such task was accomplished using SemMatcher [10], a schema matching tool that takes as argument two database schemas represented as ontologies and produces a set of semantic correspondences as well as a global similarity measure between them. The global similarity between two schemas is a value in the interval  $[0,1]$ , where 0 indicates no similarity and 1 indicates a perfect match. For the experiment, synthetic database schemas with different sizes were generated: 4, 6, and 8 elements. For each size 10 schemas were produced. Figure 12 shows that the degree of similarity between synthetic schemas and their modified version decreases as new schema modifiers are added to the process.

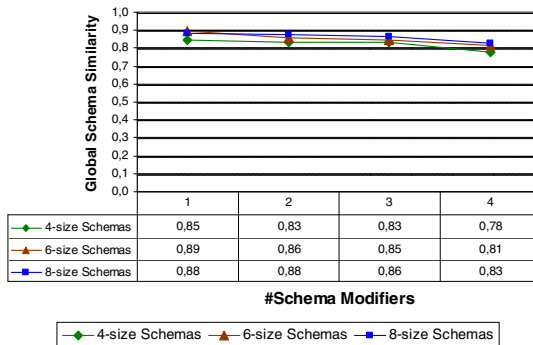


Fig. 12. Similarity between synthetic schemas

## 5 Related Work

To the best of our knowledge, there is no similar work on generating synthetic database schemas in the literature. We have found only tools that produce random data to populate empty schemas for performance testing purposes on databases [11, 12, 13]. For instance, Data Generator [11] is a free, open source script written in JavaScript, PHP and MySQL that allows the generation of large volumes of custom data in a variety of formats for use in testing software and populating databases. GS Data Generator [12] is an automated testing and data generation tool, which enables the creation of test data for software quality assurance testing, performance testing, usability testing, and database load testing. TurboData [13] is a relational database utility for system development. It can populate relations with default values and make sure that all foreign keys have a matching parent identifier. All these tools can be used to populate the synthetic schemas created by Schema Generator.

## 6 Conclusions and Further Work

In this work, we have proposed an automatic process to generate multiple synthetic database schemas. The resulting schemas can be used by applications that need to execute experiments involving a high number of data sources. The implementation has been used to produce random schemas for an existing PDMS application that simulates peer clustering [14]. During the experiments, a high number of synthetic schemas described in OWL were created. Each peer in the overlay network was associated with a unique synthetic schema. Peers were clustered according to their local schemas. There are a number of ongoing research issues concerned with the proposed schema generation process which will be the goal of our future activity. An issue to be studied in deep detail regards the specification of a process to produce synthetic queries to be executed at the synthetic schemas. Another work regards the generation of synthetic schemas considering that multiple base schemas related to the same domain are available. Finally, other schema modifiers can be developed and added to the Schema Generator tool. For instance, an algorithm that introduces noise in the synthetic schemas.

## References

1. Kantere, V., Tsoumakos, D., Sellis, T., Roussopoulos, N.: GrouPeer: Dynamic clustering of P2P databases. *Information Systems Journal* 34(1), 62–86 (2008)
2. Halevy, A.Y., Ives, Z.G., Mork, P., Tatarinov, I.: Piazza: Data Management Infrastructure for Semantic Web Applications. In: *Int. World Wide Web Conf.*, Budapest, Hungary, pp. 556–567 (2003)
3. Tatarinov, I., Halevy, A.: Efficient query reformulation in Peer Data Management Systems. In: *ACM SIGMOD Int. Conf. on Management of Data*, Paris, France, pp. 539–550 (2004)
4. Naicken, S., Livingston, B., Basu, A., Rodhetbhai, S., Wakeman, I., Chalmers, D.: The State of Peer-to-Peer Simulators and Simulations. *ACM SIGCOMM Computer Communication Review* 37(2), 95–98 (2007)
5. Ting, N.S., Deters, R.: 3LS - A Peer-to-Peer Network Simulator. In: *3<sup>rd</sup> Int. Conf. on Peer-to-Peer Computing*, p. 212 (2003)
6. Database Answers, <http://www.databaseanswers.org/>
7. The Internet Movie Database (IMDb), <http://www.imdb.com/>
8. The Movie Ontology (MO), <http://www.movieontology.org/>
9. Jena: a Semantic Web Framework for Java, <http://jena.sourceforge.net/>
10. Pires, C.E., Souza, D., Pachêco, T., Salgado, A.C.: A Semantic-based Ontology Matching Process for PDMS. In: *2<sup>nd</sup> Int. Conf. on Data Management in Grid and P2P Systems*, Linz, Austria, pp. 124–135 (2009)
11. Data Generator, <http://www.generatedata.com/#about>
12. GS Data Generator, <http://www.gsapps.com/products/datagenerator/>
13. TurboData, <http://www.turbodata.ca/>
14. Pires, C.E.: *Ontology-based Clustering in a Peer Data Management System*. PhD Thesis. Federal University of Pernambuco, Recife, Brazil (2009)

# A New Approach for Fuzzy Classification in Relational Databases

Ricardo Hideyuki Tajiri, Eduardo Zaroni Marques,  
Bruno Bogaz Zarpelão, and Leonardo de Souza Mendes

Department of Communication, School of Electrical and Computer Engineering,  
University of Campinas, Cidade Universitária Zeferino Vaz, Campinas, SP, Brazil  
r069671@dac.unicamp.br,  
{emarques,bzarpe,lmendes}@decom.fee.unicamp.br

**Abstract.** This paper presents an easy-to-use and easy-to-implement framework for fuzzy data classification and extraction in relational databases. The main benefits of the framework are: (i) a fuzzy data classification model for relational databases; (ii) flexible membership function configuration; (iii) automatic membership degree computation; (iv) a fuzzy data retrieval mechanism fully supported in SQL queries. In order to validate the proposed framework, a case study is implemented in a social welfare system using RDBMS Oracle 11g and PL/SQL programming language.

**Keywords:** Fuzzy classification, Relational database, SQL query.

## 1 Introduction

The relational model, proposed by Codd [1], is the most widespread and currently used database model [2]. In relational databases, data are queried and manipulated using SQL language. This language, developed by Chamberlin and Boyce [3], uses well defined precise conditions to retrieve data from database. Because of its dichotomous query process and the restrictions in relational databases to store only atomic data, the SQL language was not properly designed to deal with human subjectivity and imprecision, usual in natural language. Consequently, the necessity to relax query conditions in relational databases has been widely recognized as a mean to improve effectiveness in data retrieval, according to user needs [4].

In order to comply with this demand, different SQL extensions based on fuzzy sets theory were proposed [5][6]. However, some of the existing approaches are limited to a couple of membership functions, mainly trapezoid functions [2][7][8]. Other approaches do not provide tools for automatic computation of membership degrees [9][10][11]. Another characteristic of these approaches is the implementation of tools for query interpretation. These tools, if not properly implemented, may not support useful resources provided by the relational database system, like its native functions.

This paper proposes a framework for fuzzy classification in relational databases using SQL. It is an approach for flexible membership functions definition and automatic computation of membership degrees. Unlike other approaches, our work



implements a partial query interpreter, which is used directly in a SQL query as a fuzzy condition. Therefore, our approach is able to explore any resources from RDBMS allowed in SQL queries, such as native standard and non-standard database functions and user-defined functions while it retrieves information through imprecise linguistic terms. In order to validate our proposal, a case study is implemented in a social welfare system.

The paper is organized as followed: section 2 presents the proposed framework. Section 3 discusses the application of the framework in a case study. Finally, section 4 presents the conclusion and future work.

## 2 A Framework for Fuzzy Data Classification

As a solution for fuzzy classification in relational databases using SQL language, this paper proposes:

- A fuzzy metadata catalog that defines linguistic variables, linguistic terms and expressions for membership functions;
- A tool that computes the membership functions defined in the fuzzy metadata catalog, encapsulating all fuzzy classification processes inside the database. Therefore, enabling independency between the database and the application;
- An easy-to-use and easy-to-implement tool based on  $\mu_A(x)$  function [12]. This tool, defined as a fuzzy information interpreter, is implemented to write fuzzy conditions in SQL queries, requiring no additional lexical, syntactical nor semantic query analysis.

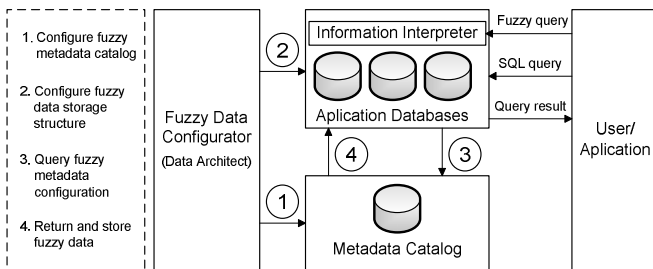


Fig. 1. Fuzzy architecture proposal

Fig. 1 shows the proposed framework architecture. Steps 1, 2, 3 and 4 illustrate the fuzzy metadata catalog configuration and fuzzy classification. These steps are presented in the following section.

### 2.1 Proposal for Fuzzy Data Classification

In order to store fuzzy sets context information, a fuzzy metadata catalog was modeled based on the one presented by Meier [11]. The main difference between both catalogs is the relation that defines the membership function for membership degree calculation. The

proposed catalog extends the *ContinuousFunction* relation of Meier’s catalog in four different relations: FUNCTIONS, FUNCTION\_CONSTANTS, CONSTANT\_VALUES and FUNCTION\_INTERVALS, as it is presented in Fig. 2. This structure improves Meier’s model, allowing the storage of expressions for the membership degree computation. With the stored expressions, it is possible to implement a stored function in the database system in order to calculate membership degrees. It decouples the database system from the application, with the fuzzy classification complexity encapsulated inside the database.

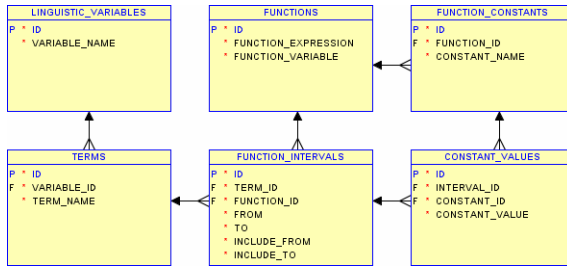


Fig. 2. Fuzzy metadata catalog

The first step of Fig. 1 represents the configuration of fuzzy metadata catalog, illustrated in Fig. 2. Each relation of this catalog stores:

- **LINGUISTIC\_VARIABLES**: the names of linguistic variables;
- **TERMS**: the names of linguistic terms, defining the domain of values of the linguistic variables;
- **FUNCTIONS**: the expressions for membership degrees calculations. These expressions must be written in a format recognized by the RDBMS (e.g.  $(a*x)$ , where  $x$  is the expression variable, also indicated in these relation, and  $a$  is a constant);
- **FUNCTION\_CONSTANTS**: the names of constants used in the expression stored in the FUNCTIONS relation (e.g.  $a$ );
- **FUNCTION\_INTERVALS**: a continuous subset of the universe of discourse  $U$  in which the expression defined by FUNCTIONS relation must be applied to a linguistic term defined by TERMS;
- **CONSTANT\_VALUES**: the numeric values of the constants defined by FUNCTION\_CONSTANTS relation (e.g.  $a=0.1$ ), so they can be used in a subset defined by FUNCTION\_INTERVALS.

This model allows the implementation of a generic deterministic stored function that reads data stored in the fuzzy metadata catalog. Then, it can automatically calculate the membership degrees of linguistic terms given a linguistic variable and a numeric value from an application table data. To perform this task, we propose a function defined as *generic\_function(vr\_name,x)*, with *vr\_name* as the linguistic variable name, defined by LINGUISTIC\_VARIABLES relation, and  $x$  as a numeric value to be classified as linguistic terms from TERMS relation. This value will replace the

expression variable defined by FUNCTIONS relation. Fig. 3 shows the *generic\_function(vr\_name,x)*'s fluxogram, illustrating the steps to calculate the membership degrees of linguistic terms.

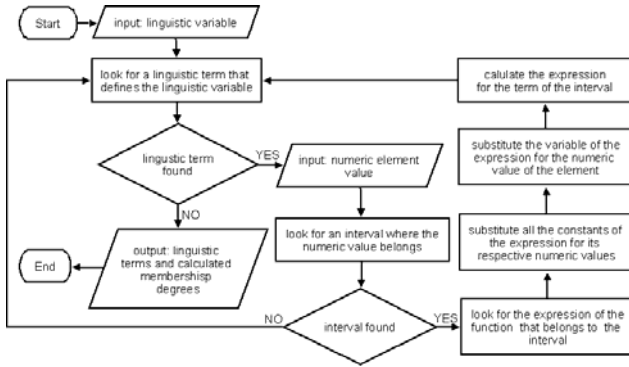


Fig. 3. *generic\_function(vr\_name,x)*'s fluxogram

Giving an element  $x$ , the *generic\_function(vr\_name,x)* must return the linguistic terms that defines the linguistic variable  $vr\_name$  and its respective membership degrees, when it is greater than zero ( $\mu_{term}(x) > 0$ ). Therefore, the *generic\_function(vr\_name,x)*'s return can be defined as the fuzzy information of the element  $x$  for the linguistic variable  $vr\_name$ . Thus, a fuzzy information format was defined to allow correct interpretation of such information. The proposed format simulates nested tables and is stated as follows:

$$term_1:degree_1|term_2:degree_2| term_3:degree_3|...| term_n:degree_n$$

The fuzzy information is stored as pairs  $\{term:degree\}$ , where “:” character simulates a column separator between the term and its respective degree. Also, the “|” character is used as a line breaker, separating two pairs  $\{term:degree\}$ . This approach allows storing the fuzzy information of a linguistic variable in only one column in the application data table, in the same tuple that contains the crispy value of the element calculated. It results in unary cardinality in the storage structure between the tuple and the fuzzy information returned by *generic\_function(vr\_name,x)*, making easier to write queries over these information.

With the fuzzy information stored in the database, it is necessary to create a mechanism that can interpret this information. This interpreter consists in a stored function implemented using the database system procedural language so it can be called directly in a SQL query. Such function is defined as  $FDEGREE(column_n,term\_name)$ , where:

1.  $t$  is a tuple of a relation;
2.  $column_n$  is an attribute of  $t$  which contains the fuzzy information;
3.  $term\_name$  is a linguistic term to be used as a search parameter value in the SQL query as a fuzzy condition.

The  $FDEGREE(column_n,term\_name)$  function reads the fuzzy information stored in  $column_n$ , and returns the membership degree of the linguistic term  $term\_name$  that defines

the linguistic variable and attribute used to calculate such fuzzy information. Thus, this function allows retrieving data from database through fuzzy classification using SQL queries recognized by the database system. The  $FDEGREE(column_b, term\_name)$  function, combined with the  $generic\_function(vr\_name, x)$  function, is a  $\mu_A(x)$  membership function [12] implementation. In other words,  $FDEGREE(column_b, term\_name) = \mu_{term\_name}(t)$ . Fig. 4 shows  $FDEGREE(column_b, term\_name)$ 's fluxogram.

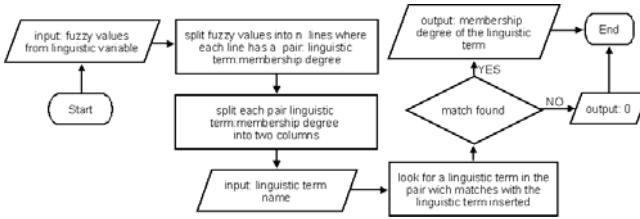


Fig. 4.  $FDEGREE(column_b, term\_name)$ 's fluxogram

As the  $FDEGREE(column_b, term\_name)$  is implemented using the database system procedural language, it is possible to use it in different clauses of a SQL query (e.g. *SELECT*, *WHERE*, *GROUP BY*, *HAVING* and *ORDER BY*) and as a aggregation function parameter (e.g. *MAX*, *MIN*, *COUNT*, etc). This approach also allows data retrieval through SQL queries by an  $\alpha$ -cut, strong  $\alpha$ -cut, support and kernel properties of fuzzy sets theory [2].

### 3 A Practical Application

During the last years, Brazil has made great investments to decrease citizen social vulnerability situation such as poverty and exclusion. One of the main factors that contributed for the improvement of social welfare is the conditional cash transfer programs offered to citizens by the government. In order to establish valid criteria for measuring family social vulnerability, a Family Development Index (IDF) [13] was created. This index assigns a real number value from the interval [0,1] to a family. The closer to zero, the greater is the evidence that this family belongs to a socially vulnerable group. Defining a sharp value to establish the families who should be considered socially vulnerable is not a simple task. Hence, classify IDF values using fuzzy sets theory becomes an attractive solution for dealing with these inaccuracies.

In this context, a case study was implemented using the fuzzy classification presented in section 2 over the IDF data table from the Social Welfare module from Integrated System for Municipal e-Gov (SIGM) [14]. This case study was implemented using RDBMS Oracle 11g and PL/SQL programming language. The adopted methodology for IDF table fuzzification was:

1. Implementation and configuration of fuzzy metadata catalog;
2.  $generic\_function(vr\_name, x)$  implementation in PL/SQL language;
3. Insertion of fuzzy information in the IDF data table;
4.  $FDEGREE(column_b, term\_name)$  implementation in PL/SQL language;

5. Data retrieval through fuzzy information using  $FDEGREE(column, term\_name)$  function in SQL queries.

Given the IDF values from the SIGM IDF data table, a linguistic variable named as “Social Vulnerability” was defined. Two linguistic terms, defining the universe of values for the linguistic variable, were chosen to classify the family’s situation through its IDF value: “High” and “Low” social vulnerability, as shown in Fig. 5.

LINGUISTIC_VARIABLES		TERMS		
ID	VARIABLE_NAME	ID	VARIABLE_ID	TERM_NAME
1	Social Vulnerability	1	1	High
		2	2	Low

Fig. 5. Definition of linguistic variable and linguistic terms

Fig. 6, 7 and 8 shows the configuration of fuzzy expressions for the membership degree calculus and IDF intervals used in each linguistic term and expression. The L function was chosen to represent the membership function for the term “High” and linear gamma function for “Low” social vulnerability [2].

FUNCTIONS				FUNCTION_CONSTANTS			
ID	FUNCTION_EXPRESSION	FUNCTION_VARIABLE		ID	FUNCTION_ID	CONSTANT_NAME	
1	a	x		1	1	a	
2	$(b-x)/(b-a)$	x		2	2	a	
3	$(x-a)/(b-a)$	x		3	3	b	
				4	4	a	
				5	5	b	

Fig. 6. Definition of expressions for membership degree calculus and its constants

ID	TERM_ID	FUNCTION_ID	FROM	TO	INCLUDE_FROM	INCLUDE_TO
1	1	1	0	0.35	Y	Y
2	2	1	0.35	0.75	N	N
3	3	2	0.35	0.75	N	N
4	4	2	0.75	1	Y	Y

Fig. 7. Definition of boundary values of IDF table and the membership expressions to be used for each linguistic term and interval

ID	INTERVAL_ID	CONSTANT_ID	CONSTANT_VALUE
1	1	1	1
2	2	2	0.35
3	3	3	0.75
4	4	4	0.35
5	5	5	0.75
6	6	4	1

Fig. 8. Definition of constant values for each expression and interval

In order to calculate the membership degree of the family IDF over the linguistic terms, the *generic\_function(vr\_name,x)* was implemented using the PL/SQL language. The fuzzy information returned by the *generic\_function(vr\_name,x)*, with “Social Vulnerability” and IDF values as the function parameters, was stored in a column named *SOCIAL\_VULNERABILITY*. Table 4 and Table 5 shows IDF table data samples before and after the addition of the column.

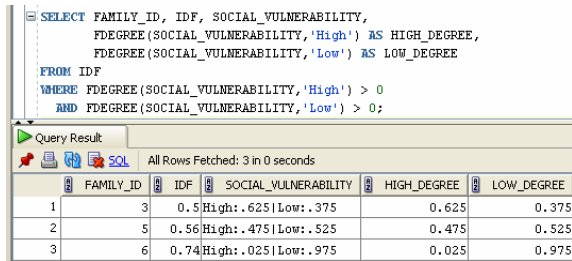
**Table 1.** IDF table sample before the fuzzy information be added

IDF	
FAMILY_ID	IDF
1	1
2	0
3	0.5
4	0.28
5	0.56
6	0.74

**Table 2.** IDF table sample with the fuzzy information column

IDF		
FAMILY_ID	IDF	SOCIAL_VULNERABILITY
1	1	Low:1.000
2	0	High:1.000
3	0.5	High:.625 Low:.375
4	0.28	High:1.000
5	0.56	High:.475 Low:.525
6	0.74	High:.025 Low:.975

With the fuzzy information stored in the data table, the *FDEGREE(column<sub>p</sub>,term\_name)* function was implemented using PL/SQL language, so that it can be used in a SQL query. The Fig. 9 illustrates a SQL query using the *FDEGREE(column<sub>p</sub>,term\_name)* function.



**Fig. 9.** Data retrieval through fuzzy classified data

## 4 Conclusion

This paper presented a framework for fuzzy classification in relational databases using SQL language. A membership function calculation mechanism and a fuzzy information format were proposed. In order to simplify the implementation of data retrieval through fuzzy classification, a fuzzy information interpreter was proposed. The use of this interpreter in a SQL query provided a simple tool for data retrieval using fuzzy classification. Finally, a case study was implemented classifying the IDF data table from SIGM system validating the proposed approach.

The encapsulation of membership degree calculus inside the database and the possibility to define expressions for the membership function (Fig. 6) are important

contributions for fuzzy classification in relational databases approaches, allowing independency between database system and system applications. In the case study, the proposed framework has shown that an existing system, such as the SIGM, can easily aggregate fuzzy functionalities, improving flexibility of data classification and retrieval, with minimum database changes.

Future work may include support to membership degrees modifiers and quantifiers and fuzzy classification of discrete and non-numeric values. Adaptations of the fuzzy metadata catalog and *generic\_function(vr\_name,x)* function must be considered.

## References

1. Codd, E.F.: A relational model of data for large shared data banks. *Commun. ACM* 13, 377–387 (1970)
2. Galindo, J.: *Fuzzy Databases: Modeling*. IGI Publishing (2006)
3. Chamberlin, D.D., Boyce, R.F.: SEQUEL: A structured English query language. In: *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control*, pp. 249–264. ACM, Ann Arbor (1974)
4. Bordogna, G., Psaila, G.: Customizable Flexible Querying in Classical Relational Databases. In: Galindo, J. (ed.) *Handbook of Research on Fuzzy Information Processing in Databases*, pp. 191–217. Information Science Reference, Hershey (2008)
5. Rosado, A., Ribeiro, R., Zadrozny, S., Kacprzyk, J.: Flexible Query Languages for Relational Databases: An Overview. In: Bordogna, G., Psaila, G. (eds.) *Flexible Databases Supporting Imprecision and Uncertainty*, vol. 203, pp. 3–53. Springer, Heidelberg (2006)
6. Zadrozny, S., Tré, G.d., Caluwe, R.d., Kacprzyk, J.: An Overview of Fuzzy Approaches to Flexible Database Querying. In: Galindo, J. (ed.) *Handbook of Research on Fuzzy Information Processing in Databases*, vol. I, pp. 34–54. Information Science Reference, Hershey (2008)
7. Galindo, J., Medina, J.M., Pons, O., Cubero, J.C.: A Server for Fuzzy SQL Queries. In: Andreasen, T., Christiansen, H., Larsen, H.L. (eds.) *FQAS 1998*. LNCS (LNAI), vol. 1495, pp. 164–174. Springer, Heidelberg (1998)
8. Zadrozny, S., Kacprzyk, J.: FQUERY for Access: towards human consistent querying user interface. In: *Proceedings of the 1996 ACM Symposium on Applied Computing*, pp. 532–536. ACM, Philadelphia (1996)
9. Veryha, Y.: Implementation of fuzzy classification in relational databases using conventional SQL querying. *Information and Software Technology* 47, 357–364 (2005)
10. Veryha, Y., Blot, J.-Y., Coelho, J.: Fuzzy Classification in Shipwreck Scatter Analysis. In: Galindo, J. (ed.) *Handbook of Research on Fuzzy Information Processing in Databases*, vol. II, pp. 516–537. Information Science Reference, Hershey (2008)
11. Meier, A., Werro, N., Albrecht, M., Sarakinos, M.: Using a fuzzy classification query language for customer relationship management. In: *Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 1089–1096. VLDB Endowment, Trondheim (2005)
12. Zadeh, L.A.: Fuzzy Sets. *Information and Control* 8, 338–353 (1965)
13. Barros, R.P.D., Carvalho, M.D., Franco, S.: O Índice de Desenvolvimento da Família (IDF). In: *Aplicada, I.D.P.E. (ed.) Instituto de Pesquisa Econômica Aplicada, Rio de Janeiro* (2003)
14. Tilli, M., Panhan, A.M., Lima, O., Mendes, L.S.: A Web-Based Architecture For E-Gov Application Development. In: *ICE-B - International Conference on e-Business, Porto - Portugal* (2008)

# Anomaly Detection for the Prediction of Ultimate Tensile Strength in Iron Casting Production

Igor Santos, Javier Nieves, Xabier Ugarte-Pedrero, and Pablo G. Bringas

S<sup>3</sup>Lab, DeustoTech - Computing, Deusto Institute of Technology  
University of Deusto,  
Avenida de las Universidades 24, 48007  
Bilbao, Spain  
{isantos,jnieves,xabier.ugarte,pablo.garcia.bringas}@deusto.es

**Abstract.** Mechanical properties are the attributes that measure the faculty of a metal to withstand several loads and tensions. In particular, ultimate tensile strength is the force a material can resist until it breaks. This property is one of the variables to control in the foundry process. The only way to examine this feature is to apply destructive inspections that make the casting invalid with the subsequent cost increment. Modelling the foundry process as an expert knowledge cloud allows machine-learning algorithms to forecast the value of a certain variable; in this case, the probability of a certain value of ultimate tensile strength for a foundry casting. Nevertheless, this approach needs to label every instance in the training dataset for generating the model that can foresee the value of ultimate tensile strength. In this paper, we present a new approach for detecting castings with an invalid ultimate tensile strength value based on anomaly detection methods. This approach represents correct castings as feature vectors of information extracted from the foundry process. A casting is then classified as correct or not correct by measuring its deviation to the representation of normality (correct castings). We show that this method is able to reduce the cost and the time of the tests currently used in foundries.

**Keywords:** Fault prediction, anomaly detection, ultimate tensile strength, industrial processes optimisation.

## 1 Introduction

Foundry can be considered as one of the axis of current economy because a huge number of castings are manufactured to be part of more complex systems e.g., the brake component of a car, the propeller of a boat, components of the wings of an aircraft and the trigger in a weapon. Therefore, if one of the pieces is faulty, it can be detrimental to both individuals and businesses activities.

Unfortunately, although there are many standards and methods to check the quality of the produced castings, these are performed once the manufacturing of



the casting has been completed. Most of the used techniques for the assurance of failure-free foundry processes are exhaustive production control and diverse simulation techniques [1] but they are extremely expensive. In this paper, we focus on the so-called Ultimate Tensile Strength (UTS). This mechanical property is defined as the force a casting can resist until it breaks; i.e., the maximum stress any material can withstand when subjected to tension. Manufactured iron castings must assure certain value (or threshold) of UTS to pass the strict quality tests. Unfortunately, the only available approach to examine the UTS breaks the piece, incurring in a cost increment within the process.

In our previous work [2–4], we have proven the ability of several machine-learning classifiers for the prediction of mechanical properties. We used Bayesian networks, support vector machines, decision trees, artificial neural networks and support vector machines, to identify the best overall machine-learning classifier capable of predicting the value of UTS and to reduce the noise in the manual data-gathering process [5]. However, machine-learning classifiers (or supervised learning methods) require a high number of labelled castings for each of the classes (i.e., faulty and not-faulty castings) to train the different models. However, it is quite difficult to acquire this amount of labelled data for a real-world problem such as production control. To generate these data, a time-consuming process of analysis is mandatory that renders in a cost increment during the process.

Given this background, we present here a method to classify castings to foresee the value of UTS that is based on anomaly detection methods. This approach is able to determine whether a casting contains a valid UTS value or not by comparing several features extracted from the production of the castings with a dataset composed only of the features of valid castings. Therefore, if the casting under prediction presents a considerable deviation to what is considered as normal (the previously stored correct castings), the casting is considered to be faulty and, thus, there may be a high probability that the casting has an invalid value of UTS. This method deals with the aforementioned problem, achieving a reduction in the required number of castings to be labelled.

Summarising, our main contributions are: (i) we select a set of variables extracted from the foundry process to determine whether a casting has a valid UTS value or not and provide a relevance measure for each variable based on information gain, (ii) we propose an anomaly-detection-based architecture for UTS prediction, by means of weighted comparison against a dataset composed of only correct castings and (iii) we evaluate the method using three different deviation measures and show that this method can reduce the need of labelling castings.

## 2 Foundry Processes and Mechanical Properties

Several factors, for instance the extreme conditions in which it is performed, make the foundry process very complex. Starting from the raw material to the manufactured item, this procedure involves numerous stages, several of which

may be performed in parallel. When it comes to iron ductile castings, this process presents the following phases:

- **Pattern making:** In this phase, the moulds (exteriors) and the kernels (interiors) are produced in wood, metal or resin to be used in the generation of the moulds where the final casting will be built.
- **Mould and kernel generation:** Although other methods exist, the sand mould is most widespread method for the manufacturing of foundry castings. The sand is mixed with clay, water or other chemical binders. Next, specialised machines generate the two halves of the mould and they unite them in order to create the container where the melt metal will be introduced.
- **Melting and pouring:** The raw metals are melt, mixed and poured onto the sand moulds.
- **Cooling:** The solidification of the castings is controlled in the cooling lines until this process is finished.
- **Finishing:** In order to finish the process, once the casting is cleaned, some actions are usually performed like thermal treatment, ratification of defects in welds and so on.

Once these phases finish, 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. It is important to know how mechanical properties affect to iron castings [6], because they directly affect the final quality of the manufactured casting. The most important mechanical properties of foundry materials are the following ones [7]: strength, hardness, resilience, elasticity, plasticity, brittleness, ductility and malleability. In this work, we focus on Ultimate Tensile Strength (UTS) that is a type of strength, which is the property that enables a metal to resist deformation under load. The testing method of UTS is conducted as follows. First, a scientist prepares a testing specimen from the original casting. Second, the specimen is placed on the tensile testing machine. Finally, this machine pulls the sample from both ends and measures the force required to break the specimen apart and how much the sample stretches before breaking.

The complexity of UTS prediction of the resulting castings arises mainly from the large number of variables involved in the production process and, therefore, this variables influence the final design of castings. The total number of variables we focus on has been reduced to 24, and more specifically, the control variables can be divided into metal-related variables and variables related to the mould.

- **Metal-related**
  - *Composition:* type of treatment, inoculation and quantities.
  - *Thermal:* Nucleation potential and quality of the mixture, obtained by thermal analysis [8].
  - *Pouring:* Pouring duration and temperature.

### – Mould-related

- Sand: types of additives used for sand, the specific characteristics of the sand.
- Mould: mould and machine parameters used.

Generally, the size and geometry of the casting play a very important and, therefore, we also included several variables to monitor these features. Similarly, the system takes into account the parameters related to the configuration of each machine working in the manufacturing process. Also, we added other variables such as cooling rate and heat treatment applied to the piece.

Although we have already obtained overall good results using a machine-learning-based approach for predicting imperfections and mechanical properties [2, 4, 5, 9–13], these approaches require a manual labour to label every instance of the training dataset. This process can be specially time-consuming and, also, means a cost increment.

We present here an anomaly-based approach that only requires labelling the correct castings and that measures the deviations of the inspected pieces with these previous stored castings. Such an approach will reduce the efforts of labelling castings, working with less information available. To this end, as we mentioned before, we manage 24 variables extracted from the foundry process. To provide a more accurate deviation measure, we apply relevance weights to each feature based on Information Gain (IG) [14]. This is done because IG provides a ratio for each characteristic that measures its importance to consider if a casting is valid or not. These weights were calculated from a real dataset acquired from a foundry specialised in safety and precision components for the automotive industry. The dataset is composed of 645 correct castings and 244 faulty castings.

## 3 Anomaly Detection

Through the features described in the previous section, our method represents valid castings as points in the feature space. When a casting is being inspected our method starts by computing the values of the point in the feature space. This point is then compared with the previously calculated points of the valid foundry castings.

To this end, distance measures are required. In this study, we have used the following distance measures:

- **Manhattan Distance:** This distance between two points  $v$  and  $u$  is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes:  $d(x, y) = \sum_{i=0}^n |x_i - y_i|$  where  $x$  is the first point;  $y$  is the second point; and  $x_i$  and  $y_i$  are the  $i^{th}$  component of first and second point, respectively.
- **Euclidean Distance:** This distance is the length of the line segment connecting two points. It is calculated as:  $d(x, y) = \sum_{i=0}^n \sqrt{v_i^2 - u_i^2}$  where  $x$  is the first point;  $y$  is the second point; and  $x_i$  and  $y_i$  are the  $i^{th}$  component of first and second point, respectively.

- **Cosine Similarity:** It is a measure of similarity between two vectors by finding the cosine of the angle between them [15]. Since we are measuring distance and not similarity we have used  $1 - \text{Cosine Similarity}$  as a distance measure:  $d(x, y) = 1 - \cos(\theta) = 1 - (\mathbf{v} \cdot \mathbf{u}) / (\|\mathbf{v}\| \cdot \|\mathbf{u}\|)$  where  $\mathbf{v}$  is the vector from the origin of the feature space to the first point  $x$ ,  $\mathbf{u}$  is the vector from the origin of the feature space to the second point  $y$ ,  $\mathbf{v} \cdot \mathbf{u}$  is the inner product of  $\mathbf{v}$  and  $\mathbf{u}$ .  $\|\mathbf{v}\| \cdot \|\mathbf{u}\|$  is the cross product of  $\mathbf{v}$  and  $\mathbf{u}$ . This distance ranges from 0 to 1, where 1 means that the two evidences are completely different and 0 means that the evidences are the same (i.e., the vectors are orthogonal between them).

Using these measures, we are capable of computing the deviation of a casting respect to a set of not faulty castings. Since we have to compute this measure with all the points representing valid castings, a combination metric is required in order to obtain a final value of distance which considers every measure performed. To this end, our system employs 3 very simplistic rules: (i) select the mean value, (ii) select the lowest distance value and (iii) select the highest value of the computed distances. In this way, when our method inspects a casting a final distance value is acquired, which will depend on both the distance measure and the combination rule.

## 4 Empirical Validation

In order to evaluate our anomaly-based faulty casting detector, we collected a dataset from a foundry, which is specialised in safety and precisions components for the automotive industry, principally in disk-brake support with a production over 45,000 tons a year.

The acceptance/rejection criterion of the studied models resembles the one applied by the final requirements of the customer. Pieces flawed with an invalid UTS 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: *Valid* (more than 370 MPa) and *Invalid* (less than 370 MPa).

We worked with two different references, in other words, type of pieces and, in order to test the proposed method, with the results of the destructive inspections of the 889 production stocks performed in beforehand. More accurately, the dataset comprises 645 correct castings and 244 faulty castings.

Specifically, we conducted the next configuration for the empirical validation:

1. **Cross validation.** Despite the small dataset, we have to use as much of the available information in order to obtain a proper representation of the data. To this extent, we performed a 5-fold cross-validation [16] over the correct castings to divide it into 5 different divisions of 552 castings for representing normality and 138 for testing. In this way, each fold is composed of 516 not faulty castings that will be used as representation of normality and 373 testing castings, from which 129 are valid castings and 244 are faulty castings.

2. **Calculating distances and combination rules.** We extracted the aforementioned characteristics and employed the 3 different measures and the 3 different combination rules described in Section 3 to obtain a final measure of deviation for each testing evidence. More accurately, we applied the following distances: (i) Manhattan Distance, (ii) Euclidean Distance and (iii) Cosine Similarity. For the combination rules we have tested the followings: (i) mean value, (ii) lowest distance and (iii) highest value.
3. **Defining thresholds.** For each measure and combination rule, we established 10 different thresholds to determine whether a casting is valid or not.
4. **Testing the method.** We evaluated the accuracy of the proposed model by measuring False Negative Rate (FNR) and False Positive Rate (FPR). In particular, FNR is defined as:  $FNR(\beta) = FN/(FN + TP)$  where  $TP$  is the number of faulty castings correctly classified (true positives) and  $FN$  is the number of faulty castings misclassified as valid castings (false negatives). On the other hand, FPR is defined as:  $FPR(\alpha) = FP/(FP + TN)$  where  $FP$  is the number of valid castings incorrectly detected as faulty castings while  $TN$  is the number of valid castings correctly classified.

**Table 1.** Results for different combination rules and distance measures. The results in bold are the best for each combination rule and distance measure. Our method is able to detect more than 78 % of the faulty castings although with a FPR higher than 40 %.

Combination	1 - Cosine Similarity			EuclideanDistance			ManhattanDistance		
	Threshold	FNR	FPR	Threshold	FNR	FPR	Threshold	FNR	FPR
Mean	0.2038	0.000	0.987	0.110	0.000	1.000	0.225	0.000	0.994
	0.2413	0.119	0.688	0.1184	0.087	0.825	0.2513	0.093	0.831
	0.2789	0.227	0.530	0.1265	0.129	0.664	0.2778	0.252	0.459
	<b>0.3164</b>	<b>0.566</b>	<b>0.189</b>	<b>0.1347</b>	<b>0.215</b>	<b>0.426</b>	<b>0.3043</b>	<b>0.520</b>	<b>0.190</b>
	0.3539	0.757	0.128	0.1428	0.551	0.192	0.3307	0.741	0.116
	0.3914	0.846	0.079	0.1510	0.724	0.114	0.3572	0.846	0.069
	0.4290	0.917	0.051	0.1591	0.831	0.062	0.3837	0.906	0.049
	0.4665	0.951	0.029	0.1673	0.931	0.029	0.4102	0.950	0.027
	0.5040	0.963	0.021	0.1754	0.972	0.006	0.4367	0.972	0.006
	0.5415	0.984	0.000	0.1836	0.999	0.000	0.4632	1.000	0.000
Maximum	0.5220	0.000	0.986	0.1810	0.000	0.978	0.4734	0.000	0.964
	0.5716	0.027	0.888	0.1887	0.081	0.899	<b>0.4984</b>	<b>0.041</b>	<b>0.891</b>
	<b>0.6212</b>	<b>0.063</b>	<b>0.818</b>	0.1964	0.170	0.810	0.5234	0.267	0.782
	0.6708	0.167	0.765	0.2042	0.238	0.686	0.5484	0.530	0.649
	0.7204	0.287	0.672	<b>0.2119</b>	<b>0.299</b>	<b>0.613</b>	0.5734	0.722	0.488
	0.7700	0.429	0.527	0.2196	0.521	0.471	0.5984	0.845	0.333
	0.8197	0.677	0.424	0.2274	0.733	0.355	0.6233	0.897	0.193
	0.8693	0.865	0.283	0.2351	0.841	0.201	0.6483	0.936	0.094
	0.9189	0.957	0.075	0.2428	0.964	0.105	0.6733	0.973	0.023
	0.9685	0.993	0.000	0.2505	0.989	0.000	0.6983	0.992	0.000
Minimum	0.0000	0.00000	0.94884	0.0000	0.000	0.978	0.0000	0.000	0.978
	<b>0.0212</b>	<b>0.804</b>	<b>0.060</b>	<b>0.0093</b>	<b>0.326</b>	<b>0.418</b>	<b>0.0169</b>	<b>0.320</b>	<b>0.415</b>
	0.0424	0.931	0.021	0.0186	0.559	0.248	0.0338	0.599	0.223
	0.0636	0.977	0.012	0.0279	0.727	0.145	0.0507	0.755	0.116
	0.0849	0.997	0.004	0.0371	0.809	0.046	0.0675	0.843	0.037
	0.1061	1.000	0.001	0.0464	0.879	0.021	0.0844	0.909	0.013
	0.1273	1.000	0.001	0.0557	0.985	0.004	0.1013	0.965	0.006
	0.1485	1.000	0.001	0.0650	0.999	0.004	0.1182	0.998	0.001
	0.1697	1.000	0.001	0.0743	0.999	0.003	0.99918	0.001	0.001
	0.1910	1.000	0.000	0.0836	1.000	0.000	0.1520	1.000	0.000

Table 1 shows the obtained results. Euclidean and Manhattan distances, despite of consuming less processing time, have achieved better results than cosine-similarity based distance for the tested threshold. Our anomaly-based faulty casting detector, for each distance measure, accomplished its best results selecting the mean value for computing the deviation of a casting respect to the not faulty castings. In particular, our detector is able to detect more than 78 % of faulty castings (using Euclidean distance), maintaining the rate of misclassified correct castings in 42.6%. Nevertheless, all distances obtain similar results.

## 5 Conclusions

Foreseeing the value of UTS in ductile iron castings is one of the most hard challenges in foundry-related research. Our work in [2, 4] pioneered the application of artificial intelligence methods to the prediction of the value of UTS.

This time, our main contribution is the anomaly-detection-based approach employed for UTS prediction. In contrast to our previous approaches, this method only need previously labelled the correct castings and it measures the deviation of castings respect to normality (castings with a valid value of UTS). Although anomaly detection systems tend to produce high error rates, in our case, the criteria establishes that a high false positive rate is tolerable whereas a high false negative rate is not. Therefore, our method is suitable for its direct application within real foundries.

Anyway, it presents some limitations that should be studied in further work. Firstly, we cannot identify different levels of warnings as we did in our previous works. In this case, we only can classify the castings as correct or faulty. Nevertheless, we could compute it using another anomaly detection techniques such as clustering based or nearest neighbour based anomaly detection.

Secondly, this kind of techniques based on the measurement of distances cannot achieve good results if the training data is disperse. In other words, if the normality cannot be represented as a compact group of instances, the threshold that allows to split the evidences between correct and faulty does not adjust to have its best behaviour. Nevertheless, this fact is solved due to the nature of the productions process, since all the castings are always produced in similar way. Hence, generated vectors of castings are close between each other, representing the normality in a good way in order to measure the distances between correct and faulty castings.

Finally, it is important to consider efficiency and processing time. Our system compares each casting against a relative big dataset (244 vectors for each fold). Despite Euclidean and Manhattan distances are easy to compute, cosine distance and more complex distance measures such as Mahalanobis distance may take too much time to process every casting under analysis. For this reason, in further work we will emphasise on improving the system efficiency by reducing the whole dataset to a limited amount of samples which is sufficiently representative.

## 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. 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 7<sup>th</sup> IEEE International Conference on Industrial Informatics (INDIN 2009), pp. 31–36 (2009)
3. Nieves, J., Santos, I., Peña, Y.K., Brezo, F., Bringas, P.G.: Enhanced foundry production control. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6261, pp. 213–220. Springer, Heidelberg (2010)
4. 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)
5. Nieves, J., Santos, I., Bringas, P.: Overcoming data gathering errors for the prediction of mechanical properties on high precision foundries. In: World Automation Congress (WAC), pp. 1–6. IEEE, Los Alamitos (2010)
6. Gonzaga-Cinco, R., Fernández-Carrasquilla, J.: Mechanical properties dependency on chemical composition of spheroidal graphite cast iron. *Revista de Metalurgia* 42, 91–102 (2006)
7. Lung, C.W., March, N.H.: *Mechanical Properties of Metals: Atomistic and Fractal Continuum Approaches*. World Scientific Pub. Co. Inc., Singapore (1992)
8. Larrañaga, P., Sertucha, J., Suárez, R.: Análisis del proceso de solidificación en fundiciones grafiticas esferoidales. *Revista de Metalurgia* 42(4), 244–255 (2006)
9. 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. 554–561. Springer, Heidelberg (2009)
10. Santos, I., Nieves, J., Bringas, P., Peña, Y.: Machine-learning-based defect prediction in highprecision foundry production. In: Becker, L.M. (ed.) *Structural Steel and Castings: Shapes and Standards, Properties and Applications*, pp. 259–276. Nova Publishers (2010)
11. Nieves, J., Santos, I., Peña, Y.K., Brezo, F., Bringas, P.G.: Enhanced foundry production control. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6261, pp. 213–220. Springer, Heidelberg (2010)
12. Santos, I., Nieves, J., Bringas, P.: Enhancing fault prediction on automatic foundry processes. In: World Automation Congress (WAC), pp. 1–6. IEEE, Los Alamitos (2010)
13. Santos, I., Nieves, J., Peña, Y.K., Bringas, P.G.: Towards noise and error reduction on foundry data gathering processes. In: Proceedings of the International Symposium on Industrial Electronics (ISIE), pp. 1765–1770 (2010)
14. Kent, J.: Information gain and a general measure of correlation. *Biometrika* 70(1), 163–173 (1983)
15. Tata, S., Patel, J.: Estimating the Selectivity of tf-idf based Cosine Similarity Predicates. *SIGMOD Record* 36(2), 75–80 (2007)
16. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the International Joint Conference on Artificial Intelligence, vol. 14, pp. 1137–1145 (1995)

# *LinkedPeers*: A Distributed System for Interlinking Multidimensional Data

Athanasia Asiki, Dimitrios Tsoumakos, and Nectarios Koziris

School of Electrical and Computer Engineering  
National Technical University of Athens, Greece  
{aassiki, dtsouma, nkoziris}@cslab.ece.ntua.gr

**Abstract.** In this paper we present *LinkedPeers*, a distributed system designed for efficient distribution and processing of multidimensional hierarchical data over a Peer-to-Peer overlay. The system design aims at incorporating two important features, namely large-scale support for partially-structured data and high-performance, distributed query processing including multiple aggregates. To achieve that, *LinkedPeers* utilizes a conceptual chain of DHT rings that stores data in a hierarchy-preserving manner and is able to adjust both the granularity of indexing and the amount of pre-computation according to the incoming workload. Extensive experiments prove that our system is very efficient achieving over 85% precision in answering queries while minimizing communication cost and adapting its indexing to the incoming queries.

## 1 Introduction

Our era can be characterized by an astonishing explosion in the amount of produced data, at a rate even bigger than Moore's law [1]. Market globalization, business process automation, new regulations, the increasing use of sensors, all mandate even more data retention from companies and organizations as a brute force method to reduce risk and increase profits. In most applications, data are described by multiple characteristics (or *dimensions*) such as time, customer, location, etc. Dimensions can be further annotated at different levels of granularity through the use of *concept hierarchies* (e.g., Year → Quarter → Month → Day). Concept hierarchies are important because they allow the structuring of information into categories, thus enabling its search and reuse.

Besides the well-documented need for efficient analytics, web-scale data poses extra challenges: While size is the dominating factor, the lack of a centralized or strict schema is another important aspect: Data without rigid structures as those found in traditional database systems are provided by an increasing number of sources, for example data produced among different sources in the Web [2]. The distribution of data sources renders many centralized solutions useless in performing on-line processing. Consequently, any modern analytics platform is required to be able to perform efficient analytics tasks on distributed, multi-attribute structured data without strict schema.

In this paper, we present the *LinkedPeers* system that efficiently stores and processes data described with multiple dimensions, while each dimension is organized by a concept hierarchy. We choose a Distributed Hash Table (DHT) substrate to organize *any* number of commodity nodes participating in *LinkedPeers*. Data producers can individually insert and update data to the system described by a predefined group of concept



hierarchies, while the number of dimensions may vary for each data item. Queries are processed in a fully distributed manner triggering adaptive, query-driven reindexing and materialization mechanisms to minimize communication costs.

The motivation behind the design of LinkedPeers is to provide a large-scale distributed infrastructure to accommodate collections of partially-structured data. In contrast to approaches where both data and their relationships are pre-defined by rigid schemas, we intend to support a higher degree of freedom: System objects are described by  $d$  dimensions, each of which is further annotated through a corresponding concept hierarchy. LinkedPeers does not require that each inserted fact be described by values for all dimensions. On the contrary, it attempts to fully support it and not restrict the ability to efficiently process it.

LinkedPeers manages to preserve all hierarchy-specific information for each dimension, using a tree-like data structure to store data and interlinking trees among different dimensions. A natural ordering of the dimensions that stems from their importance, query skew, etc, yields to a corresponding organization of the DHT layer: LinkedPeers comprises of multiple ‘virtual’ overlays, one for each dimension. This strategy results with each object being split into  $d$  parts and ending up in nodes of the *primary* and *secondary* rings. Trees at secondary rings maintain information towards related trees of the primary ring.

The purpose of this design is to couple the operational autonomy of the primary ring with a powerful meta-indexing structure integrated at the secondary rings, allowing our system to return fast aggregated results for the queried values by minimizing the communication cost. By allowing adaptive result caching and precomputation of related queries, this efficacy is further enhanced.

The proposed scheme enables the processing of complex aggregate queries for any level of any dimension, such as: “*Which Cities belong to Country ‘Greece’ ?*” or “*What is the population of Country ‘Greece’ ?*” or “*Which Cities of Country ‘Greece’ have population above 1 million in Year ‘2000’?*”, considering that the Location and Time hierarchies describe a numerical fact for population. The enforced indexing allows to find the location of any value of any stored hierarchy without requiring any knowledge, while aggregation functions can be calculated on the nodes that a query ends up.

To summarize, this work presents the LinkedPeers system which offers the following innovative features:

- A complete storage, indexing and query processing system for data described by an arbitrary number of dimensions and annotated according to defined concept hierarchies. LinkedPeers is able to perform efficient and online incremental updates and maintain data in a fault-tolerant and fully distributed manner.
- A query-based materialization engine that pro-actively precomputes relevant views of a processed query for future reference.
- Query-based adaptation of the indexing granularity of its indexing according to incoming requests.

Finally, to support our analysis, we present a thorough performance evaluation in order to identify the behavior of our scheme under a large range of data and query loads.

## 2 LinkedPeers System Description

### 2.1 Notation and Definitions

Data items are described by tuples containing values from a data space domain  $D$ . These tuples are defined by a set of  $d$  dimensions  $\{d_0, \dots, d_{d-1}\}$  and the actual fact(s). Each dimension  $d_i$  is associated with a concept hierarchy organized along  $L_i$  levels of aggregation  $\ell_{ij}$ , where  $j$  ( $j \in [0, L_{i-1}]$ ) represents the  $j$ -th level of the  $i$ -th dimension. We define that  $\ell_{ik}$  lies *higher* (*lower*) than  $\ell_{il}$  and denote it as  $\ell_{ik} < \ell_{il}$  ( $\ell_{ik} > \ell_{il}$ ) iff  $k < l$  ( $k > l$ ), i.e., if  $\ell_{ik}$  corresponds to a less (more) detailed level than  $\ell_{il}$  (e.g., *Month* < *Day*). Tuples are shown in the form:

$$\langle v_{0,0}, \dots, v_{0,L_0-1}, \dots, v_{d-1,0}, \dots, v_{d-1,L_{d-1}-1}, f_0, \dots \rangle$$

where  $v_{i,j}$  represents the value of the  $j$ -th level of the  $i$ -th dimension. Note also that any *value-set*  $(v_{i,0}, \dots, v_{i,L_i-1})$  for the  $i$ -th dimension may be absent from a tuple and that *fact<sub>j</sub>* may be of any type (e.g., numerical, text, vector, etc). Level  $\ell_{i0}$  is called the *root level* for the  $i$ -th dimension and its hashed value  $v_{i0}$  is called *root key*. The values of the  $\ell_{iL_i-1}$  are also referred to as *leaf values*.

The values of the hierarchy levels in each dimension are organized in tree structures, one per root key. Without loss of generality, we assume that each value of  $\ell_{ij}$  has at most one parent in  $\ell_{i(j-1)}$ . To insert tuples in the multiple overlays, one level from each dimension hierarchy is chosen; its hashed value serves as its *key* in the underlying DHT overlay. We refer to this level as *pivot level* and to its hashed value as *pivot key*. The pivot key that corresponds to the primary dimension (or *primary ring*) is called the *primary key*. The highest and lowest pivot levels of each hierarchy for a specific root key are called *MinPivotLevel* and *MaxPivotLevel* respectively.

The value-set of a dimension along the aggregated fact are organized as nodes of a *tree structure*, which contributes to the preservation of semantic relations and search. Figure 1 describes our running example. The shown tuples adhere to a 3-dimensional schema. The primary dimension is described by a 4-level hierarchy, while the other two are described by a 3-level and a 2-level hierarchy respectively. Note that the last two tuples do not contain values in  $d_1$  and  $d_2$  respectively. The selected pivot level for the primary dimension is  $\ell_{02}$  and thus all the shown tuples have the same pivot key in the primary dimension. All the value-sets in each dimension are organized in tree-structures with common root keys.

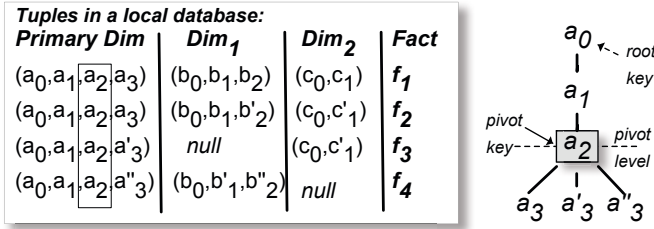
The basic type of query supported in LinkedPeers is of the form:

$$q = (q_{0k}, \dots, q_{ij}, \dots, q_{(d-1)m})$$

over the fact(s) using an appropriate aggregate function. By  $q_{ij}$  we denote the value for the  $j$ -th hierarchy level of the  $i$ -th dimension which can also be the special '\*' (or *ALL*) value.

### 2.2 Data Insertion

Our system handles both bulk insertions and incremental updates in a unified manner. As our design implies one virtual overlay per dimension, one key (using the SHA1 hash function for instance) for a selected pivot value of each dimension is generated.

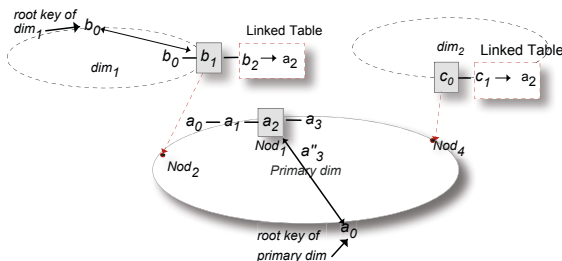


**Fig. 1.** A group of tuples with various value combinations among dimensions and the resulted tree structure for the primary dimension

During data insertions, the information about the pivot value is vital (only for initial insertions the pivot level can be selected according to the needs of the application). The design of LinkedPeers assumes if a value  $v_{ij}$  is selected as a pivot key during the insertion of a tuple, every other tuple that contains  $v_{ij}$  must also select it as its pivot key for dimension  $i$ . To comply with this assumption, a node should be aware of the existing pivot keys during the insertion of a new tuple. Thus, a fully decentralized catalogue storing information about root keys and their respective pivot keys in the network is implemented in LinkedPeers. Each root key is stored at the node with ID closest to its value. Every time that a new pivot key corresponding to this root key is inserted in the system, the root key node is informed about it and adds it in a list of known pivot keys. The root key node is also aware of the MaxPivotLevel used during the insertion of its values in the specific dimension.

The procedure for inserting the values of a tuple appropriately in all dimensions constitutes of the following basic steps:

- Inform each root key of every dimension about the corresponding value-set  $(v_{i,0}, \dots, v_{i,L_i-1})$  of the tuple, so as to decide for the appropriate pivot level.
- Insert each value-set  $(v_{i,0}, \dots, v_{i,L_i-1})$  to the corresponding  $i^{th}$ -ring.
- Create or update links among the trees of secondary dimensions towards the primary dimension.



**Fig. 2.** The created data structures after the insertion of the first tuple of Figure 1

Initially, the initiator contacts the root key of the primary dimension’s value-set. The root key of the primary dimension is informed about the new tuple and indicates the appropriate pivot level (if the same pivot key already exists, then its pivot level is used, otherwise the `MaxPivotLevel`). Afterwards, the DHT operation for the insertion of the tuple in the primary dimension starts and the tuple ends up to the node responsible for the decided pivot key. The node responsible for the pivot key of the primary dimension stores its value set in a tree structure and the whole tuple in a store defined as its *local database*. Moreover, it stores the result(s) of the aggregate function(s) over all these tuples (i.e., the results for a (pivot key, \*, ..., \*) query). Figure 2 demonstrates the insertion of the value-set  $(a_0, a_1, a_2, a_3)$  in the primary ring of an overlay consisting of nodes referred to as  $Nod_i$ . The root key  $a_0$  does not exist in the overlay and  $\ell_{02}$  is selected randomly as pivot level. The root index is created from  $a_0$  towards  $a_2$  and the tuple is inserted to the node  $Nod_1$ , which is responsible for the pivot key  $a_2$  according to the DHT protocol.  $Nod_1$  inserts all the values of the tuple in its local database as well.

The next step is to store the value-sets for the remaining dimensions in the corresponding ring. The node responsible for the primary key contacts each node responsible for the root keys and is informed about the appropriate pivot level in  $d_i$ . Since the pivot levels for the secondary dimensions are determined, the value-set of each dimension is stored in the node responsible for its pivot key. Again, the respective aggregates are also maintained in the nodes of the trees. The values of the secondary dimensions are associated to the primary dimension through the primary key. Each leaf value of a secondary tree structure maintains a list of the primary keys that is linked to. The structure storing the mappings among the leaf values and the primary keys is referred to as *Linked Table*. In case of an update taking place, the existing indices for any value of the tuple are also updated.

In Figure 2 the tree structures comprising of only one branch for the secondary dimensions are shown as well. During the insertion of value-set  $(b_0, b_1, b_2)$ , the root index  $b_0$  is created (the pivot level for  $c_0$  is the root level and no further indexing is needed). Figure 3 shows the final placement of the values of the tuples of Figure 1 among the nodes of the overlay.

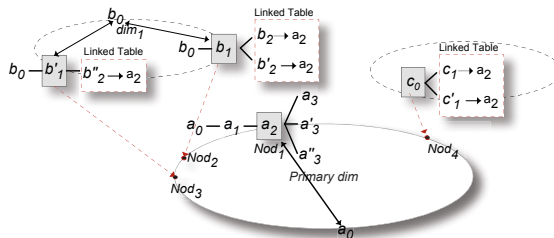


Fig. 3. Final placement and indexing of the tuples of Figure 1 in *LinkedPeers*

### 3 Query Processing

The queries posed to the system are expressed by conjunctions of multiple values. When a query includes a pivot value, then the node responsible for this value can be found with a simple DHT lookup. Otherwise, the native DHT mechanisms are not adequate to query the rest of the stored values. The proposed techniques can be further utilized to enable the search for any stored value.

The idea behind the approach followed for the insertion of tuples in the DHT overlay is the maintenance of the linking among the multiple dimensions, which can be searched either independently from each other or in conjunction with others. When the query does not define a specific value for a dimension (a ‘\*’-value), then any possible value is acceptable for the query. A query is assumed to include up to  $d-1$  ‘\*’ for  $d$  dimensions.

LinkedPeers allows adaptive change of pivot levels according to the query skew. Therefore, query initiators are not aware if any of the queried values correspond to a pivot value, forcing them to issue consecutive lookups for any value contained in the query according to the dimension priority, until they receive a result. Initially, a *lookup* operation is initiated for the value of the dimension with the highest priority. If the node holding the queried value cannot be located by the DHT lookup, then a lookup for the next non-‘\*’ value follows. If no results are returned for all the values in the query, then the query is flooded among the nodes of the overlay.

#### 3.1 Exact Match Queries

Queries concerning a pivot value of any ring are called *exact match* queries and can be answered by the DHT lookup mechanism. There are two categories of an exact match query:

*Category 1:* Query is  $q = (q_{0pivotlevel}, \dots)$ , where a pivot value of the primary dimension is defined in the query. Other values may be included as well. The DHT lookup ends up at the node responsible for the pivot key of the primary dimension. If this is the only value asked, the corresponding tree structure is searched for the aggregate fact. Otherwise, the local database is scanned and the results are filtered according to the remaining values locally.

*Category 2:* Query is  $q = (q_{0j}, \dots, q_{ipivotlevel}, \dots)$ , where  $j \neq pivotlevel$ . In this case, a queried value in one of the secondary dimensions is a pivot value. The strategy followed to resolve this query is that consecutive queries are issued until the node responsible for  $q_{ipivotlevel}$  is reached. If the query contains no other values, then the tree structure of this node is adequate to answer it, otherwise the query is forwarded to all the nodes of the primary dimension that store tuples containing  $q_{ipivotlevel}$ . These nodes query their local databases to retrieve the relative tuples and send back the results to the initiator. If more than one pivot values are present, then the query is resolved by the dimension with the highest priority. In the example of Figure 3, a query for value  $b_1$  can be resolved by the aggregated fact stored in *Nod*<sub>3</sub>. On the other hand, a query for the combination of values  $(a_3, b_1, *)$  reaches *Nod*<sub>2</sub>, which does not store adequate information to answer it and (using its Linked Table) forwards it to *Nod*<sub>1</sub>, which queries its local database.

### 3.2 Flood Queries

Queries not containing any pivot value cannot be resolved by the native DHT lookup. The only alternative is to circulate the query among all nodes and process it individually. The hierarchical structure of data together with the imposed indexing scheme enable a controlled flooding strategy that significantly reduces the communication cost.

Initially, a flood query is forwarded from a node to its closest neighbour in the DHT substrate. Each visited node searches its tree structures for any of the values included in the query. The visited nodes without any relative data are avoided for future forwarding of the query during the rest of the procedure for the flood resolution.

Query forwarding continues until any of the queried values is found in the stored trees. This node becomes the *coordinator* of the flood procedure. If more than one of the queried values are found in the same node, then the query is resolved in the ‘virtual’ ring of the dimension with the highest priority.

The found value may belong to a level either below the pivot level or above the pivot level. In the first case, there are no other trees with the specific value. The node either sends the aggregated fact to the initiator of the query or forwards it to the nodes of the primary dimension following the same strategy described for the second category of the exact match queries. Otherwise, there may exist other trees with the same value. For example, if a flood message for value  $a_1$  in Figure 3 reaches  $Nod_1$ , other nodes with the value  $a_1$  and different pivot keys may also exist. Yet, it is certain that this value is not stored at a node corresponding to a different root key. Thus, the flood message is forwarded to the node with the corresponding root key, which becomes the coordinator of the procedure from now on. This node forwards the queries to the nodes whose pivot keys it knows of, with each of them either returning the aggregated fact (when the value belongs to the primary dimension or only a single dimension is queried) or a set of candidate nodes that are linked in the primary dimension.

### 3.3 Materialized Views

In many high-dimensional storage systems, it is a common practice to pre-compute different views (GROUP-BYs) to improve the response time. For a given data set  $R$  described by  $d$  (dimensions) annotated by single-level hierarchies, a view is constructed by an aggregation of  $R$  along a subset of the given attributes resulting in  $2^d$  different possible views (i.e., exponential time and space complexity). The number of levels in each dimension adds to the exponent of the previous formula. In LinkedPeers, we consider a query-based approach to tackle the view selection problem: The selection of which views to pre-compute is query-driven, as we take advantage of the evaluation process to calculate parts of various views that we anticipate to need in the future.

Figure 4 depicts all the possible combinations of the query values  $(a_1, b_2, c_1)$ , relative to Figure 3. The attributes participating correspond to levels  $\{\ell_{01}, \ell_{12}, \ell_{31}\}$  respectively. Each combination (or *view identifier*) consists of a subset of attribute values in  $\{d_0, d_1, d_2\}$  ordered according to the priorities of dimensions in decreasing order. Moreover, each view identifier in the  $i$ -th level of the tree structure in Figure 4 is deduced by its successor view identifier in  $(i-1)$ -th level by omitting the participation of one dimension each time. When a value of a dimension is omitted in a view identifier, then it is

considered that its value is a ‘\*’-value. The identifiers that have already registered on the left-side of this tree are omitted.

Let  $S_i \subset S$  be the subset of view identifiers that start with the attribute value defined in dimension  $d_i$ . We call the subset of the specific view identifiers as  $Partition_{d_i}$  and the dimension that participates in all identifiers of the dimension as  $Root_{d_i}$ . In Figure 4,  $Partition_0$  comprises of all view identifiers that contain  $a_1$ , which is the  $Root_0$ , while  $a_1$  does not appear in any identifier of the remaining partitions.

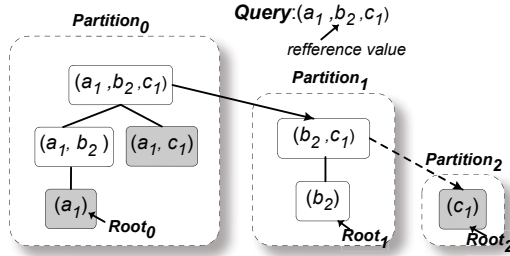


Fig. 4. All possible view identifiers for a query combining values in 3 dimensions

According to the strategy followed during flooding, all the nodes with trees containing the found value used for the resolution of the query (hence *reference value*) are definitely contacted. Thus, we conclude with certainty that there exist no extra nodes with tuples containing the reference value. This assumption is not valid for the rest of the values included in the query. This observation is significant for determining which views can be materialized and stored for future queries in a distributed manner:

Let  $S$  be the set of all the  $2^d$  identifiers. We deduce that only a subset  $S_{partial} \subset S$  of the view identifiers can be fully materialized, namely only the identifiers of the combinations including the reference value. In the example of Figure 4, let us assume that the flooded query for the combination  $(a_1, b_2, c_1)$  reaches  $Nod_2$  and the reference value is  $b_2$ . The query will be forwarded to  $Nod_1$  and it will be resolved. Nevertheless, it is not ensured that there are no other nodes storing tuples with  $a_1$  or  $c_1$ . Thus,  $S_{partial}$  comprises of the view identifiers in the *non-grey* boxes.

In more detail, the calculation of the views occurs among the nodes of LinkedPeers as follows: each peer that returns a found aggregated fact in a flooded query, also calculates the available view identifiers in  $S_{partial}$  stored in its local database. Due to the flooding strategy, every peer with trees containing the reference value will be definitely contacted. According to this procedure, the following conclusions are made:

- The  $S_{partial}$  may comprise only of identifiers belonging to  $Partition_{d_0}, Partition_{d_1}, \dots, Partition_{d_{ref}}$ , where the  $Root_{d_{ref}}$  of  $Partition_{d_{ref}}$  is the reference value used for the resolution of the flooded query.
- The upper bound of view identifiers that can be materialized is  $2^d - 1$  ('ALL' is not materialized), if the query does not contain any ‘\*’-value and without the restriction of the flood strategy. In case of ‘\*’-values, the number of view identifiers is  $2^{d-n} - 1$ ,

where  $n$  is the number of ‘\*’-values. According to the type of the inserted dataset (number of dimensions, number of tuples), the type of the query workload (average number of ‘\*’-values per query) and the specifications of the system, various policies can be defined to limit the number of calculated aggregated results.

Upon the reception of all the results, the coordinator merges the returned aggregated facts for each view identifier. Afterwards, it calculates the hash value of each  $Root_{d_j}$  and inserts each  $Partition_{d_j}$  ( $j \in [0, d_{ref}]$ ) to the overlay. The node responsible for the  $Root_{d_{ref}}$  also creates *indices* towards the locations of its tree structures to forward any query that cannot be resolved by the stored materialized views. The idea behind the splitting of the partitions is that the stored combinations need to be located with the minimum message cost, namely with the primitive DHT lookup. Since a query is disassembled in its elements and the queries are issued according to the priority of the dimensions, each identifier is stored to the dimension with the highest priority of its values.

Although any approach of existing relational schemas for storing views could be utilized to store the aggregated facts, we maintain simple ‘linked-listed’ structures. As shown in Figure 5, the view identifiers of Figure 4 are stored to the nodes responsible for the values appearing in the ‘dark grey’ boxes. All the queries arriving at the node responsible for  $Root_0$  (namely  $a_1$ ) should also include the  $Root_{d_{ref}}$ , which is  $b_2$ . The combination of value(s) that a query should include so as to be resolved by the existing view identifiers are marked with red boxes.

The created indices and views are soft-state and they expire after a predefined period of time, which is renewed each time that an existing index is used. The indices are bidirectional to ensure data consistency during re-indexing operations. Finally, we pose a limit to the maximum number of indices held by each node. Overall, the system tends to preserve the most “useful” indices towards the most frequently queried data items.

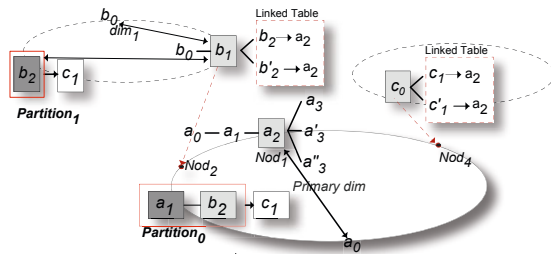


Fig. 5. Distribution of materialized view identifiers among the nodes of LinkedPeers

### 3.4 Indexed Queries

When a query reaches a node holding an index, then the stored views (if any) are searched for the combination of values included in the query. If the combination is found, the aggregated value is returned to the initiator. In the case that the combination does not exist, but the index is aware of the nodes with the pivot keys for the specific value, the query is forwarded to the respective pivot keys. If the query is simple or the



found value belongs to the primary dimension, then the aggregated facts for the query are returned. Otherwise, the reached nodes return the locations of the primary ring that are correlated with the indexed value. The query is forwarded to these nodes contacting their local database. After an indexed query which has not been resolved with the use of a stored view identifier, the procedure for materializing all the possible view identifiers described in the Section above is followed.

## 4 Adaptive Query-Driven Re-indexing

A significant feature of our system is that it dynamically adapts its indexing level on a per node basis to incoming queries. To achieve this, we introduce two re-indexing operations regarding the selection of pivot level: *Roll-up* towards more general levels of the hierarchy and *drill-down* to levels lower than the pivot level.

The idea behind individual re-indexing of stored tuples is based on the fact that each node has a global view of the queries regarding each level  $\ell_{ij} < pivotlevel$ , but only a partial view of the queries for each level  $\ell_{ij} > pivotlevel$  of a tree. Therefore, it has sufficient information to decide if a drill-down will be favorable for the values of this tree. On the other hand, a node has to cooperate with other peers that store a value of a level  $\ell_{ij} < pivotlevel$  in order to decide if this level is more appropriate. The decision for a possible re-indexing operation is made according to statistics collected by the incoming queries in the trees responsible for the specific value used during the resolution of a query. The goal is to increase the number of queries answered as exact matches in each dimension. The decision process for a possible re-index is triggered only after an indexed or flooded query only for the reference value, following the procedure described in our previous work [3]. Nevertheless, major enhancements have been made for the customization of the re-indexing operations in multiple dimensions due to the requirements arisen from the interconnection among the rings.

*Roll-up:* In general, if a node detects that the demand on a value above the pivot level relatively exceeds the demand for the other levels, it initiates the procedure to decide if a roll-up towards this level would be beneficial (communicating with the other interested nodes). A positive decision leads to the re-insertion of all trees containing the specific value with the new hash value in the overlay and the trees with the old pivot value are deleted. During a roll-up, one or more nodes re-insert their trees (or the whole tuples in the case of the primary dimension), which end up in one node responsible for the new pivot key. Each node also informs the root key about the new location and the new pivot key and erases all the soft-state indices towards any value of the re-indexed trees. The views containing any of these values in other rings are not affected, since the relocation of the trees does not affect the stored aggregated facts. The final step is the update of the links among the primary and the secondary rings: Each participating node signals the nodes that is linked to so as to replace the old pivot levels of the secondary ring with the new ones in their local databases (roll-up is performed in a secondary ring) or the links in all trees of the secondary rings related to the rolled-up trees, since the links need to be valid for the resolution of future queries.

*Drill-down:* The drill-down procedure is less complex, due to the fact that only one node holds the unique tree with values for this level. Thus, the node can locally decide

if the drill-down is needed. In this case, it splits the tree to tuples grouped by the new pivot key and re-inserts them in LinkedPeers. The root key is also informed about the new situation and all existing indices towards these trees are erased. Finally, the node that decided the drill-down updates the links among itself and the rest of the rings as described for the roll-up procedure.

## 5 Experimental Results

### 5.1 Simulation Setup

We now present a comprehensive evaluation of LinkedPeers. Our performance results are based on a heavily modified version of the FreePastry [4] using its simulator for the network overlay, although any DHT implementation could be used as a substrate. The network size is 256 nodes, all of which are randomly chosen to initiate queries.

Our synthetic data are trees (one per dimension) with each value having a single parent and a constant number of *mul* children. The tuples of the fact table to be stored are created from combinations of the leaf values of each dimension tree plus a randomly generated numerical fact. By default, our data comprise of 1M tuples, organized in a 4-dimensional, 3-level hierarchy. The number of distinct values of the top level is  $base = 100$  with  $mul=10$ . The level of insertion is, by default,  $\ell_1$  in all dimensions. For the query workloads, a 3-step approach is followed: We first identify which part of the initial database (i.e., tuple) the query will target (*TupleDist*). Next, the probability of a dimension  $d$  not being included (i.e., a ‘\*’ in the respective query) is  $P_{d*}$ . Finally, for included dimensions, we choose the level that the query will target according to the *levelDist* distribution. In our experiments, we express a different bias using the uniform, 80/20 and 90/10 distributions for *TupleDist* and *levelDist*, while  $P_{d*}$  increases gradually from 0.1 for the primary dimension to 0.8 for the last utilized dimension. Generated queries arrive at an average rate of  $1 \frac{query}{time\_unit}$ , in a 50k time units total simulation time.

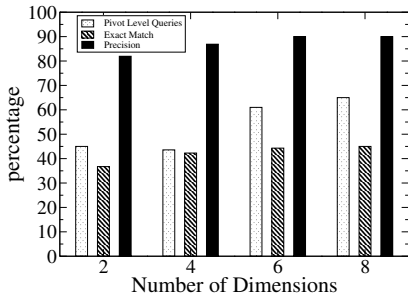
In this section, we intend to demonstrate the performance of our system for different types of inserted data and query workloads. The experimental results focus on the achieved *precision* (i.e., the percentage of queries which are answered without being flooded) and cost in terms of messages per query.

### 5.2 Performance under Different Number of Dimensions and Levels

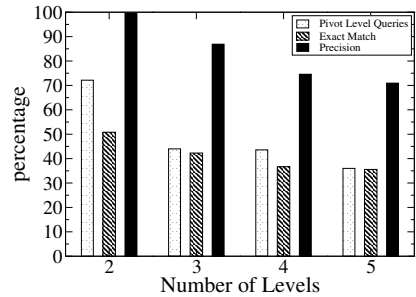
In the first set of experiments, we identify the behavior of our system under a variety of data workloads for different number of dimensions and different number of levels. The queries target uniformly any tuple of the dataset and the levels of the hierarchies in each dimension. In the first set of the experiments, we vary the number of dimensions, while each dimension is described by a 3-level concept hierarchy. Figure 6 demonstrates the percentage of the queries of the query workload that include at least one pivot value (*Pivot Level Queries*), the percentage of the queries resolved as exact match queries in LinkedPeers (*Exact Match*) and the achieved precision. The precision for non-flooded queries remains above 85% for all types of datasets, despite the number of dimensions. Queries that are not directed towards the pivot level are answered with

the use of an index or a materialized view assuring that the precision remains high. The difference among the exact matches and the pivot level queries is due to the utilized strategy that it is preferred for a query to be resolved as an indexed query in a dimension with higher priority than as an exact match to a dimension with a lower priority.

In Figure 7 the results for 4-dimensional workloads with varying number of level in the hierarchies are demonstrated. The decrease in the precision (from 99% to about 70%) is due to the fact that the increase of levels results to the decrease in the probability of querying a value that it is already indexed. Since the probability of utilizing an index decreases, all the queries targeting the initial pivot level ( $\ell_1$ ) even in the secondary rings are resolved as exact matches.



**Fig. 6.** Percentage of queries for different number of dimensions with 3-level hierarchies



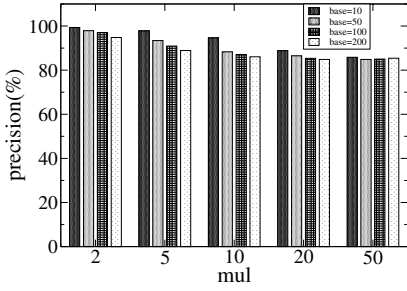
**Fig. 7.** Precision for different number of levels in 4-dimensional datasets

### 5.3 Query Resolution for Different Types of Datasets

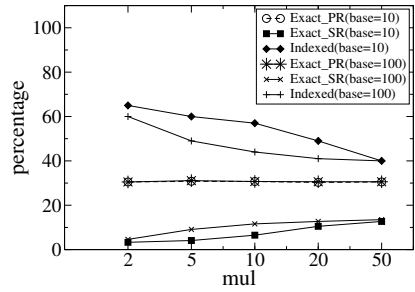
In this experiment, the achieved precision of LinkedPeers for various types of datasets is demonstrated in Figure 8. The number of distinct values in the top level base and the number of children mul are altered changing the density of the dataset. Base and mul influence the connections among primary and secondary rings and the number of distinct values in each level. As shown in Figure 8, there is a small decrease in the precision of non-flooded queries, while the base and the mul increases (this decreases the dataset density). Nevertheless, LinkedPeers achieves to resolve the majority of queries without flooding. The percentage of exact match queries in the primary dimension (Exact\_PR) remains stable for all datasets as shown in Figure 9, since it depends on the query workload. Nevertheless, the exact matches in the secondary rings (Exact\_SR) increase as the indexed queries decrease, since the indices of the primary dimension are used less, and more queries are resolved by the secondary rings.

### 5.4 Precision for Skewed Workloads

The adaptive behavior of LinkedPeers is identified in this set of experiments by testing the system under a variety of query loads. Specifically, we vary the *TupleDist* and the number of queries directed to each level by increasing the bias of *levelDist*. In Figure 10, data are skewed towards the higher levels of the hierarchy. The percentage of queries including at least one value in  $\ell_0$  or  $\ell_1$  are denoted as *Queries\_L0* and *Queries\_L1*



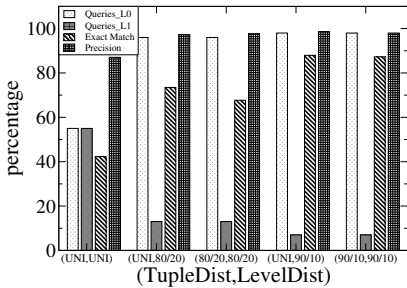
**Fig. 8.** Impact of mul and base in the achieved precision



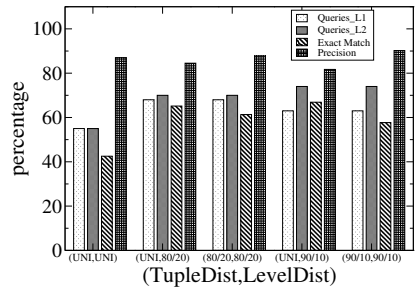
**Fig. 9.** Percentage of each query category for different data workloads

respectively. By making more biased the *levelDist*, we observe remarkably high precision rates (close to 100%). Despite the fact that the percentage of queries towards  $\ell_1$  (*Queries\_L1*) decreases significantly as the *levelDist* becomes more biased, the reindexing operations that take place ensure that the majority of queries are resolved as exact match queries by adjusting appropriately the pivot levels in each dimension.

Figure 11 depicts the results, when the query workload favors the lower levels of the hierarchies. The decrease to the precision for more biased *levelDist* is due to the fact that lower levels of the hierarchy have a considerably larger number of values. By increasing the number of queries towards these values, we increase the probability of queries targeting non-indexed values until the re-indexing mechanisms adapt the pivot levels of the popular trees appropriately.



**Fig. 10.** Precision and exact match queries for skew towards higher levels and various ( *TupleDist, levelDist*) combinations



**Fig. 11.** Precision and exact match queries for skew towards lower levels and various ( *TupleDist, levelDist*) combinations

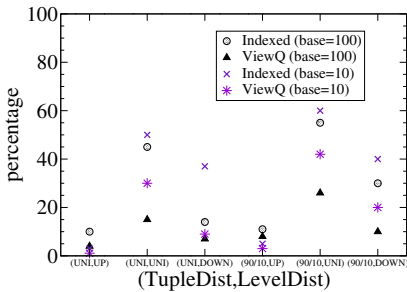
### 5.5 Testing against the Use of Materialized Views

Apart from the re-indexing operations, the materialized views can be also utilized to minimize the query cost. In the next experiment, we test our method against query workloads targeting the dataset either uniformly or biased (90/10) ( *TupleDist*) with uniform and biased (90/10) skew ( *levelDist*) towards the higher levels (denoted as UP)

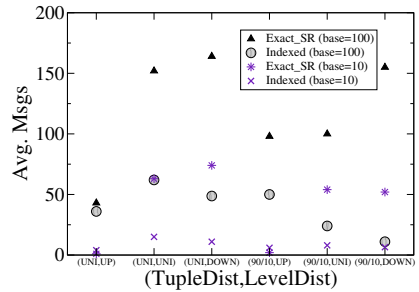
and towards the lower levels (DOWN). As shown in Figure 12, the queries resolved with the utilization of materialized views (ViewQ) increase in the query workloads targeting a part of the dataset at most, since the probability also increases for querying a materialized combination. More existing views are utilized, when the queries target uniformly all the levels of the hierarchies in all dimensions. In this case, the reindexing mechanisms cannot adjust the pivot levels to all the incoming queries and nearly the majority of indexed queries (Indexed) are resolved with the use of a materialized view.

## 5.6 Cost of the Various Types of Query Resolution

The cost of a query is considered as the messages that need to be issued for its resolution. A query resolved as exact match in the primary dimension utilizes only the DHT lookup mechanism. Figure 13 depicts the average number of messages only for exact queries resolved by secondary dimensions (Exact\_SR), which number is significantly smaller (less than 20% of all queries in all cases) and indexed queries (Indexed). The average number of messages for Exact\_SR depends on the type of dataset, namely the number of links among secondary pivot keys and primary pivot keys. When the query workload is skewed towards the higher levels (UP), then the messages decrease due to the fact that popular trees roll-up towards  $\ell_0$ . Thus, the secondary keys are connected to a smaller number of primary keys. The opposite observation is valid for the (DOWN) query workloads. It is important to notice, that the majority of the indexed queries (Indexed) in the workloads with higher cost for the indexed queries are resolved with views (see Figure 12), thus avoiding this cost.



**Fig. 12.** Utilization of materialized views compared to queries resolved as indexed



**Fig. 13.** Average number of messages for exact matches in secondary rings and indexed queries

## 5.7 Performance for Dataset of the APB Benchmark

The adaptiveness of the system is also tested using some realistic data. For this reason, we generated query sets by the APB-1 benchmark [5]. APB-1 creates a database structure with multiple dimensions and generates a set of business operations reflecting basic functionality of OLAP applications. The generated data are described by 4-dimensions. The customer dimension (C) is 100 times the number of members in the channel dimension and comprises of 2 levels. The channel dimension (Ch) has one level and 10

members. The product (P) dimension is a steep hierarchy with 6 levels and 10.000 members. Finally, the time dimension (T) is described by a 3-level dimension and made up of two years. The dataset is sparse (0.1 density) and comprises of 1.3M tuples.

Figure 14 shows the percentage of exact match queries resolved in primary and secondary rings compared to all exact match queries of a 25K query workload and for different rings combinations of ordering of dimensions. For all orderings, the precision of non-flooded queries is over 98%. The selection of the primary dimension influences the number of exact match queries in the primary ring. Figure 15 presents the average number of messages for exact matches resolved by a secondary ring and indexed queries, since only a DHT lookup is performed for exact match queries in the primary ring. The average number of messages is small for both exact and indexed queries, apart from the case that the customer dimension has been selected as a primary dimension. In the rest of the cases, the resolution of the queries occurs with a very low cost in terms of additional nodes to visit, even though the majority of the exact queries are resolved by a secondary dimension, as shown in Figure 14. The increase of messages for the CPChT dataset is due to the large number of distinct values used as pivot keys and thus each node responsible for a pivot key stores smaller portion of the total dataset in its local database. For all combinations of datasets, the overhead of the additional indexing structures needed by LinkedPeers such as tree structures, root indices, links and indices and statistical information is up to 1%. Thus, LinkedPeers can be considered as a lightweight solution for indexing multidimensional hierarchical data.

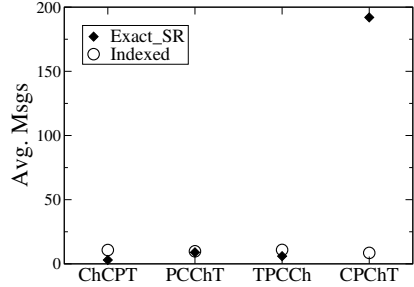
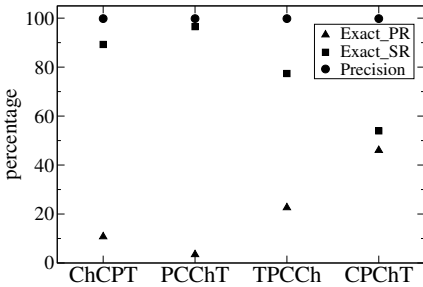


Fig. 14. Precision for APB query workload in LinkedPeers

Fig. 15. Average number of messages for exact match and indexed queries

## 6 Related Work

P2P systems based on Distributed Hash Tables (DHTs), for example [6], appear greatly effective for storing and locating *key – value* pairs. Nevertheless, complex queries cannot be supported without the implementation of additional indexing mechanisms. An approach to enable advanced search facilities in DHTs is the replacement of the hash function and the respective modification of the structure and behavior of the overlay to serve multi-attribute queries. Space Filling Curves [7], [8] are usually used as a replacement of the cryptographic hash function of the DHT protocols to produce a locality preserving mapping of multiple attribute values to a single key. In [8], an SFC

hash function is utilized over hierarchical attributes. Nevertheless, it is assumed that the full path from the root level towards the searched level is known and the values for all attributes in a query are given. Afterwards, the queries are transformed to range queries resolved by consecutive DHT lookups, usually resulting in flooding among all nodes. Our implementation does not pose any requirement for querying all the dimensions and allows the querying of any level of the hierarchy separately. There has been also significant work in the area of databases over P2P networks. PIER [9] proposes a distributed architecture for relational databases supporting operators such as join and aggregation of stored tuples. The Chatty Web [10] considers P2P systems that share (semi)-structured information but deals with the degradation, in terms of syntax and semantics, of a query propagated along a network path. In GrouPeer [11], SPJ queries are sent over an unstructured overlay in order to discover peers with similar schemas. Peers are gradually clustered according to their schema similarity. PeerDB [12] also features relational data sharing without schema knowledge. All these approaches offer significant and efficient solutions to the problem of sharing structured and heterogeneous data over P2P networks. Nevertheless, they do not deal with the special case of hierarchies over multidimensional datasets.

## 7 Conclusions

In this work, we described *LinkedPeers*, a distributed infrastructure for storing and processing multi-dimensional hierarchical data. Our scheme distributes large amount of partially-structured data over a DHT overlay in a way that hierarchy semantics and correlations among dimensions are preserved. Each data item can be described by an arbitrary number of dimensions and aggregate queries are resolved in a fully distributed manner. Re-indexing and pre-computation mechanisms are triggered dynamically during the resolution of queries. Our experimental evaluation over multiple and challenging workloads confirmed our premise: Our system manages to efficiently answer the large majority of queries using very few messages. It adds small overhead in storing hierarchical data and provides a lightweight indexing scheme, resolves efficiently aggregated queries and adapts to sudden shifts in skew by enabling re-indexing operations.

## References

1. MacManus, R.: The coming data explosion (2010), <http://www.readwriteweb.com/archives/>
2. Linked Data - Connect Distributed Data across the Web, <http://linkeddata.org/>
3. Asiki, A., Tsoumakos, D., Koziris, N.: Distributing and searching concept hierarchies: an adaptive dht-based system. *Cluster Computing* 13, 257–276 (2010)
4. FreePastry, <http://freepastry.rice.edu/FreePastry>
5. OLAP Council APB-1 OLAP Benchmark, <http://www.olapcouncil.org/research/resrchly.htm>
6. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In: *Proc. of the ACM SIGCOMM* (2001)

7. Schmidt, C., Parashar, M.: Squid: Enabling search in dht-based systems. *Journal of Parallel and Distributed Computing* 68(7), 962–975 (2008)
8. Lee, J., Lee, H., Kang, S., Kim, S.M., Song, J.: CISS: An efficient object clustering framework for DHT-based peer-to-peer applications. *Computer Networks* 51(4)
9. Huebsch, R., Hellerstein, J., Boon, N.L., Loo, T., Shenker, S., Stoica, I.: Querying the Internet with PIER. In: *VLDB* (2003)
10. Aberer, K., Cudre-Mauroux, P., Hauswirth, M.: The Chatty Web: Emergent Semantics Through Gossiping. In: *WWW Conference* (2003)
11. Kantere, V., Tsoumakos, D., Sellis, T., Roussopoulos, N.: GrouPeer: Dynamic clustering of P2P databases. *Inf. Syst.* 34(1), 62–86 (2009)
12. Ooi, B., Shu, Y., Tan, K., Zhou, A.: PeerDB: A P2P-based System for Distributed Data Sharing. In: *ICDE* (2003)



# A Vertical Partitioning Algorithm for Distributed Multimedia Databases

Lisbeth Rodriguez and Xiaou Li

Department of Computer Science, CINVESTAV-IPN, Mexico D.F., Mexico  
lisbethr@computacion.cs.cinvestav.mx, lixo@cs.cinvestav.mx

**Abstract.** Efficient retrieval of multimedia objects is a key factor for the success of distributed multimedia databases. One way to provide faster access to multimedia objects in these databases is using vertical partitioning. In this paper, we present a vertical partitioning algorithm for distributed multimedia databases (MAVP) that takes into account the size of the multimedia objects in order to generate an optimal vertical partitioning scheme. The objective function of MAVP minimizes the amount of access to irrelevant data and the transportation cost of the queries in distributed multimedia databases to achieve efficient retrieval of multimedia objects. A cost model for evaluating vertical partitioning schemes in distributed multimedia databases is developed. Experimental results clarify the validness of the proposed algorithm.

**Keywords:** Distributed multimedia databases, Vertical partitioning.

## 1 Introduction

In traditional (relational and object-oriented) database systems, the main performance concern is efficiency (how long it takes to answer a query) [1-3]. In multimedia database systems, efficiency is even more important due to the large size of multimedia objects [4]. Multimedia databases (MMDBs) are usually accessed remotely over a network. Multimedia objects identified as relevant to the query must be retrieved from the server and transmitted to the client for presentation [5].

Vertical partitioning (VP) techniques can be applied to MMDBs to provide faster access to relevant objects. This is because for these databases, attributes tend to be of very large size objects and without any particular storage technique queries are processed by sequential scanning of the complete database even when the queries do not require all the attributes of the database, the attributes stored in a database which are irrelevant (i.e., not required by a query) add considerably to the retrieval and processing cost of the queries, this will lead to too many disk accesses. Using VP techniques we can reduce the irrelevant attributes accessed by the queries in order to provide efficient retrieval of multimedia objects [6].

In distributed MMDBs, queries must retrieve data from multiple sites. A major cost in retrieving multimedia data from multiple sites is the cost incurred in transferring multimedia objects from different sites to the site where the query

is initiated [7]. In these databases, when the relevant attributes (i.e., required by the queries) are in different fragments and allocated to different sites, there is an additional cost due to remote access of data. Thus one of the desirable characteristics of these databases that need to be achieved through VP is to maximize local accessibility, i.e., each site must be able to process the queries locally with minimal access to data located at remote sites. Ideally, we would like any query to access only the attributes of a single fragment with no or minimal access of irrelevant attributes in that fragment. But this is impossible to achieve in the general case because queries access different and overlapping subsets of attributes and the relevant attributes may reside in different sites. So, we only have to look for a vertical partitioning scheme (VPS) which minimizes the amount of irrelevant attributes and remote attributes accessed [8].

Most VP techniques do not consider the size of the attributes in the VP process, this is suitable for traditional databases which only contain alphanumeric attributes because the size of different alphanumeric attributes is similar. Nevertheless, MMDBs contain both alphanumeric and multimedia attributes [9], therefore the size of the attributes is very varied. We must take into account the size of the attributes in order to reduce the access to irrelevant multimedia attributes and the transportation of multimedia attributes in distributed MMDBs.

We are concerned with creating a VP algorithm for distributed MMDBs called MAVP (Multimedia Adaptable Vertical Partitioning), which takes into account the size of the attributes in the VP process. The objective function of MAVP can efficiently increase local accessibility and reduce the amount of access to irrelevant multimedia attributes, so that an optimal VPS could be obtained.

## 2 Background

### 2.1 Vertical Partitioning

VP of a table  $T$  produces fragments  $fr_1, fr_2, \dots, fr_p$  each of which contains a subset of the attributes of  $T$ 's attributes as well as the primary key of  $T$ . The objective of VP is to divide a table into a set of smaller tables so that many of the queries will run on only one fragment. In this context, an "optimal" partitioning is one that produces a VPS which minimizes the execution time of queries that run on these fragments [10].

Several VP techniques have been developed in traditional databases. The main classification of VP is *affinity-based* and *cost-driven* approaches [11].

The former is based on affinity which measures if two attributes are accessed together by the queries, the VP process in these approaches is as follows. First, they use as input an *Attribute Usage Matrix (AUM)*, which has information about queries with regard to their attributes accessed and their frequency. Next, an *Attribute Affinity Matrix (AAM)* is constructed according to this information. If a query use two attributes together, then the value of affinity of these attributes is increased in AAM with the value of the frequency of such query. Then, the AAM is clustered in a way that attributes with most affinity between them are

grouped together. Finally, the AAM is partitioned to get the fragments. Research work includes [12–16].

The main disadvantage of *affinity-based* approaches is that this measure does not reflect the closeness or affinity when more than two attributes are involved. Hence algorithms which use AAM are using a measure that has no bearing on the affinity measured with respect to the entire cluster [8]. *Cost-driven* approaches offer a solution to this problem. In *cost-driven* algorithms, a cost model is used to get the vertical fragments. The VP process in these algorithms is as follows. First, they use an AUM as input like the *affinity-based* approaches. Then, they cluster the attributes and get the fragments according to their cost model, the main objective is to reduce the cost of the queries. Research work includes [8], [11], [17].

## 2.2 Multimedia Database Partitioning

Current VP techniques only consider alphanumeric data. Although some approaches to fragment MMDBs have been recently provided in the literature [18], [19], [9], they only focus on horizontal partitioning. To the best of our knowledge, VP of MMDBs only has been addressed in [20]. However, the fragments are obtained using a cost model based on the savings in number of disk accesses and it does not take into account the transportation cost which is a major cost in distributed MMDBs [7].

**Scenario.** To illustrate and motivate our approach throughout the paper, we consider the following scenario of a simple MMDB used to manage equipment in a machinery sell company. The database consists of table *EQUIPMENT*(*id*, *name*, *image*, *graphic*, *audio*, *video*), where each tuple describes information about a specific equipment, including its image, graphic, audio, and video objects. Let us also consider the following queries:

- q<sub>1</sub>:Find all images and graphics of chain saws
- q<sub>2</sub>:Find name, audio and video with id "MB01"
- q<sub>3</sub>:Find all graphic, audio and video
- q<sub>4</sub>:Find all image of water pumps

**Motivation.** Both *affinity-based* and *cost-driven* VP approaches only focus on the attributes used by the queries and their frequency as input to the process of VP. Traditional databases only contain alphanumeric attributes, the size of different alphanumeric attributes is very similar. For example, we take the AUM in [14] to illustrate how the difference in size between the attributes is not relevant. The 0/1 entries in the AUM show whether or not the attributes (A) are used by the queries (Q). The frequency column (F) shows for each query the frequency of access to attributes per unit time period (e.g., a day). The attribute size row (S) gives the number of bytes of each attribute. As we can see in Table 1, the smallest attribute is *a*<sub>7</sub> with 3 bytes and the largest attribute is *a*<sub>5</sub> with 15 bytes, so the difference between them is 12 bytes.

**Table 1.** Attribute Usage Matrix of a traditional database

Q/A	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$	$a_7$	$a_8$	$a_9$	$a_{10}$	F
$q_1$	1	0	0	0	1	0	1	0	0	0	25
$q_2$	0	1	1	0	0	0	0	1	1	0	20
$q_3$	0	0	0	1	0	1	0	0	0	1	25
$q_4$	0	1	0	0	0	0	1	1	0	0	35
$q_5$	1	1	1	0	1	0	1	1	1	0	25
$q_6$	1	0	0	0	1	0	0	0	0	0	25
$q_7$	0	0	1	0	0	0	0	0	1	0	25
$q_8$	0	0	1	1	0	1	0	0	1	1	15
S	10	8	4	6	15	14	3	5	9	12	

Nevertheless, in MMDBs the size of attributes is very important, because multimedia attributes usually require a larger amount of memory and disk storage (e.g., it is common that a video clip occupies more than 100 MB of disk storage) than alphanumeric attributes. Since MMDBs have both alphanumeric and multimedia attributes, the size of different attributes is very varied. For example, in the AUM of the table EQUIPMENT of Table 2 we can see that the smallest attribute is *id* with 8 bytes and the largest attribute is *video* with 39518 bytes, so the difference between them is 39510 bytes.

**Table 2.** Attribute Usage Matrix of a multimedia database (table EQUIPMENT)

Q/A	id	name	image	graphic	audio	video	F
$q_1$	0	1	1	1	0	0	15
$q_2$	1	1	0	0	1	1	10
$q_3$	0	0	0	1	1	1	25
$q_4$	0	1	1	0	0	0	20
S	8	20	900	500	4100	39518	

Traditional database applications use data of fixed size but the size of multimedia data in MMDBs like audio and video is not fixed and can vary dynamically. Nevertheless, the database administrator (DBA) is in charge to specify the maximum size of a multimedia attribute. For example, in Table 2 the size of *video* is 39518 bytes, this means that all the videos in the database may have a maximum size of 39518 bytes.

With current fragmentation approaches all the attributes are considered equal because they do not consider the size of attributes. Nevertheless, in distributed MMDBs it is not the same accessing an irrelevant attribute *id* than an irrelevant attribute *video* and it is also different accessing remotely an attribute *id* than an attribute *video*. We need to develop a VPS taking into account the size of attributes, because we can avoid the access to irrelevant very large multimedia objects and avoid the transference of very large multimedia objects in order to get a better VPS.

### 3 Multimedia Adaptable Vertical Partitioning Algorithm (MAVP)

MAVP requires as input AUM and the size of attributes. It is an extension of the AVP algorithm in [17], so it supports n-way VP as well as best-fit VP of MMDBs. The former is to generate the specific number of fragments required by the user, the latter is to generate an overall optimal partitioning that minimizes the processing cost of queries without restriction on the number of fragments generated.

#### 3.1 Partition Tree

AVP is based on a bottom-up approach. It first begins with single attribute fragments. And then, it forms a new fragment by selecting and merging two fragments of them. This process is repeated until a fragment composed of all data attributes is made. This kind of bottom-up approach generates a binary tree, which is called a partition tree (PT). Fig. 1 shows the PT of the table EQUIPMENT obtained by MAVP.

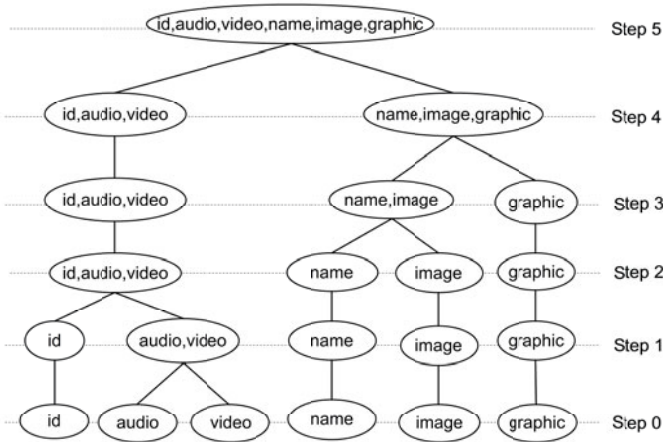


Fig. 1. Partition tree of the table EQUIPMENT obtained by MAVP

When two fragments are merged the amount of remote relevant attributes accessed is decreased while the amount of irrelevant attributes accessed by the queries is increased. For example, in Table 2 we have that both queries  $q_2$  and  $q_3$  access the multimedia attributes *audio* and *video*, at the beginning of the process they are in different fragments, but if they are merged into a fragment,  $q_2$  and  $q_3$  have only to access such fragment, resulting in decreased the access to remote attributes.

In addition, merging the attributes causes that the amount of irrelevant attributes accessed by the queries is increased. For example, if the attributes *name* and *image* are merged into a fragment, the query  $q_2$  only uses the alphanumeric attribute *name*, but it also has to access the multimedia attribute *image*. Therefore, the merged fragment will increase the amount of accesses to irrelevant attributes.

In AVP in each step during constructing a PT, two nodes (fragments) are selected which maximizes the merging profit defined below when they are merged into a node (fragment).

$$MergingProfit(AVP) = c \cdot d\_DF - n\_IA \tag{1}$$

where

$d\_DF$  : the decreased #DF (i.e. the total frequency of accessing different fragments)

$n\_IA$ : the newly created #IA (i.e the total frequency of interfered accesses between data queries)

$c$ : a proportional constant between #DF and #IA.

### 3.2 Merging Profit of MAVP

One disadvantage of the merging profit of AVP is that it uses a constant  $c$ , but they do not give a guideline to set this parameter because using different values we get different VPSs, it is necessary to find the adequate value of  $c$  in order to get the optimal VPS, so this complicates the use of merging profit.

In our merging profit function we eliminate such complexity because we do not need any constant; all the values are easily obtained with the AUM. Taking into account the size of the attributes we define the merging profit function as

$$MergingProfit(MAVP) = \#Q1 \cdot DRA - \#Q2 \cdot IIA \tag{2}$$

where

#Q1: the number of queries that reduce the access to remote attributes

DRA: the decreased amount of remote attributes accessed

#Q2: the number of queries that increase the access to irrelevant attributes

IIA: the increased amount of irrelevant attributes accessed

In each step during constructing a PT, MAVP produces a VPS merging two fragments which maximize the merging profit function defined in equation 2, so when the PT is finished we have a set of VPSs  $VPS = \{vps_1, vps_2, \dots, vps_n\}$ , every  $vps_i$  has a set of fragments  $FR = \{fr_1, fr_2, \dots, fr_p\}$ .

Two select two fragments of  $p$  fragments which can maximize the merging profit  $\binom{p}{2} (= \frac{p(p-1)}{2})$  pairs should be examined. For example, in Step 0 (VPS<sub>6</sub>)  $p=n$  (where  $n$  is the number of attributes) because each attribute is located in a different fragment, so there are six fragments in Step 0 of Fig. 1 and it is necessary examine the merging profits of  $\binom{6(6-1)}{2} (= 15)$  pairs and merge one

pair with the maximum merging profit among them, which generates the VPS<sub>5</sub> of Step 1 in Fig. 1

Table 3 shows MAVP Merging Profit Matrix (MPM) of the table EQUIPMENT in Step 0. In Algorithm 1, we show the process to get the MPM.

**Table 3.** Merging Profits of table EQUIPMENT in Step 0

	id	name	image	graphic	audio	video
id	-280	-27840	-15960	40880	395060	
name		55400	-38700	-390800	-3755510	
image			-44000	-700000	-5658520	
graphic				-18000	-195090	
audio					3053260	
video						

MAVP is presented in algorithm 2, it uses as input the AUM of the table with the attribute size row and generates the optimal vertical partitioning scheme.

Table 4 shows the VPSs of the table EQUIPMENT obtained using AVP and MAVP.

**Table 4.** Resulting fragments of the table EQUIPMENT

VPS Algorithms	
AVP	MAVP
vps <sub>1</sub> fr <sub>1</sub> (a <sub>1</sub> ,a <sub>2</sub> ,a <sub>3</sub> ,a <sub>4</sub> ,a <sub>5</sub> ,a <sub>6</sub> )	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>2</sub> ,a <sub>3</sub> ,a <sub>4</sub> ,a <sub>5</sub> ,a <sub>6</sub> )
vps <sub>2</sub> fr <sub>1</sub> (a <sub>1</sub> ,a <sub>2</sub> ,a <sub>3</sub> ) fr <sub>2</sub> (a <sub>4</sub> ,a <sub>5</sub> ,a <sub>6</sub> )	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>5</sub> ,a <sub>6</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>3</sub> ,a <sub>4</sub> )
vps <sub>3</sub> fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>3</sub> ) fr <sub>3</sub> (a <sub>4</sub> ,a <sub>5</sub> ,a <sub>6</sub> )	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>5</sub> ,a <sub>6</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>3</sub> ) fr <sub>3</sub> (a <sub>4</sub> )
vps <sub>4</sub> fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>3</sub> ) fr <sub>3</sub> (a <sub>4</sub> ) fr <sub>4</sub> (a <sub>5</sub> ,a <sub>6</sub> )	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>5</sub> ,a <sub>6</sub> ) fr <sub>2</sub> (a <sub>2</sub> ) fr <sub>3</sub> (a <sub>3</sub> ) fr <sub>4</sub> (a <sub>4</sub> )
vps <sub>5</sub> fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>3</sub> ) fr <sub>3</sub> (a <sub>4</sub> ) fr <sub>4</sub> (a <sub>5</sub> ) fr <sub>5</sub> (a <sub>6</sub> )	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ) fr <sub>3</sub> (a <sub>3</sub> ) fr <sub>4</sub> (a <sub>4</sub> ) fr <sub>5</sub> (a <sub>5</sub> ,a <sub>6</sub> )
vps <sub>6</sub> fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ) fr <sub>3</sub> (a <sub>3</sub> ) fr <sub>4</sub> (a <sub>4</sub> ) fr <sub>5</sub> (a <sub>5</sub> ) fr <sub>6</sub> (a <sub>6</sub> )	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ) fr <sub>3</sub> (a <sub>3</sub> ) fr <sub>4</sub> (a <sub>4</sub> ) fr <sub>5</sub> (a <sub>5</sub> ) fr <sub>6</sub> (a <sub>6</sub> )
a <sub>1</sub> = id, a <sub>2</sub> = name, a <sub>3</sub> = image, a <sub>4</sub> = graphic, a <sub>5</sub> = audio, a <sub>6</sub> = video	

## 4 Cost Model

MAVP computes the cost of each step to get the optimal VPS. We modify the cost model of AVP to adapt it to distributed MMDBs, because it does not take into account the size of attributes. MAVP will select the VPS which minimizes the access to irrelevant attributes and maximizes the local accessibility (minimizes the transportation cost) of the queries.

MAVP produces a VPS at each step in the construction of the PT, so when the PT is finished we have a set of VPSs  $VPS = \{vps_1, vps_2, \dots, vps_n\}$ , every  $vps_i$  has a set of fragments  $FR = \{fr_1, fr_2, \dots, fr_p\}$ , and each fragment  $fr_k$  has  $n_k$  attributes, we suppose that the network has nodes  $N_1, \dots, N_p$ , the allocation of the fragments to the nodes gives rise to a mapping  $\lambda: \{1, \dots, p\} \rightarrow \{1, \dots, p\}$  called

---

**Algorithm 1.** getMPM

---

input: AUM of the table (a set of alphanumeric and multimedia attributes  $A=\{a_1, a_2, \dots, a_n\}$ , a set of queries  $Q=\{q_1, q_2, \dots, q_m\}$ , the frequency  $f_k$  of each query  $q_k$   $F=\{f_1, f_2, \dots, f_m\}$ , the size  $s_i$  of each attribute  $a_i$   $S=\{s_1, s_2, \dots, s_n\}$ ).

output: MPM Merging Profit Matrix

begin

```

for each attribute  $a_i \in A$ ,  $1 \leq i \leq n-1$  do
  for each attribute  $a_j \in A$ ,  $i+1 \leq j \leq n$  do
    for each query  $q_k \in Q$ ,  $1 \leq k \leq m$  do
      if  $AUM(q_k, a_i) = 1$  &  $AUM(q_k, a_j) = 1$  then
        #Q1 =#Q1+1
        DRA=DRA+ $f_k \cdot (s_i+s_j)$ 
      else
        if  $AUM(q_k, a_i) = 1$  &  $AUM(q_k, a_j) = 0$  then
          #Q2=#Q2+1
          IIA=IIA+ $f_k \cdot s_j$ 
        else
          if  $AUM(q_k, a_i) = 0$  &  $AUM(q_k, a_j) = 1$  then
            #Q2=#Q2+1
            IIA=IIA+ $f_k \cdot s_i$ 
          end_if
        end_if
      end_if
    end_for
  end_for
  merging_profit=#Q1·DRA-#Q2·IIA
  MPM(i,j)=merging_profit
end_for
end_for
end. {getMPM}

```

---



---

**Algorithm 2.** MAVP

---

input:AUM

output: Optimal Vertical Partitioning Scheme (VPS)

begin

```

for each step in PT do
  getMPM(AUM, MPM)
  select two nodes with maximum merging_profit
  merge the nodes
end_for
compute the cost of each step
optimal VPS = step with minimum cost
end. {MAVP}

```

---



location assignment [11]. To determine the optimal  $vp_s_i$  we need to evaluate how VP affects the total query costs, we use the AUM to do this. The cost of a  $vp_s_i$  is composed of two parts: irrelevant attribute access cost and transportation cost.

$$cost(vp_s_i) = IAAC(vp_s_i) + TC(vp_s_i) \tag{3}$$

The irrelevant attribute access cost measures the amount of data from irrelevant attributes to be accessed during a query. The transportation cost provides a measure for transporting between the nodes of the network.

The irrelevant attribute access cost is given by

$$IAAC(vp_s_i) = \sum_{k=1}^p IAAC(fr_k) \tag{4}$$

The AUM of a table has a set of alphanumeric and multimedia attributes  $A=\{a_1, a_2, \dots, a_n\}$ , the size  $s_i$  of each attribute  $a_i$   $S=\{s_1, s_2, \dots, s_n\}$ , a set of queries  $Q=\{q_1, q_2, \dots, q_m\}$ , the frequency  $f_j$  of each query  $q_j$   $F=\{f_1, f_2, \dots, f_m\}$ , a set of elements  $u_{ji}$ , where  $u_{ji}=1$  if query  $q_j$  uses attribute  $a_i$ , or  $u_{ji}=0$  otherwise. The irrelevant attribute access cost of each fragment  $fr_k$  is given by

$$IAAC(fr_k) = \begin{cases} \sum_{q_j \in P_k} f_j \sum_{i|a_i \in fr_k \wedge u_{ji}=0} s_i & \text{if } n_k > 1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Where  $P_k$  is a set of queries which use at least one attribute and access at least one irrelevant attribute of the fragment  $fr_k$ . This is

$$P_k = \{q_j | u_{ji} = 0 \wedge u_{jl} = 1, 1 \leq i, l \leq n_k\} \tag{6}$$

For example,  $IAAC(vp_s_3)$  of AVP in Table 4 is computed as follows.

$IAAC(vp_s_3) = IAAC(fr_1) + IAAC(fr_2) + IAAC(fr_3)$ ,  $IAAC(fr_1)=0$  because  $n_1=1$ ,  $P_1=\{\emptyset\}$  because there is only one attribute in the fragment  $fr_1$ ,  $P_2=\{q_2\}$  because  $q_2$  only uses the alphanumeric attribute *name* but it also has to access the irrelevant multimedia attribute *image* due to both attributes are located in the same fragment.  $P_3=\{q_1, q_2\}$  because  $q_1$  only uses the multimedia attribute *graphic* but it also has to access the irrelevant multimedia attributes *audio* and *video* due to they are located in the same fragment. On the other hand,  $q_2$  uses the attributes *audio* and *video* but it has to access the irrelevant multimedia attribute *graphic*. Therefore,  $IAAC(fr_2)=10*900=9000$ , because  $f_2=10$  and  $s_{image}=900$  and  $IAAC(fr_3)=15*(4100+39518)+10*(500)=659270$ , because  $f_1=15$  and  $f_2=10$  and  $s_{audio}=4100$ ,  $s_{video}=39518$  and  $s_{graphic}=500$ . So  $IAAC(vp_s_3)=0+9000+659270=668270$ .

Given a location assignment we can compute the transportation cost of a  $vp_s_i$ . The transportation cost of  $vp_s_i$  is the sum of the costs of each query multiplied by its frequency squared, i.e.

$$TC(vp_s_i) = \sum_{j=1}^m TC(q_j) \cdot f_j^2 \tag{7}$$

The transportation costs of query  $q_j$  depend on the sizes of the relevant remote attributes and on the assigned locations, which decide the transportation cost factor between every pair of sites. It can be expressed by

$$TC(q_j) = \sum_h \sum_{h'} c_{\lambda(h)\lambda(h')} \cdot s(h') \tag{8}$$

Where  $h$  ranges over the nodes of the network for  $q_j$ ,  $s(h')$  are the sizes of the relevant remote attributes,  $\lambda(h)$  indicates the node in the network at which the query is stored, and  $c_{ij}$  is a transportation cost factor for data transportation from node  $N_i$  to node  $N_j$  ( $i, j \in \{1, \dots, p\}$ ) [11].

For example,  $TC(vps_3)$  of AVP in Table 4 is computed as follows. There are three fragments, so we suppose that there are three nodes  $N_1, N_2, N_3$  and each fragment  $fr_i$  is located in each node  $N_i$ , we also assume that each query is located in the node where the larger attributes that it uses are situated and  $c_{ij} = 1$ . In Fig. 2 is presented the location assignment of  $vps_3$ .

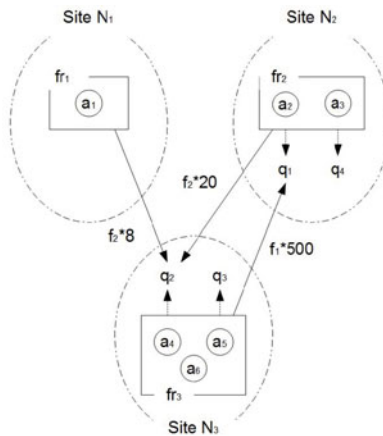


Fig. 2. Location assignment of  $vps_3$

$TC(vps_3) = (s_{graphic} * 15^2) + ((s_{id} + s_{name}) * 10^2) = 115300$  because  $q_1$  accesses the remote multimedia attribute *graphic* and  $q_2$  accesses the remote alphanumeric attributes *id* and *name*. Therefore,  $cost(vps_3) = 668270 + 115300 = 783570$ .

## 5 Experiments

In this section, we show some experimental results to evaluate the efficiency of the MAVP algorithm. In these experiments the cost model proposed in this paper is used as a basis to compare our MAVP algorithm with AVP of Son et al [17]. For the first experiment, considering the information of AUM of table EQUIPMENT from Table 2, we compute the cost of the VPSs obtained by AVP and our MAVP of Table 4 and the results are given in Table 5.

**Table 5.** Cost comparison between AVP and MAVP of the first experiment

VPS	AVP			MAVP		
	IAAC	TC	Cost	IAAC	TC	Cost
vps <sub>1</sub>	1574110	0	1574110	1574110	0	1574110
vps <sub>2</sub>	668550	115300	783850	47200	314500	361700
vps <sub>3</sub>	668270	115300	783570	9200	427000	436200
vps <sub>4</sub>	9000	427800	436800	200	439500	439700
vps <sub>5</sub>	9000	3400300	3409300	0	440300	440300
vps <sub>6</sub>	0	3412800	3412800	0	3412800	3412800

As we can see in Table 5, the VPSs obtained with MAVP have lower cost than AVP in most cases, this is because AVP only considers information about queries with respect to their attributes used and their frequency in the VP process, while MAVP also takes into account the size of the attributes, using this information MAVP can avoid accessing to irrelevant large multimedia attributes and accessing to remote large multimedia attributes, reducing the cost of query execution considerably.

The first and last schemes (vps<sub>1</sub> and vps<sub>6</sub>) are equal for both algorithms. In vps<sub>1</sub> (table not fragmented) the transportation cost is zero and the cost of accessing irrelevant attributes is maximum, this is because all the attributes are located in the same fragment and queries do not access any remote attribute but they scan all the table in order to get the relevant attributes accessing many irrelevant attributes. On the other hand, when the fragments are equal to the number of attributes of the relation (vps<sub>6</sub>), the transportation cost is maximum and the cost to access irrelevant attributes is zero, because each attribute is located in a different fragment, as a result, a query accesses many remote attributes but any irrelevant attribute.

The cost of irrelevant attributes accessed by the queries (IAAC) is considerably reduced in the other schemes obtained by MAVP (vps<sub>2</sub> to vps<sub>5</sub>), this is because AVP focuses on the minimization of access to different fragments and this factor only implies a lower transportation cost, AVP also considers the minimization of irrelevant attributes accessed by the queries, this factor is very important in MMDBs because attributes tend to be of very large size objects and queries usually access only subsets of the attributes in a table, so if we reduce the irrelevant multimedia attributes accessed by the queries we obtain a significant reduction in the cost of the query execution.

AVP finds an optimal VPS when the number of fragments is equal to three, this is  $vps_3 = \{fr_1 = (id), fr_2 = (name, image), fr_3 = (graphic, audio, video)\}$ , but with their cost model which takes into account only the transportation cost. Nevertheless, with our cost model the optimal solution of the AVP schemes is  $vps_4 = \{fr_1 = (id) fr_2 = (name, image) fr_3 = (graphic) fr_4 = (audio, video)\}$  with a cost of 436800. We find a better optimal solution with MAVP when the number

of fragments is equal to two  $vps_2 = \{fr_1 = (id, audio, video), fr_2 = (name, image, graphic)\}$ , with a cost of 361700, this VPS has the lowest cost.

In the second experiment we use the AUM of a traditional database from Table 1, which was used in Navathe et al. [14]. In this case most of the VPSs obtained by MAVP are equal to AVP and both algorithms also obtained the same optimal solution (i.e.,  $vps_3 = \{fr_1 = (a_1, a_5, a_7), fr_2 = (a_2, a_3, a_8, a_9), fr_3 = (a_4, a_6, a_{10})\}$ ) because the size of the attributes is very similar. We modify the size of the attributes and we obtained the VPSs of Table 6.

**Table 6.** Resulting fragments of the second experiment with different size

VPS	Algorithms	
	AVP	MAVP
vps <sub>1</sub>	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>2</sub> ,a <sub>3</sub> ,a <sub>4</sub> ,a <sub>5</sub> ,a <sub>6</sub> ,a <sub>7</sub> ,a <sub>8</sub> ,a <sub>9</sub> ,a <sub>10</sub> )	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>2</sub> ,a <sub>3</sub> ,a <sub>4</sub> ,a <sub>5</sub> ,a <sub>6</sub> ,a <sub>7</sub> ,a <sub>8</sub> ,a <sub>9</sub> ,a <sub>10</sub> )
vps <sub>2</sub>	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>2</sub> ,a <sub>3</sub> ,a <sub>5</sub> ,a <sub>7</sub> ,a <sub>8</sub> ,a <sub>9</sub> ) fr <sub>2</sub> (a <sub>4</sub> ,a <sub>6</sub> ,a <sub>10</sub> )	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>2</sub> ,a <sub>3</sub> ,a <sub>5</sub> ,a <sub>7</sub> ,a <sub>8</sub> ,a <sub>9</sub> ) fr <sub>2</sub> (a <sub>4</sub> ,a <sub>6</sub> ,a <sub>10</sub> )
vps <sub>3</sub>	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>5</sub> ,a <sub>7</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>3</sub> ,a <sub>8</sub> ,a <sub>9</sub> ) fr <sub>3</sub> (a <sub>4</sub> ,a <sub>6</sub> ,a <sub>10</sub> )	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>5</sub> ,a <sub>7</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>3</sub> ,a <sub>8</sub> ,a <sub>9</sub> ) fr <sub>3</sub> (a <sub>4</sub> ,a <sub>6</sub> ,a <sub>10</sub> )
vps <sub>4</sub>	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>5</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>3</sub> ,a <sub>8</sub> ,a <sub>9</sub> ) fr <sub>3</sub> (a <sub>4</sub> ,a <sub>6</sub> ,a <sub>10</sub> ) fr <sub>4</sub> (a <sub>7</sub> )	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>3</sub> ,a <sub>8</sub> ,a <sub>9</sub> ) fr <sub>3</sub> (a <sub>4</sub> ,a <sub>6</sub> ,a <sub>10</sub> ) fr <sub>4</sub> (a <sub>5</sub> ,a <sub>7</sub> )
vps <sub>5</sub>	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>5</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>8</sub> ) fr <sub>3</sub> (a <sub>3</sub> ,a <sub>9</sub> ) fr <sub>4</sub> (a <sub>4</sub> ,a <sub>6</sub> ,a <sub>10</sub> ) fr <sub>5</sub> (a <sub>7</sub> )	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>3</sub> ,a <sub>8</sub> ,a <sub>9</sub> ) fr <sub>3</sub> (a <sub>4</sub> ,a <sub>6</sub> ,a <sub>10</sub> ) fr <sub>4</sub> (a <sub>5</sub> ) fr <sub>5</sub> (a <sub>7</sub> )
vps <sub>6</sub>	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>5</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>8</sub> ) fr <sub>3</sub> (a <sub>3</sub> ,a <sub>9</sub> ) fr <sub>4</sub> (a <sub>4</sub> ,a <sub>6</sub> ) fr <sub>5</sub> (a <sub>7</sub> ) fr <sub>6</sub> (a <sub>10</sub> )	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ) fr <sub>3</sub> (a <sub>3</sub> ,a <sub>9</sub> ,a <sub>8</sub> ) fr <sub>4</sub> (a <sub>4</sub> ,a <sub>6</sub> ,a <sub>10</sub> ) fr <sub>5</sub> (a <sub>5</sub> ) fr <sub>6</sub> (a <sub>7</sub> )
vps <sub>7</sub>	fr <sub>1</sub> (a <sub>1</sub> ,a <sub>5</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>8</sub> ) fr <sub>3</sub> (a <sub>3</sub> ,a <sub>9</sub> ) fr <sub>4</sub> (a <sub>4</sub> ) fr <sub>5</sub> (a <sub>6</sub> ) fr <sub>6</sub> (a <sub>7</sub> ) fr <sub>7</sub> (a <sub>10</sub> )	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ) fr <sub>3</sub> (a <sub>3</sub> ,a <sub>9</sub> ) fr <sub>4</sub> (a <sub>4</sub> ,a <sub>6</sub> ,a <sub>10</sub> ) fr <sub>5</sub> (a <sub>5</sub> ) fr <sub>6</sub> (a <sub>7</sub> ) fr <sub>7</sub> (a <sub>8</sub> )
vps <sub>8</sub>	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ,a <sub>8</sub> ) fr <sub>3</sub> (a <sub>3</sub> ,a <sub>9</sub> ) fr <sub>4</sub> (a <sub>4</sub> ) fr <sub>5</sub> (a <sub>5</sub> ) fr <sub>6</sub> (a <sub>6</sub> ) fr <sub>7</sub> (a <sub>7</sub> ) fr <sub>8</sub> (a <sub>10</sub> )	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ) fr <sub>3</sub> (a <sub>3</sub> ) fr <sub>4</sub> (a <sub>4</sub> ,a <sub>6</sub> ,a <sub>10</sub> ) fr <sub>5</sub> (a <sub>5</sub> ) fr <sub>6</sub> (a <sub>7</sub> ) fr <sub>7</sub> (a <sub>8</sub> ) fr <sub>8</sub> (a <sub>9</sub> )
vps <sub>9</sub>	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ) fr <sub>3</sub> (a <sub>3</sub> ,a <sub>9</sub> ) fr <sub>4</sub> (a <sub>4</sub> ) fr <sub>5</sub> (a <sub>5</sub> ) fr <sub>6</sub> (a <sub>6</sub> ) fr <sub>7</sub> (a <sub>7</sub> ) fr <sub>8</sub> (a <sub>8</sub> ) fr <sub>9</sub> (a <sub>10</sub> )	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ) fr <sub>3</sub> (a <sub>3</sub> ) fr <sub>4</sub> (a <sub>4</sub> ) fr <sub>5</sub> (a <sub>5</sub> ) fr <sub>6</sub> (a <sub>6</sub> ,a <sub>10</sub> ) fr <sub>7</sub> (a <sub>7</sub> ) fr <sub>8</sub> (a <sub>8</sub> ) fr <sub>9</sub> (a <sub>9</sub> )
vps <sub>10</sub>	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ) fr <sub>3</sub> (a <sub>3</sub> ) fr <sub>4</sub> (a <sub>4</sub> ) fr <sub>5</sub> (a <sub>5</sub> ) fr <sub>6</sub> (a <sub>6</sub> ) fr <sub>7</sub> (a <sub>7</sub> ) fr <sub>8</sub> (a <sub>8</sub> ) fr <sub>9</sub> (a <sub>9</sub> ) fr <sub>10</sub> (a <sub>10</sub> )	fr <sub>1</sub> (a <sub>1</sub> ) fr <sub>2</sub> (a <sub>2</sub> ) fr <sub>3</sub> (a <sub>3</sub> ) fr <sub>4</sub> (a <sub>4</sub> ) fr <sub>5</sub> (a <sub>5</sub> ) fr <sub>6</sub> (a <sub>6</sub> ) fr <sub>7</sub> (a <sub>7</sub> ) fr <sub>8</sub> (a <sub>8</sub> ) fr <sub>9</sub> (a <sub>9</sub> ) fr <sub>10</sub> (a <sub>10</sub> )
s <sub>1</sub> = 8, s <sub>2</sub> = 20, s <sub>3</sub> = 20, s <sub>4</sub> = 15, s <sub>5</sub> = 10, s <sub>6</sub> = 250		
s <sub>7</sub> = 900, s <sub>8</sub> = 500, s <sub>9</sub> = 4100, s <sub>10</sub> = 39518		

The cost comparison of the schemes obtained with AVP and MAVP is presented in Table 7. The VP schemes of AVP remain static when the size of the attributes changes. Nevertheless, MAVP adapts the schemes according to the changes in the size of the attributes. Therefore, the schemes and the best-fit (optimal) solution obtained by AVP are the same even when the size of the attributes have changed. MAVP finds different schemes, in this case only the first three schemes are equal for both algorithms. Using the AVP cost model the best-fit solution is still  $vps_3$ , but with our cost model we get a better optimal solution  $vps_2 = \{fr_1 = (a_1, a_2, a_3, a_5, a_7, a_8, a_9), fr_2 = (a_4, a_6, a_{10})\}$ . In this case MAVP finds the same optimal solution, but again MAVP generates VPSs with lower cost in more cases (vps<sub>5</sub> to vps<sub>9</sub>).

**Table 7.** Cost comparison between AVP and MAVP

VPS	AVP			MAVP		
	IAAC	TC	Cost	IAAC	TC	Cost
vps <sub>1</sub>	8001555	0	8001555	8001555	0	8001555
vps <sub>2</sub>	502750	927000	1429750	502750	927000	1429750
vps <sub>3</sub>	188130	2137750	2325880	188130	2137750	2325880
vps <sub>4</sub>	165000	2149000	2314000	187850	2147750	2335600
vps <sub>5</sub>	0	3774000	3774000	165000	2154000	2319000
vps <sub>6</sub>	0	3999250	3999250	164200	2216500	2380700
vps <sub>7</sub>	0	3999250	3999250	0	3779000	3779000
vps <sub>8</sub>	0	4004250	4004250	0	3854000	3854000
vps <sub>9</sub>	0	4004250	4004250	0	3866750	3866750
vps <sub>10</sub>	0	4079250	4079250	0	4079250	4079250

## 6 Discussion

We compare MAVP versus AVP and we obtained VPSs with lowest cost using MAVP in more cases. Thus, the size of attributes is a very important factor to consider in the VP process in order to get the optimal solution in distributed MMDBs. The main advantages of MAVP over other approaches are:

1. Most VP algorithms [2], [20], [8], [12], [13, 15, 16], [17] do not consider the size of the attributes as input in the vertical partitioning process. Therefore, they are not suitable for MMDBs where size of attributes is very varied. MAVP takes into account the size of the attributes to generate vertical partitioning schemes which reduce the query processing cost considerably.
2. MAVP generates the optimal solution using a cost model based on the savings of the access to irrelevant attributes (reducing disk accesses) and the access to remote attributes (reducing transportation cost), while the cost model of other VP approaches [6], [20] only consider the savings in number of disk accesses, disregarding the transportation cost which is a major cost in distributed MMDBs.

## 7 Conclusion and Future Work

Vertical partitioning can provide efficient retrieval of multimedia objects in distributed MMDBs. The novel aspects of our research include the following research contributions. First, a VP algorithm for distributed MMDBs has been developed which takes into account the size of the attributes to generate an optimal vertical partitioning scheme. Second, a cost model for distributed MMDBs has been proposed, this cost model considers that the overall query processing cost in a distributed multimedia environment consists of irrelevant attribute access cost and transportation cost. An experimental evaluation shows that our algorithm outperforms AVP in most cases.

We assumed in this research that the queries that run against the MMDB are static. Distributed MMDBs are accessed by many users simultaneously, therefore queries tend to change over time and a good VPS can be degraded resulting in very long query response time. Present research can be extended to derive the vertical partitioning dynamically in MMDBs, based on the changes in the queries. Thus the VPS of the MMDB can be adaptively modified to achieve efficient retrieval of multimedia objects all the time. In the future we also want to develop an algorithm for hybrid partitioning in MMDBs and to extend MAVP to Object-Oriented databases.

## References

1. Guinepain, S., Gruenwald, L.: Automatic Database Clustering Using Data Mining. In: DEXA 2006 (2006)
2. Bellatreche, L., Simonet, A., Simonet, M.: Vertical Fragmentation in Distributed Object Database Systems with Complex Attributes and Methods. In: DEXA 1996 (1996)
3. Khan, S.I., Hoque, A.S.M.L.: A New Technique for Database Fragmentation in Distributed Systems. *International Journal of Computer Applications* 5(9), 20–24 (2010)
4. Yu, C., Brandenburg, T.: Multimedia Database Applications: Issues and Concerns for Classroom Teaching. *The International Journal of Multimedia and its Applications (IJMA)* 3(1) (2011)
5. Lu, G.: *Multimedia Database Management Systems*. Artech House computing library (1999)
6. Fung, C., Karlapalem, K., Li, Q.: Cost-driven Vertical Class Partitioning for Methods in Object Oriented Databases. *The VLDB Journal* 12(3), 187–210 (2003)
7. Kwok, Y., Karlapalem, K., Ahmad, I., Pun, N.M.: Design and Evaluation of Data Allocation Algorithms for Distributed Multimedia Database Systems. *IEEE Journal on Selected Areas and Communications* 14(7) (1996)
8. Chakravarthy, S., Muthuraj, J., Varadarajan, R., Navathe, S.: An Objective Function for Vertically Partitioning Relations in Distributed Databases and its Analysis. *Distributed and Parallel Databases* 2(21), 183–207 (1994)
9. Chbeir, R., Laurent, D.: Towards a Novel Approach to Multimedia Data Mixed Fragmentation. In: *Proc. of the Int. Conf. on Manage. of Emergent Digital EcoSyst, MEDES* (2009)
10. Özsu, M.T., Valduriez, P.: *Principles of Distributed Database Systems*, 3rd edn. Springer, Heidelberg (2011)
11. Ma, H.: *Distribution Design for Complex Value Databases*. PhD Thesis, Massey University (2007)
12. Hoffer, J.A., Severance, D.G.: The Use of Cluster Analysis in Physical Database Design. In: *Proc. of the 1st VLDB Conf.*, pp. 69–86 (1975)
13. Navathe, S., Ra, M.: Vertical Partitioning for Database Design: A Graphical Algorithm. In: *Proc. of ACM SIGMOD* (1989)
14. Navathe, S., Ceri, S., Wiederhold, G., Dou, J.: Vertical Partitioning Algorithms for Database Design. *ACM TODS* 4, 680–710 (1984)
15. Son, J.H., Kim, M.H.:  $\alpha$ -Partitioning Algorithm: Vertical Partitioning Based on the Fuzzy Graph. In: Mayr, H.C., Lazanský, J., Quirchmayr, G., Vogel, P. (eds.) *DEXA 2001*. LNCS, vol. 2113, pp. 537–546. Springer, Heidelberg (2001)

16. Marir, F., Najjar, Y., Alfaress, M., Abdalla, H.I.: An Enhanced Grouping Algorithm for Vertical Partitioning Problem in DDBS. In: 22nd Int. Symposium on Computer and Information Sciences, pp. 39–44 (2007)
17. Son, J.H., Kim, M.H.: An Adaptable Vertical Partitioning Method in Distributed Systems. *J. of Syst. and Software* 73, 551–561 (2004)
18. Saad, S., Tekli, J., Chbeir, R., Yétongnon, K.: Towards multimedia fragmentation. In: Manolopoulos, Y., Pokorný, J., Sellis, T.K. (eds.) ADBIS 2006. LNCS, vol. 4152, pp. 415–429. Springer, Heidelberg (2006)
19. Getahun, F., Tekli, J., Atnafu, S., Chbeir, R.: The Use of Semantic-based Predicates Implication to Improve Horizontal Multimedia Database Fragmentation. In: Workshop on The many Faces of Multimedia Semantics (MS), pp. 29–38 (2007)
20. Fung, C., Leung, E.W., Li, Q.: Efficient Query Execution Techniques in a 4DIS Video Database System for eLearning. *Multimedia Tools and Applications* 20, 25–49 (2003)

# Diffusion in Dynamic Social Networks: Application in Epidemiology

Erick Stattner, Martine Collard, and Nicolas Vidot

LAMIA Laboratory  
University of the French West Indies and Guiana  
France

estattne, mcollard, nvidot@univ-ag.fr  
<http://lamia.univ-ag.fr>

**Abstract.** Structure and evolution of networks have been areas of growing interest in recent years, especially with the emergence of Social Network Analysis (SNA) and its application in numerous fields. Researches on diffusion are focusing on network modeling for studying spreading phenomena. While the impact of network properties on spreading is now widely studied, involvement of network dynamicity is very little known. In this paper, we address the epidemiology context and study the consequences of network evolutions on spread of diseases. Experiments are conducted by comparing incidence curves obtained by evolution strategies applied on two generated and two real networks. Results are then analyzed by investigating network properties and discussed in order to explain how network evolution influences the spread. We present the *MIDEN* framework, an approach to measure impact of basic changes in network structure, and *DynSpread*, a 2D simulation tool designed to replay infections scenarios on evolving networks.

**Keywords:** Information Spreading, Dynamic network, Evolution, Framework, Simulation.

## 1 Introduction

Network modeling involves a set of items, represented by nodes (also called vertices), that are linked by connections. While seminal works were first conducted in mathematics through the graph theory, this domain has known a significant growth in recent years.

In last decades, network analysis has been the subject of an active research domain, so-called “*Science of Networks*” [25], an emerging scientific discipline that focuses on relationships maintained between entities, and not on entities themselves. Social science, with the occurrence of WEB 2.0 and Social Network Analysis (SNA), has provided the most popular works, with Milgram [19] on small world phenomenon, or Bott [4] on families. Other domains are also related to network analysis, such as biology [16], ethology [11] or computer science [3],



since this kind of representation is particularly fitted to understand information spreading phenomena. For instance spreading of viruses, diseases, rumors, knowledge or fashion behave very similarly [5,12].

This work is focused on diffusion phenomenon and addresses the particular issue of spread of diseases. Indeed, in numerous cases of disease spreading, social contacts seem to be main factors of transmission. Sexual Transmitted Diseases (STDs) are a good example of diseases that depend solely on personal contacts for dissemination. Thus, several studies have focused on social networks and have demonstrated their relevance in disease spreading [18,9,7]. However, although the effect of network properties on spreading is now widely studied, the impact of network changes is an emerging field. One obvious pitfall is the lack of real data and most works commonly handle snapshots of a network at a given time, that do not fully reflect real world networks.

In this paper, we address the issue of dissemination in dynamic networks. Unlike methods that study spreading on static networks, we propose a framework to assess impact of evolving mechanisms, by selecting and comparing some typical evolution strategies. Experiments are conducted by comparing incidence curves obtained with several evolution strategies and are then analyzed by investigating network properties. In this first stage, we have restricted the analysis to strategies that only create new links. Our results provide an original insight about how and why network evolution impacts the spread. We show indeed that new links appearing in the social network generally emphasize the epidemic spread and speed up the process. Of course this assertion has to be modulated according to the evolution strategy applied. In epidemiology, understanding such implications is essential to help public health officials in the prevention and the development of appropriate strategies taking into account changes occurring on real world networks.

This paper is organized in 7 sections as follows. In Section 2 we present main previous works on networks, with particular emphasis on their application in epidemiology. Section 3 presents our motivations and objectives, and details the framework we propose to measure the involvement of network dynamics. Experiments and results are presented in Section 4. In Section 5, we discuss these results by investigating both effects of evolution strategies and changes on network properties. Section 6 is devoted to *DynSpread*, the 2D simulation tool that implements our framework. We conclude and present future directions in Section 7.

## 2 Previous Works

Epidemiology is the science that focuses on infectious diseases. De and Das [12] defined epidemic theory as being “*the study of the dynamics of how contagious diseases spread in a population, resulting in an epidemic*”. More formally, epidemiology is the study of patterns of health and illness and associated factors at population or individual level. It refers to all methods of modeling [24], analysis [22] or monitoring [7] of the spread in a given system, for identifying risk factors and determining optimal intervention approaches to clinical practice and preventative medicine.

### 2.1 Compartment Models

Although the biological interest for this phenomenon is undeniable, other scientific communities have contributed to its understanding: anthropology [4], mathematics [17] or computer science [10,3].

Mathematicians were the first to address modeling issues, through compartment models [24]. This kind of models assume that (1) a population can be divided into a set of compartments, according to the level of the disease development, and (2) individuals have equal probability to change compartment. The two main compartment models defined in epidemic literature are *Susceptible – Infected* Model (*SI Model*), that assumes individuals may become infected with probability  $\beta$  and *SIR Model*, which adds a *Recover* state reached by infected individuals with a certain probability  $\gamma$ . On the same paradigm, many other models [24] can be found in the literature: *SIS*, *SIRS*, etc.



Fig. 1. Two Examples of Compartments Models

However, such approaches remain very simple and do not reflect the real complexity of human interactions. Indeed, in real world people are actually connected to a small portion of individuals, and this portion is obviously not chosen randomly. Introduced by pioneering works of Klondahl [18] on AIDS, network modeling have found various applications in epidemiology, since a significant factor of the outbreak and the behavior of diseases is the structure and the nature of human interactions through which it spreads.

### 2.2 Networks and Epidemics

Traditionally, a network is described by a graph  $G$ , defined  $G = (V, E)$ , where  $V$  is the set of vertexes and  $E$  the set of edges in the graph  $E \subseteq V \times V$ . The neighbors  $N(i)$  of vertex  $i$  is defined as  $N(i) = \{j \mid ij \in E\}$ . Other individual measures prove to be interesting for networks of very different types [20]. The *Degree* is the number of neighbors of a vertex. More formally, the degree  $d_i$  of node  $i$  is the cardinality of the set of its neighbors, *i.e.*  $d_i = |N(i)|$ . The *Clustering Coefficient*  $C_i$  of a node  $i$  indicates how close the neighbors of node  $i$  are to being a clique,  $C_i = \frac{2t_i}{d_i(d_i-1)}$ , where  $t_i$  is the number of triangles for which node  $i$  is a part.

Initially, networks were studied with the objective to understand various real systems in disciplines ranging from communication networks to ecological webs. Newman [20] classifies works according to three categories: Node-Based Measure, Statistical Properties of Networks and Dynamics of Networks.

**At node level**, networks are characterized by some individual properties of their nodes. Typical works try to classify or identify the role of nodes by

understanding which individuals are more connected to others or have most influence, or whether and how individuals are connected to one another through the network, etc. In epidemiology, such measures have been used to identify high-risk individuals. For example, Christley et al. [9] compare various node-based measures to identify this kind of individuals. Chen et al. [6] show the changes in the degree of nodes according to several kinds of interactions and their role for the transmission.

**At global level**, scientists try to classify networks focusing on the distribution of given properties. A typical example is the network classification we can find in the literature [1,20]. In epidemiology, works have been conducted to understand the propagation phenomena [14] and propose intervention strategies taking into account network topology [22,7].

**According to the dynamic point of view**, many recent studies try to understand and reproduce the manner in which a network evolves [25]. Thus, several models have been proposed to reflect growth processes inducing to particular structural features observed on real world networks [20,13]. However, it is interesting to note that the issues of *dynamics of networks* and the *dynamics on networks* are still independent fields. Indeed, the impact of network evolution on spread of infectious diseases is a very new research axis. In this area, the mathematical approach of Gross et al. [15] studies the impact of links deletion on spreading. Read et al. [21] show how changes in the frequency of encounters between individuals may impact the dissemination. More recently, Christensen et al. [8] have measured the effect of changes in demographic attributes within population on the disease transmission.

In this paper, we focus on this last issue of dynamicity in networks. The next section is devoted to the method we propose to understand how different changes in a network have consequences on the transmission of disease.

### 3 Objectives and Method

Networks are alive and animated objects, in which nodes can appear and disappear, links can be created, removed, or can even evolve. Thus, focusing on the dynamic issue in networks inevitably raises multiple questions, both on networks and on information diffusion: How does a network evolve? What do the changes operate on its properties? How is the network topology influenced by the way the network evolves and how does it affect the spread behavior?

#### 3.1 Motivations and Objectives

Among social networks, we can identify on one hand, static structures that do not meet much evolution. For instance, co-author networks have few new links created. But on the other hand, networks based on geographic contacts are likely to meet much evolution with frequent deletion and creation of links, according to individual mobility. We focus on this kind of networks.

In epidemiology, a concrete motivation to study this question is demonstrated by intervention strategies that are currently proposed and are generally focused

on node-based measures. For example, the intervention strategy that gives best results is to vaccinating individuals with the highest degree. However, it is realistic to think that individuals with the highest degree at time  $t$  will probably not be in the same state at time  $(t + 1)$ , due to changes that occur in the network. Therefore, dynamic appears to have a strong and real impact on the spread, and may be an essential factor for the behavior of a disease.

Nevertheless, to the best of our knowledge, no empirical or even comparative studies to assess the effect of different network evolutions are available. Indeed, we may assume that dynamic plays an important role in the diffusion of information through the network, and therefore should be taken into account to understand diseases behavior in evolving networks and propose suitable strategies to prevent and control epidemics. Thus, our objectives are **define a framework** to assess impacts of network evolution **measure and compare** the impact of network dynamic, by comparing the incidence curves induced by several evolution strategies, and **understand causes and effects** at global and individual levels.

### 3.2 Evolution Strategies

In order to carry out such a work, we compare the effects of four well known evolution models, accepted in the literature as reproducing changes observed in real world networks. These strategies are highlighted by schemes on Figure 2.

**Random (R)** is a process that creates links randomly between nodes. This evolution mechanism is particularly used to model evolution of networks for which we have no knowledge on the development.

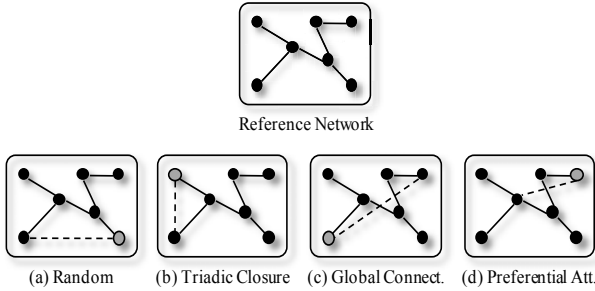
**Triadic Closure (TD)** is one of the first social networking concepts. It can be expressed as “*friends of my friends become my friends*”. More formally, it is the process through which a node is likely to create a link with the neighbors of its neighbors.

**Global Connection (GC)** corresponds to an abstracted mechanism of social link formation, through which a node creates links solely outside of its circle of close friends *i.e.* beyond friends of its friends.

**Preferential Attachment (PA)** is a kind of evolution in which a node is more likely to connect to one with high degree. Although it is fairly recent, it is now widely studied, particularly in social science where it makes sense. Indeed it is more likely for a person to connect to someone with a large number of connections, as those people tend to be more social and popular.

### 3.3 Studied Networks

In this work, we focus on four networks: two generated networks *GEN1*, *GEN2* and two real networks *HS* and *PL*. We have chosen these networks because they are representative of different types of networks currently referenced in the literature.



**Fig. 2.** Starting from a reference network, examples of links generated with (a)Random, (b)Triadic Closure, (c)Global Connection, (d)Preferential Attachment

- **GEN1 and GEN2** are respectively obtained by Erdos-Renyi [13] and Barabasi-Albert [20] models. *GEN1* is a classic random network that has been the subject of intensive researches. *GEN2* represents the kind of network most commonly observed in real world networks, such as the Internet, telephone calls network, sexual network or friendship network, known as scale-free network.
- **HS and PL** are respectively obtained by real and synthetic situations. *HS* correspond directly to the high school interactions graph proposed by Salathe [23] and *PL* represents a synthetic population of the city of Portland extracted from *EpiSims*, an epidemiological simulation system prior to *EpiSimdemics* [3].

Generated networks are used because the work we are conducting is not confined to epidemiology, since mechanisms of transmission of information, or rumors are very similar. Therefore, it is important to understand how the disease behaves on evolving generic networks, to transpose it in other paradigms. Figure 3 details the main characteristics of the networks described above ( $\#comp$  is the number of connected components and  $cc$  is the clustering coefficient of the network).

### 3.4 MIDEN Framework

For each network, we have no a priori knowledge about their evolutions. This allows us to make several assumptions about its development by fitting the network with evolution mechanisms. Thus, to assess the impact of network changes, we apply each evolution strategy (*R*, *TD*, *GC* and *PA*) to these networks, and compare their effect. In this preliminary work, we study the dynamics in an empirical manner, by setting the number of nodes and by considering the dynamic through the addition of new links only.

Let us give  $T = \langle t_0, t_1, \dots, t_m \rangle$ , as the time sequence over which the disease transmission is studied with  $\forall j \in [0..m]$ ,  $t_j < t_{j+1}$  and  $G = \langle G_{t_0}, G_{t_1}, \dots, G_{t_m} \rangle$  as the sequence of networks, where each  $G_{t_j} = (V, E_{t_j})$  represents the state of

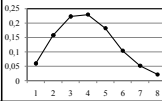
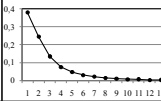
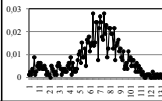
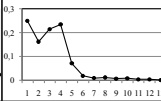
		Networks							
		GEN1	GEN2	HS	PL				
general	Origine	Generated	Generated	High School	Portland				
	#nodes	4771	3233	788	4829				
	#links	7481	5154	26801	7455				
	Density	0.000657	0.000986	0.086873	0.0006395				
	#comp	17	1	1	1				
degree	avg	3.136	3.188	68.195	3.087				
	max	11	118	174	17				
	Distribution								
		cc	0.00056	0.00427	0.49877	0.60880			

Fig. 3. Detailed Networks Features

the network at time  $t_j$ , with  $t_j \in T$ .  $V$  is the set of individuals and  $E_{t_j}$  is the set of edges present in the network at time  $t_j$ , with  $E_{t_{(j-1)}} \subseteq E_{t_j} \subseteq V \times V$ .

Transmission of an infectious agent is simulated according to the *Suceptible – Infectious – Recover* model.

Let us denote  $F_{t_j}$  the function that returns the state of a node  $i$  at time  $t_j$ ,  $F_{t_j} : V \rightarrow \{S, I, R\}$ . Thus, at each time  $t_j$  of  $T$ , we have the set of infected nodes  $I_{t_j} = \{v \in V | F_{t_j}(v) = I\}$ . Let us also define  $N_{t_j}^i$  as the set of infected neighbors of the node  $i$ ,  $N_{t_j}^i = \{v \in V | iv \in E_{t_j} \text{ and } F_{t_j}(v) = I\}$  at  $t_j$ .

Let  $\beta$  be the probability of transmission per contact and  $\gamma$  the probability of recover. We denote  $W$  the applied evolution strategy and  $Q$  the speed of evolution, *i.e.* the number of links created at each iteration.

**1. Outbreak:** Before introduction of the infectious agent, at  $t_0$ , the entire population is assumed to be susceptible and we assume that no link is created yet in the network  $G_{t_0}$ . We begin by selecting the patient zero, the first individual that will be contaminated by the pathogen. This step simply consists in randomly selecting an individual,  $z$ , among the population and change his state  $S$  to  $I$ , *i.e.*  $\exists! z \in V, F_{t_0}(z) = I$  and  $\forall v \in V - \{z\}, F_{t_0}(v) = S$ .

**2. Wave Transmission ( $S \rightarrow I$ ):** Once the patient zero is infected, disease can spread, with a certain probability, from an infected individual to a susceptible individual if there exists an edge between them. More formally, if a susceptible node  $n_i$  has  $k_i$  infected neighbors at time  $t_j$ , it can become infected with the probability  $1 - (1 - \beta)^{k_i}$  with  $k_i = |N_{t_j}^i|$ . In this way, the probability of being infected increases with the number of infected neighbors.

**3. Network evolution :** After the wave transmission, the proportion of infected individuals is stored and the network evolves to the  $G_{(t_j+1)}$  state, where  $(t_j + 1) \in T$ . The evolution is carried out according to the strategy  $W$  chosen and the evolution speed  $Q$ . In practical terms,  $Q$  nodes are randomly selected from the network and link building is applied from these nodes.

**4. Recovery ( $I \rightarrow R$ ):** Each infected individual has a probability  $\gamma$  to recover. Once the node is in  $R$  state, it cannot transmit the infectious agent again. Its immunity is supposed to be permanent and it cannot return in the  $S$  state.

Processes 2, 3 and 4 are repeated until susceptible individuals become fully extinct, *i.e.*  $I_{t_j} = \emptyset$ . Once the epidemic is over, duration of infection is stored. The algorithm below sums up the proposed approach.

**Data:** Probability of transmission  $\beta$ , and probability of recover  $\gamma$   
**Input:** Network  $G$ , Strategy  $W$ , and Evolution Speed  $Q$   
**Result :** List  $L$ , of infected individuals during  $T$

**Function** MIDEN(  $G$  : Network,  $W$  : Strategy,  $Q$  : Speed ) : List

```

|  $L$  : List  $\leftarrow \emptyset$ 
|  $t$  : Time  $\leftarrow 0$ 
| Infect Patient Zero
| While ( $I_t \neq \emptyset$ ) do
|   | Infect  $S$ -nodes  $i$  with probability  $1 - (1 - \beta)^{k_i}$ , with  $k_i = |N_t^i|$ 
|   | add  $\frac{|I_t|}{|V|}$  to L
|   | Network evolves to  $G_{(t+1)}$  according to  $W$  and  $Q$ 
|   | Recover  $I$ -nodes  $i$  with probability  $\gamma$ 
|   |  $t \leftarrow t + 1$ 
| done
| return  $L$ 
End

```

**Algorithm 1:** MIDEN(framework for Measure Impacts of Dynamic on Epidemic Networks)

## 4 Experiments and Results

The framework described above was experimented to analyze effects of these different evolution strategies. As a first step, this section shows and analyzes the direct effects of the different strategies on the strength of the epidemic and its appearance in time. Afterwards, in the next section, we deepen our analytical work by investigating explanations on the side of changes in network properties.

### 4.1 Test Bed

Disease behavior depends on many parameters such as the number of initial infected individuals, the probability of transmission or the probability of recover. However, changes in these parameters often influence only the virulence of the epidemic and are quite well known. In the issue we address, the most relevant parameter seems to be the evolution speed of the network. Thus to study the implication of this parameter on the behavior of a disease, we vary *the network*, *the evolution strategy*, and *the evolution speed*.

Epidemics may have varying durations, so we set  $T = 120$ . Thus the data collection was restricted over a period of 120 iterations. The probability of

transmission was set at 0.1 and the probability of recover was set at 0.2, *i.e.*  $\beta = 0.1$  and  $\gamma = 0.2$ . Each test was performed upon 100 runs. Then, the average of the result obtained was calculated. For each test, a single evolution strategy was applied and the evolution speed of the network remained constant. All tests were performed by *DynSpread*, our simulation tool presented in Section 6 and were conducted with the following simulation environment: Intel Core 2 Duo P8600 2.4Ghz, 3Go Ram, Microsoft Windows Vista 32Bits, Java JDK 1.6.

## 4.2 Results

Figure 4 plots resulting an incidence curve for each strategy, according to the kind of network and the speed of evolution ( $x$  axis). For a given network and a given speed, results obtained by the four strategies are presented on the same scheme, through curves representing the percentage of infected nodes at each iteration of the *MIDEN* algorithm, also called incidence curves. The incidence curves obtained by epidemic *without evolution* strategy are always plotted as a reference to compare to the others.

If we first underline common characteristics observed on *GEN1*, *GEN2* and *PL*, we obviously observe the direct impact of evolution speed. Creating new links obviously emphasizes the epidemic spread. More precisely, two main observations can be made on: **(1)** The virulence of the epidemic, since for these three networks, peaks values increase with the speed. **(2)** The timing of epidemics, since we can observe that when the evolution speed increases, epidemic peaks appear earlier.

We can observe specific behavior, depending on speed evolution for each kind of network. *GEN1* and *PL* behave rather similarly. For example, we can observe that a speed of 10 links per iteration is not sufficient to generate an epidemic in networks *GEN1* and *PL*. However, when the *speed* is set at 50, *PA* is the only strategy able to generate an epidemic on these two networks. We notice that after a speed threshold approximately equals to 100, all strategies generate an epidemic peak that is increasing according to the speed.

Although all strategies generate an epidemic on network *GEN2*, at *speed* 10, the difference between strategies is insignificant and results remain very close to those obtained without evolution. From *speed* 50, the same trends, as for *GEN1* and *PL*, are observed: epidemic peak increases according to the speed.

The case of *HS* network is unusual. To understand why different strategies have no effect, we must address the data collection process. Data were collected by wireless sensors grafted on individuals, and interactions were recorded when two sensors were close enough. Although only interactions above 1 min were considered, the resulting network is very dense and thus allows a good spreading of the disease (in our test bed). We can conclude that for this kind of network, impact of dynamic is negligible. Moreover, it was impossible to generate the results for strategy *GC*, since all nodes rapidly become directly connected.

Finally, when the dynamic is high enough, common trends can be observed: *PA* strategy generally gives an epidemic curve with a peak that systematically is higher than in the others three strategies. It induces the earliest occurrence of the epidemic. *R* and *GC* strategies are always very close, since they have



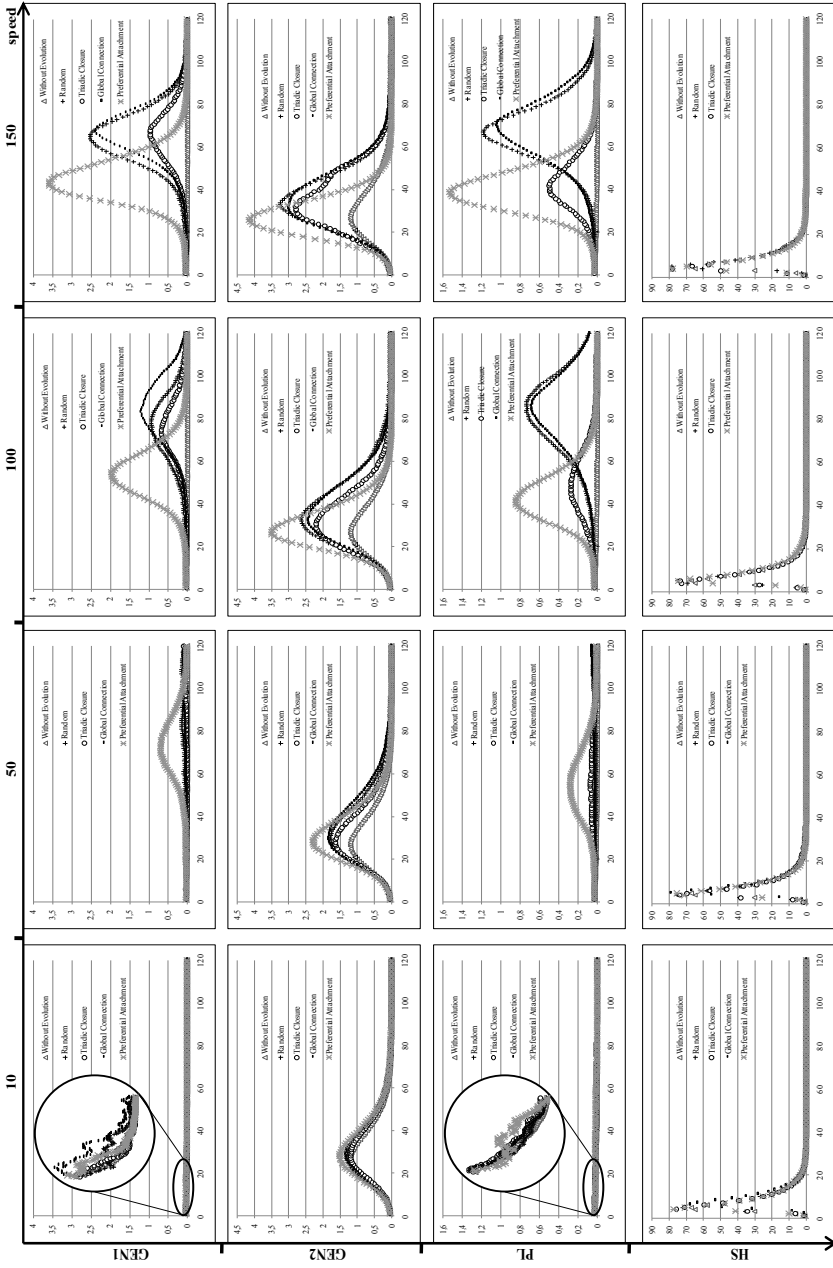
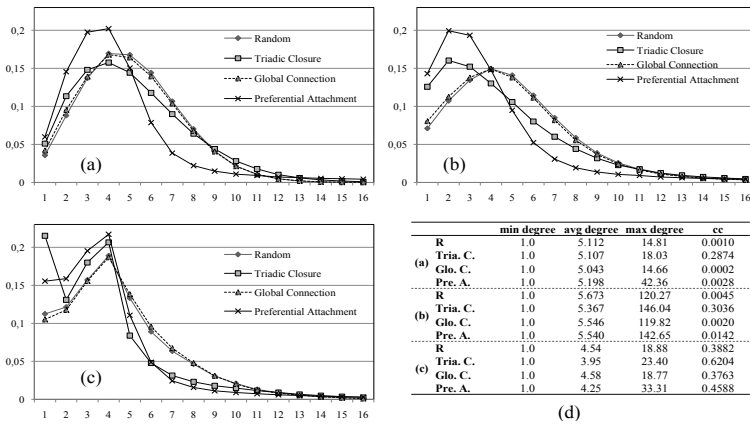


Fig. 4. Incidence curves according to the Network and the Evolution Speed

epidemic curves that follow the same variations in time. The peak obtained by *TD* strategy is always the lowest.

### 5 Discussion

Since the impact of the dynamic appears quite obviously in the results presented above, we investigate now the question and tend to explain what happens at the network level. For this, we focus on changes that occur on network features. As shown in Figure 4, there is a strong difference between strategies beginning from *speed* 100. Thus we have compared changes occurred on network features according to each strategy *R*, *TC*, *GC* and *PA*, after an epidemic diffusion with *speed* 100 and at time  $t_{120}$  (since we consider 120 iterations on *MIDEN* algorithm). Results are shown in Figure 5, and were obtained by averaging 100 runs. Degree distribution, for each evolution strategy, is plotted for *GEN1* 5(a), *GEN2* 5(b) and *PL* 5(c) and features are depicted on 5(d).



**Fig. 5.** Network properties for each evolution strategy. Degree distribution is plotted for Network (a) *GEN1*, (b) *GEN2* and (c) *PL*, after epidemics (at time  $t_{120}$ ) with *speed* 100.  $x$  axis is the degree and  $y$  axis is the fraction of node with degree  $k$ . For each strategy, main network features are shown in (d), where *cc* is the clustering coefficient

In order to explain the results, we discuss first the effect of strategies on the network properties (1) and then consequences on spread (2).

**(1) Direct effects on network properties.** First of all, by comparing features of original networks presented on Figure 3 and those obtained on Figure 5, it is easy to observe direct effects of evolution strategies. While it is expected that the average degree increases, as our approach only considers the addition of new links, significant differences can be observed on degree distribution and clustering coefficient resulting from each strategy.

“*Random*” seems to tend towards a normal degree distribution, as we can particularly observe on Figure 5(b). For all networks, a high proportion of individuals moderately connected and a lower proportion of individuals weakly

and strongly connected can be observed. Indeed Figures 5(a), (b), (c) present an higher proportion of nodes with degree between 5 and 10 with this strategy. The most significant results is obtained for *GEN2*, where a notable change from the original distribution is highlighted.

“*Triadic Closure*” allows neighbors of a same node to become neighbor themselves. It strengthens links within groups of nodes that result in a significant increase in the overall clustering coefficient: from 0.00056 to 0.2874 for *GEN1* and from 0.00427 to 0.3036 for *GEN2*. Unlike *TC* other strategies may even reduce *cc*. For example, it is the case of network *PL*: at  $t_0$  *cc* is 0.60880, with *R*, *GC* and *PA*, it respectively takes values 0.3882, 0.3763 and 0.4588 at  $t_{120}$ .

“*Global Connection*” allows a node to connect with any node outside its immediate community (*friends of friends*). This explains that, except for *cc*, observed properties with *R* and *GC* strategies are very close. Obviously, *cc* is low for this kind of evolution, since as shown on *cc* values of Figure 5, it does not allow creating “*triangles*” as the *R* strategy is likely to do. For example, at  $t_{120}$  in *GEN1*, *cc* is 0.0002 for *GC*, and is 0.0010 for *R*.

“*Preferential Attachment*” reinforces links of most connected nodes, since nodes prefer to make connections with most popular nodes, as observed on the growth of max degree. For example, at  $t_0$ , max degree of *PL* is 17 (Figure 3), against 18.88 for *R*, 23.40 for *TC*, 18.77 for *GC* and 33.31 for *PA* at  $t_{120}$ .

**(2) Consequences on spread.** As expected, strategies *R* and *GC* provide very similar results on spreading, since their effects on network properties prove to be very similar, as shown on Figure 5.

For networks *GEN1* and *PL*, and *speed* 50, as seen in Section 4.2, *PA* is the only strategy able to generate an epidemic (see Figure 4), because it enables emergence of individuals sufficiently connected to allow the transmission of disease within the network. Indeed, strategies *R*, *GC* and *TD* maintain, at same speed, low connected nodes that do not allow spreading in the network.

Beginning from *speed* 100, the trend is accentuated for *PA* that shows the earliest occurrence of the epidemic peak (see Figure 4, column 100 and 150). This peak is higher than with other strategies because the *max degree* is always very high, as shown on Figure 5: 42.36 for *GEN1*, 142.65 for *GEN2* and 33.31 for *PL*. While the strategy *TC* allows the emergence of highly connected nodes, it also generates a network with a high clustering coefficient. So the epidemic is less virulent, since the transmission occurs mainly within a same community.

Network *GEN2* contains highly connected nodes at  $t_0$  (see Fig. 3), as a scale free network. Its *max degree* is 118 against 11 for *GEN1* and 17 for *PL*. For such a network, even without evolution, epidemic spreads and strategies *R*, *GC*, *TD*, and *PA* show different impacts although this is not noticeable before *speed* 50.

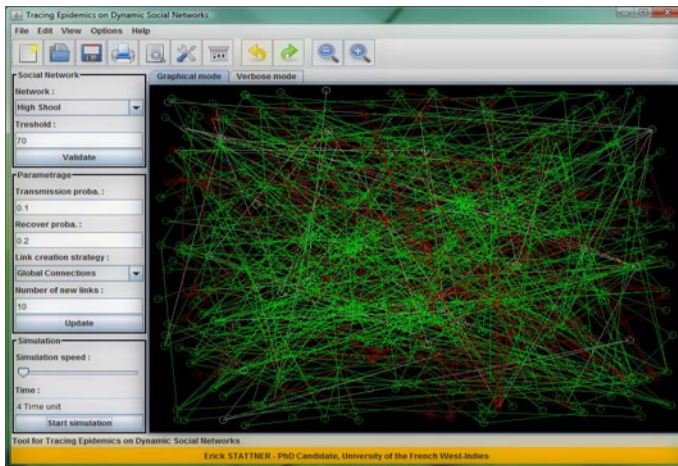
An issue not yet discussed remains the case of the occurrence of the epidemic peak when the network evolves according to *TC* strategy. As shown on Figure 4, the timings of epidemic peaks obtained with *TC*, for networks *GEN2* and *PL*, are close to *PA* peak. However in network *GEN1*, this trend is not confirmed, as we can observe on Figure 4: *TC* epidemic peak appears approximately at the same time than *R* and *GC* peaks. This phenomenon is due to the connected

components contained in  $GEN1$  (17 for  $GEN1$  as shown on Figure 3). Indeed, the  $TC$  strategy does not allow creating links between components. For such an evolution, an epidemic can spread only through the component of patient zero.

Results presented on this section were confirmed even when we vary values of the two parameters transmission or recover probability between 0.1 and 1.

## 6 DynSpread Tool

Common tools designed for simulating spread of diseases in networks do not integrate the ability to manage the evolution of networks. For this reason, we designed DynSpread<sup>1</sup>, a tool for studying the “Dynamic of Spreading”, with the purpose of providing an experimental environment for simulating spread of diseases in evolving networks.



**Fig. 6.** Screenshot of *DynSpread* interface showing an epidemic in High School Network

*DynSpread* is a graphical application that shows two panels: one is used to load the network and calibrate the simulation, the other allows user to monitor the simulation either through a 2D view or a verbose mode. Green nodes correspond to susceptible individuals, red nodes to infected and white nodes to recovered. The *MIDEN* framework described in Section 3 was integrated in *DynSpread* and all tests presented in previous sections were performed with the tool. An example of the *DynSpread* interface, used for simulate the introduction of a disease in High School Network, is shown in Figure 6.

The tool is fully customizable and flexible. First, it is possible to load a network. Afterwards, user can define the probability of transmission, the probability of recover, the evolution strategy, the evolution speed and the simulation speed.

<sup>1</sup> DynSpread: <http://erickstattner.com/DynSpread/>

## 7 Conclusion and Future Works

In this work, we have addressed the question of information spreading and we have focused on the particular issue of disease spreading. Unlike works that handle static networks, we have tackled the emerging and fundamental issue of spreading in evolving networks. Our objectives were to understand how and why the evolution of networks could affect the spread. Our results provide an original contribution on three subjects:

**About dynamic networks**, we have provided original insight on evolution impact on network properties, by showing and comparing changes at features level, according to several network evolution strategies that have been restricted to link creation. We have shown that nodes degree and clustering coefficient can be differently modified from one strategy to another.

**About epidemiology**, this work gives an interesting view about the way a disease spreads through an evolving network. We have highlighted information on the impact of the network dynamics upon epidemics characteristics that should be useful for prevention campaign into communities.

**About information diffusion**, the *MIDEN* framework we have defined to measure impact of basic changes in network structure and the *DynSpread* tool that implements *MIDEN* to simulate diffusion in dynamic networks are flexible enough to fit similar spreading cases such as spreading of rumors, knowledge or fashion. Let us indeed refer to Borner et al. [5] who explained “If we are interested in the spreading of computer viruses, then epidemiological models can be readily applied even though the virus host is now a computer instead of a living being”.

Moreover, *DynSpread* should help scientists and health professionals to have a better understanding of spread mechanisms in real world networks. Our future works in a very short term will be devoted to extending this study in order to capture full evolution strategies with link and node creation/deletion, inspired by real world networks.

## References

1. Albert, R., Barabasi, A.L.: Statistical mechanics of complex networks. *Reviews of Modern Physics* 74, 51 (2002)
2. Barabasi, A.L.: *Linked: The New Science of Networks*. Perseus Books, Cambridge (2002)
3. Barrett, C.L., Bisset, K.R., Eubank, S.G., Feng, X., Marathe, M.V.: *Episimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks*. In: *ACM/IEEE Conference on Supercomputing* (2008)
4. Bott, E.: *Family and social network*, New-York (1957)
5. Brner, K., Sanyal, S., Vespignani, A.: *Network science*. In: Cronin, B. (ed.) *Annual Review of Information Science & Technology*, vol. 41, pp. 537–607 (2007)
6. Chen, Y.-D., Tseng, C., King, C.-C., Wu, T.-S.J., Chen, H.: *Incorporating geographical contacts into social network analysis for contact tracing in epidemiology: A study on taiwan SARS data*. In: Zeng, D., Gotham, I.J., Komatsu, K., Lynch, C., Thurmond, M., Madigan, D., Lober, B., Kvach, J., Chen, H. (eds.) *Intelligence and Security Informatics 2007*. LNCS, vol. 4506, pp. 23–36. Springer, Heidelberg (2007)

7. Christakis, N.A., Fowler, J.H.: Social network sensors for early detection of contagious outbreaks. *PloS one* 5(9)(9) (September 2010)
8. Christensen, C., Albert, I., Grenfell, B., Albert, R.: Disease dynamics in a dynamic social network. *Physica A: Statistical Mechanics and its Applications* 389(13), 2663–2674 (2010)
9. Christley, R.M., Pinchbeck, G.L., Bowers, R.G., Clancy, D., French, N.P., Bennett, R., Turner, J.: Infection in social networks: Using network analysis to identify high-risk individuals. *American Journal of Epidemiology* 162(10), 1024–1031 (2005)
10. Corley, C.D., Mikler, A.R., Cook, D.J., Singh, K.: Dynamic intimate contact social networks and epidemic interventions. *International Journal of Functional Informatics and Personalised Medicine* 1(2), 171–188 (2008)
11. Croft, D.P., James, R., Krause, J.: *Exploring Animals Social Networks*. Princeton University Press, Princeton (2008)
12. De, P., Das, S.K.: Epidemic Models, Algorithms, and Protocols in Wireless Sensor and Ad Hoc Networks, pp. 51–75. John Wiley & Sons, Chichester (2008)
13. Dorogovtsev, S.N., Mendes, J.F.F.: Evolution of networks. *Adv. Phys.* (2002)
14. Gallos, L.K., Liljeros, F., Argyrakis, P., Bunde, A., Havlin, S.: Improving immunization strategies. *Phys. Rev. E* 75(4) (April 2007)
15. Gross, T., D’Lima, C.J., Blasius, B.: Epidemic dynamics on an adaptive network. *Physical Review Letters* 96(20) (2006)
16. Jeong, H., Tombor, B., Albert, R., Oltvai, Z.N., Barabasi, A.-L.: The large-scale organization of metabolic networks. *Nature* 407, 651–654 (2000)
17. Kermack, W.O., McKendrick, A.G.: A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London* 115, 700–721 (1927)
18. Klodahl, A.S.: Social networks and the spread of infectious diseases: the aids example. *Soc. Sci. Med.* 21(11), 1203–1216 (1985)
19. Milgram, S.: The small world problem. *Psychology Today* 1, 61–67 (1967)
20. Newman, M.E.J.: The structure and function of complex networks. *Siam Review* 45, 167–256 (2003)
21. Read, J.M., Eames, K.T.D., Edmunds, W.J.: Dynamic social networks and the implications for the spread of infectious disease. *J. R. Soc. Interface* 5(26) (2008)
22. Salathe, M., Jones, J.H.: Dynamics and control of diseases in networks with community structure. *PLoS Comput. Biol.* 6(4), 04 (2010)
23. Salathe, M., Kazandjieva, M., Lee, J.W., Levis, P., Feldman, M.W., Jones, J.H.: A high-resolution human contact network for infectious disease transmission (2010)
24. Stattner, E., Vidot, N., Collard, M.: Social network analysis in epidemiology: Current trends and perspectives. In: 5th IEEE Internatinal RCIS (2011)
25. Toivonen, R., Kovanen, L., Kivela, M., Onnela, J.P., Saramaki, J., Kaski, K.: A comparative study of social network models: network evolution models and nodal attribute models. *Social Networks* 31(4), 240–254 (2009)

# Probabilistic Quality Assessment Based on Article's Revision History<sup>\*</sup>

Jingyu Han<sup>1</sup>, Chuandong Wang<sup>1</sup>, and Dawei Jiang<sup>2</sup>

<sup>1</sup> School of Computer Science and Technology, Nanjing University of Posts and Telecommunications

Nanjing, P.R. China 210003

[hjysky@gmail.com](mailto:hjysky@gmail.com), [chdwang@njupt.edu.cn](mailto:chdwang@njupt.edu.cn)

<sup>2</sup> School of Computing

National University of Singapore

Singapore 119077

[jiangdw@comp.nus.edu.sg](mailto:jiangdw@comp.nus.edu.sg)

**Abstract.** The collaborative efforts of users in social media services such as Wikipedia have led to an explosion in user-generated content and how to automatically tag the quality of the content is an eminent concern now. Actually each article is usually undergoing a series of revision phases and the articles of different quality classes exhibit specific revision cycle patterns. We propose to Assess Quality based on Revision History (AQRH) for a specific domain as follows. First, we borrow Hidden Markov Model (HMM) to turn each article's revision history into a revision state sequence. Then, for each quality class its revision cycle patterns are extracted and are clustered into quality corpora. Finally, article's quality is thereby gauged by comparing the article's state sequence with the patterns of pre-classified documents in probabilistic sense. We conduct experiments on a set of Wikipedia articles and the results demonstrate that our method can accurately and objectively capture web article's quality.

## 1 Introduction

Nowadays user-generated content (UGC), such as Wikipedia, Facebook, etc. is coming into existence. Each document is collaboratively created and maintained by collaborative writing. This in turn results in how to control the quality levels of data contributed by various users. Among them, automatic data quality assessment is a key issue.

In this paper, we ground our work on Wikipedia as it is a typical collaborative content repository and its quality varies dramatically. According to Wikipedia quality scale<sup>1</sup>, all the articles are classified into seven quality classes, namely Featured Article(FA), A-Class(A), Good Article(GA), B-Class(B), C-Class(C),

---

<sup>\*</sup> The work is fully supported by National Natural Science Foundation of China under grants 61003040, 60903181 and China 973 program of No. 20100471353.

<sup>1</sup> [http://en.Wikipedia.org/wiki/Wikipedia\\_database](http://en.Wikipedia.org/wiki/Wikipedia_database)

Start-Class(ST) and Stub-Class(SU). The fundamental policy identifying article's quality class falls on human judgement. For instance, Wikipedia community constantly review the articles labelling their quality. The disadvantage of this approach is evident. First, the manual assessment will eventually cease to be feasible due to the rapidly growing size of content. Second, human judgement is subject to bias. To overcome the drawbacks of human judgement, an alternative solution is to design automatic or semi-automatic quality assessment policies [1,2] and our work belongs to this category.

We observe that every collaboratively generated article is always going through a series of revision phases, such as building structure, contributing text and so on, gradually reaching a convergence state. Later we use state referring to the phase. An article's current state mainly depends on its previous state whenever a contributor edits the article. In other words, the evolution of the article exhibits Markov property. This revision history can be directly extracted from the UGC site such as Wikipedia's view history pages. Furthermore, different kinds of quality classes exhibit groups of dominant revision cycle patterns. For example, some Featured Articles first go through cycles of building structure, then cycles of contributing text and finally discussing text. In contrast, some Featured Articles go through a series of cycles, each cycle containing a sequence of building structure, contributing text and discussing text.

Based on above observations, we propose a machine learning approach to Assess Quality based on Revision History (AQRH) for a specific domain. The AQRH approach consists of two phases, namely preprocessing and probabilistic quality assessment. In preprocessing phase an article's revision history is formally turned into a sequence of revision states, which clearly describes an article's evolution. In probabilistic quality assessment phase, an article's quality rating is given with two steps. First, each quality class is refined into quality corpora by clustering and each quality corpus is represented by a set of revision cycle patterns which often co-occur. The intuition is to identify *all* the typical groups of dominant revision cycle patterns. Second, each article's rating is given by comparing its own revision sequence with quality corpora. If an article highly matches one corpus, the article belongs to the quality class containing the corpus with a high probability.

Our contributions of the paper include: (1) Hidden Markov Model is employed to formally describe article's evolution. (2) Each quality class is represented by a group of quality corpora, each of which specifies a group of frequently co-occurring revision cycle patterns. (3) Through comparing an article with all quality corpora, the article's quality rating is probabilistically determined.

The remaining part is organized as follows. Section 2 discusses related work. Section 3 formally describes problem setting and how to map an article's revision history into state sequence. The details of quality assessment by machine learning are described in section 4. In section 5, we validate and discuss our method by detailed experiment. Finally the conclusion is given.



## 2 Related Work

Data quality is an important issue to all the content contributor and its evaluation approaches can be divided into two categories. The first category focuses on *qualitatively* analysing data quality dimensions [3,4,5,6]. The second category deals with how to *quantitatively* assess quality of data. The most obvious quality assurance approach is grammar check. The writer's workbench was a program to detect some quality metrics such as split infinitives, overly long sentences, wordy phrases, etc [7]. Literature [8] points out that cohesion is an important measurement of writing quality and proposes to use Latent Semantic Analysis (LSA) algorithm to measure cohesion. The result shows that LSA could be used to achieve human accuracy in holistic judgement of quality but its limitation is that the domain must be well defined and a representative corpus of the target domain must be available.

These years many works focus on Wikipedia article's quality. Literature [2] explores a significant number of quality indicators to assess Wikipedia article quality. Literature [9] propose to use maximum entropy model to learn how to identify Wikipedia article quality. Literature [10] discusses seven IQ metrics which can be evaluated automatically on Wikipedia content. These methods mainly focus on analysing different kinds of quality indicators' effectiveness and they do not touch on how to use revision history. Another work relevant to ours is using revision history to assess the trustworthiness of articles [11,12]. But they focus on article's trustworthiness and do not touch on how to assess data quality.

## 3 Problem Setting and Data Preprocessing

Let  $\mathcal{P}$  be a set of articles. Each  $P \in \mathcal{P}$  is associated with a specific content version  $v_i$  at a particular time  $t_i$ . Formally, we define revision history of  $P$  as a sequence of versions  $\langle (t_1, v_1), (t_2, v_2), \dots, (t_i, v_i), \dots, (t_l, v_l) \rangle$  where

1.  $l$  refers to the number of versions during the whole life of  $P$  and version  $v_i$  results from a revision applied to previous version  $v_{i-1}$  at time  $t_i$ ,  $i \in [1, l]$ .
2.  $v_i \neq v_{i+1}$  for each  $i \in [1, l]$ .

**Definition 1 (Quality Class).** *Quality class is defined by Wikipedia community including Featured Article (FA), A-Class (A), Good Article (GA), B-Class (B), C-Class (C), Start-Class (ST) and Stub-Class (SU).*

The goal of this paper is identifying to which quality class an article belongs based on its revision history. We observe that an article is always evolving from state to state and thus we employ Hidden Markov Model (HMM) to describe this probabilistic process.

**Definition 2 (HMM).** *A HMM is a 5-ary tuple  $\lambda = (\mathcal{H}, \mathcal{O}, \Gamma, \Lambda, \Pi)$ , where  $\mathcal{H} = \{h_1, h_2, \dots, h_n\}$  ( $n > 1$ ) is a set of hidden states,  $\mathcal{O} = \{o_1, o_2, \dots, o_m\}$  ( $m > 1$ ) is a set of observations,  $\Gamma$  is the state transition probability matrix,  $\Lambda$  is the emission probability matrix and  $\Pi$  is the initial state distribution.*

At each time point  $t$ , the model is in a particular hidden state  $q_t \in \mathcal{H}$  and a particular observation  $r_t \in \mathcal{O}$  is observed. Given a sequence of observations and model  $\lambda$ , Viterbi algorithm is employed to find the state sequence that is mostly likely to generate that output observation sequence [13].

We argue that within a specific application domain each quality class has its own state transition and emission probability and these can be learned by Baum-Welch algorithm [14]. We now detail how to model the revision history using HMM.

### 3.1 States of Revision History

We observe that the article is evolving from time to time and at each time the article evolves at one hidden state of  $\mathcal{H}$ . Formally,  $\mathcal{H} = \{\text{building structure, contributing text, discussing text, contributing structure and text, discussing structure and text, content agreement}\}$ . They mean as follows.

**building structure ( $\mathcal{B}$ ):** The architecture elements of the article such as lead section and section headings, are being constructed .

**contributing text ( $\mathcal{C}$ ):** The contributors mainly focus on giving new piece of text to make the article more comprehensive, not giving much attention to the organization and accuracy of text.

**discussing text ( $\mathcal{D}$ ):** The contributors mainly focus on discussing the content of text. From time to time, the contributors add or modify some parts of the text to make it more accurate.

**contributing structure and text ( $\mathcal{F}$ ):** The contributors not only focus on giving new piece of text to make the article more comprehensive, but also focus on constructing new architecture element to make the article well organized.

**discussing structure and text ( $\mathcal{E}$ ):** The contributors discuss both the architecture and content of the article. From time to time, the contributors add or modify some parts of architecture elements or text to make the article more complete and more orderly.

**content agreement ( $\mathcal{A}$ ):** The contributors have a rough agreement on the facts described in the article. But occasionally contributors still give a slight revision that does not affect the users' understanding.

Different articles tend to evolve in different fashions. For example, some articles first go through a series of states regarding building structures then the text is filled step by step. These articles often demonstrate the state sequence such as  $\{\text{building structure} \rightarrow \text{building structure}, \dots, \rightarrow \text{contributing text}, \dots, \rightarrow \text{discussing text}\}$ . While some articles tend to evolve part by part without a predefined global organization and these articles frequently demonstrate the state sequence such as  $\{\text{contributing structure and text} \rightarrow \text{contributing text}, \dots, \text{discussing structure and text} \rightarrow \text{contributing text}, \dots, \text{discussing structure and text} \rightarrow \text{contributing text}, \dots\}$ .

### 3.2 Observations of Revision History

Given an article's history, each revision, namely the increment between two consecutive versions, is extracted by comparing the version with its previous version. Every revision is a *deletion*, *insertion* or *modification* with reference to *structure*, *content*, *format*, *structure+content* and *content+format*. Actually, revision can be captured in different granularity. It is captured as a 2-ary tuple  $(rang^r, perc)$ . Here  $rang^r$  denotes the maximum extent the revision spans, which includes heading level, word level, sentence level, paragraph level, section level and link level.  $perc$  denotes the ratio of size of the revision to the size of the  $rang^r$ . To sum up, a revision is captured in terms of three aspects, namely update type, content type and granularity type. Formally, it is a 3-ary tuple  $(U, C, G)$ . Here  $U$  stands for update type, including *insertion*, *deletion* and *modification*.  $C$  stands for content type, including *structure*, *content*, *format*, *structure+content* and *content+format*.  $G$  stands for revision granularity.

Now the article's observation sequence is defined as

**Definition 3 (Observation Sequence).** *Given a revision history sequence  $\langle (t_1, v_1), (t_2, v_2), \dots, (t_i, v_i), \dots, (t_l, v_l) \rangle$ , its observation sequence is  $\langle o_1, o_2, \dots, o_i, \dots, o_l \rangle$ , where  $o_i$  is the revision between  $v_{i-1}$  and  $v_i$  ( $i \in [1, l]$ ).*

In fact users tend to save intermediate results during editing to avoid the loss of work. Many consecutive revisions are of the same type and are related to the same part. So they can be merged to get a more compact view by Algorithm [11](#).

## 4 Probabilistic Quality Assessment by Learning Patterns

After each article's revision history is mapped into a sequence of revision states, the articles of one quality class is regarded as a database of state sequences. Each quality class exhibits specific revision cycle patterns, which is defined as follows.

**Definition 4 (Revision Cycle Pattern).** *Revision Cycle Pattern is a subsequence or a series of subsequence which occur often in articles' state sequence database.*

Here the subsequence is formally defined as

**Definition 5 (Subsequence and Landmark).** *Sequence  $S' = \langle e'_1 e'_2 \dots e'_m \rangle$  is a subsequence of another sequence  $S = \langle e_1 e_2 \dots e_n \rangle$  ( $m \leq n$ ), denoted by  $S' \sqsubseteq S$ , if there exists a sequence of integers(positions)  $1 \leq l_1 < l_2 \dots < l_m \leq n$  s.t.  $S'[i] = S[l_i]$  (i.e.,  $e'_i = e_{l_i}$ ) and  $|l_i - l_{i-1}| = 1$  for  $i = 2, \dots, m$ . Such a sequence of integers  $\langle l_1, \dots, l_m \rangle$  is called a landmark of  $S'$  in  $S$ .*

One quality class' state sequence database usually exhibits groups of dominant revision cycle patterns which often occur simultaneously. It is defined as quality corpus.

**Algorithm 1.** mergeRevision

---

**Input:** A sequence of observations  $R = \langle o_1, o_2, \dots, o_N \rangle$   
**Output:** A reduced sequence of observations  $R'$   
*// merge the observations of the same granularity*

```

1 Initialization;
2 while not end of R do
3    $o_{last} \leftarrow o_i, o_{cur} \leftarrow o_{i+1};$ 
4   // Only merge the observations made by the same contributor
5   if  $o_{last}.author = o_{cur}.author$  then
6     // merge the granularity of adjacent observations
7     if  $(o_{last}.updatetype = o_{cur}.updatetype) \wedge$ 
8        $(o_{last}.contenttype = o_{cur}.contenttype)$  then
9       if  $same(o_{last}.granu, o_{cur}.granu) \wedge$ 
10          $(both\ o_{last}\ and\ o_{cur}\ relate\ to\ the$ 
11          $same\ article\ part)$  then
12          $o_i \leftarrow mergeGranu(o_{last}, o_{cur});$  // merge into one
13         granularity
14       end
15     end
16    $R' \leftarrow R' \cup o_i; i \leftarrow i+2;$ 
17 end
18 else
19    $R' \leftarrow R' \cup o_i; i ++;$ 
20 end
21 end
22 // merge the consecutive observation pair into one modification
23 observation
24 Reinitialization;
25 while not end of R' do
26    $o_{last} \leftarrow o_i, o_{cur} \leftarrow o_{i+1};$ 
27   // Only merge the observations made by the same contributor
28   if  $o_{last}.author = o_{cur}.author$  then
29     if  $(o_{last}.updatetype = insert \wedge o_{cur}.updatetype = delete) \vee$ 
30        $(o_{cur}.updatetype = insert \wedge o_{last}.updatetype = delete)$  then
31       if  $o_{last}, o_{cur}$  relate to the same part then
32          $o'_i \leftarrow mergeInsertDelete(o_{last}, o_{cur});$ 
33         replace  $o_{last}, o_{cur}$  with  $o'_i$  in R';
34       end
35     end
36    $i ++;$ 
37 end
38 else
39    $i ++;$ 
40 end
41 end
42 return R';

```

---

**Definition 6 (Quality Corpus).** *Quality Corpus is one sub-type of a quality class, which exhibit a group of revision cycle patterns which often occur simultaneously.*

In other words, a quality class is represented as a group of quality corpora.

Based on above understanding, we achieve the probabilistic assessment by machine learning with three steps. First, for each quality class we extract the revision cycle patterns borrowing frequent items mining [15]. Second, each quality class is refined into quality corpora by clustering to discover the dominant revision cycle patterns. Finally, each article’s quality rating is determined by comparing its frequent patterns with all of the learned quality corpora. This three steps are detailed in subsection 4.1, 4.2 and 4.3 respectively.

### 4.1 Extracting Revision Cycle Patterns

Revision cycle patterns are divided into two groups, namely non-gapped revision cycle patterns and gapped revision cycle patterns. The former captures the consecutive revision practice and is formally defined as follows.

**Definition 7 (Non-gapped Revision Cycle Pattern).** *Given a sequence database  $\mathcal{S} = \{S_1, S_2, \dots, S_W\}$ , a frequent subsequence  $S'$  and a threshold  $\alpha$ , if  $S'$  has more than  $\alpha$  instances in  $\mathcal{S}$ ,  $S'$  is a non-gapped revision cycle pattern.*

To capture the dependency of several non-gapped edit subsequence, gapped revision cycle pattern is formally defined by the followings.

**Definition 8 (Gapped Subsequence and Instance).** *Given a series of  $r$  subsequences  $S'_r = \langle \langle e_1^1, e_2^1, \dots, e_{l_1^{len}}^1 \rangle \dots \langle e_1^j, e_2^j, \dots, e_{l_j^{len}}^j \rangle \dots \langle e_1^{j+1}, e_2^{j+1}, \dots, e_{l_{j+1}^{len}}^{j+1} \rangle \dots \langle e_1^r, e_2^r, \dots, e_{l_r^{len}}^r \rangle \rangle$  and a sequence  $S = \langle e_1, e_2, \dots, e_n \rangle$ , if there exists a series of integers (positions)  $lan = \langle \langle l_1^1, l_2^1, \dots, l_{l_1^{len}}^1 \rangle \dots \langle l_1^j, l_2^j, \dots, l_{l_j^{len}}^j \rangle \dots \langle l_1^{j+1}, l_2^{j+1}, \dots, l_{l_{j+1}^{len}}^{j+1} \rangle \dots \langle l_1^r, l_2^r, \dots, l_{l_r^{len}}^r \rangle \rangle$  where  $\langle l_1^j, l_2^j, \dots, l_{l_j^{len}}^j \rangle (1 \leq j \leq r)$  is the landmark of subsequence  $\langle e_1^j, e_2^j, \dots, e_{l_j^{len}}^j \rangle$  and  $l_1^{j+1} - l_{l_j^{len}}^j \geq 2$  holds for all  $j (1 \leq j \leq r)$ .  $S$  is a instance of gapped subsequence  $S'_r$ .*

**Definition 9 (Gapped Revision Cycle Pattern).** *Given a sequence database  $\mathcal{S} = \{S_1, S_2, \dots, S_W\}$ , a gapped subsequences  $S'_r$  and a threshold  $\beta$ , if  $S'_r$  has more than  $\beta$  instances in  $\mathcal{S}$ ,  $S'_r$  is a gapped revision cycle pattern.*

For example, suppose that  $\mathcal{S} = \{S_1, S_2, S_3\}$ ,  $S_1 = 'ABCDABFEABDEFC'D'$ ,  $S_2 = 'ABCDABFEABAEFC'D'$ ,  $S_3 = 'ABCDABFEABDAB'$ ,  $\alpha = 3$  and  $\beta = 2$ . 'AB' is a non-gapped revision cycle pattern because  $S_1, S_2$  and  $S_3$  are all instances of 'AB'. 'AB...EFCD' is a gapped revision cycle pattern because it has two instances  $S_1$  and  $S_2$  in  $\mathcal{S}$ . Actually the first denotes how frequently 'AB' occurs in the sequence database while the latter implies that the frequent pattern 'AB' is often followed by a frequent pattern 'EFCD'. These two types of revision cycle patterns are mined by adapting approaches given in literature [15].

---

**Algorithm 2.** ExtractQualityCorpus

---

**Input:** Revision cycle patterns  $\{F_1, F_2, \dots, F_M\}$  of quality class  $\Sigma$ , total number of corpora  $k$ , scale factor  $A$  and  $B$  ( $A > B$ ), converge threshold  $\Delta$

**Output:** A set of quality corpora  $(\Omega_1, \Omega_2, \dots, \Omega_k)$

// 1. Initialize the medoids

1  $M_{init} \leftarrow$  random patterns from  $\Sigma$  of size  $A.k$ ;

2  $M_{seed} \leftarrow$  selectSeed( $M_{init}$ ,  $B.k$ );

// 2. Iterative Phase

3  $BestGain \leftarrow 0$ ;

4  $M_{best} \leftarrow \phi$ ;

5  $M_{cur} \leftarrow$  random set of medoids  $\{m_1, m_2, \dots, m_k\} \subset M_{seed}$ ;

6 **repeat**

7      $(\Omega_1, \Omega_2, \dots, \Omega_k) \leftarrow AssignPatterns(\Sigma \setminus M_{cur}, M_{cur})$ ;

8      $gainFun \leftarrow \frac{\sum_{i=1}^k \cup cov(\Omega_i)}{\sum_{i,j} \text{olap}(\Omega_i, \Omega_j)}$ ;

9      $S_{bad} \leftarrow \phi$ ;

10    **if**  $gainFun > BestGain$  **then**

11        $BestGain \leftarrow gainFun$ ;

12        $M_{best} \leftarrow M_{cur}$ ;

13        $S_{bad} \leftarrow$  choose the bad ones in  $M_{best}$ ;

14    **end**

15    compute  $M_{cur}$  by replacing  $S_{bad}$  with random points from  $M_{seed}$ ;

16 **until** the variance of  $BestGain$  is within  $\Delta$  in 4 runs;

17 **return**  $(\Omega_1, \Omega_2, \dots, \Omega_k)$ ;

---

During mining, different combination of values of  $\alpha$ ,  $\beta$  produce different set of revision cycle patterns. To find a set of good revision cycle patterns which not only cover all the articles for the quality class but also differ as much as possible from other quality classes. We heuristically select the revision cycle patterns with the following steps.

**Step 1: Generating Candidate Revision Cycle Patterns**

First, sets of revision cycle patterns are generated by exploring different values of  $\alpha$  and  $\beta$ . Second, for each candidate revision cycle pattern group, we remove the redundant patterns or the Non-gapped pattern which is a subsequence of another gapped pattern. Third, suppose that  $\{F_1, F_2, \dots, F_j, \dots, F_K\}$  is one candidate pattern group of quality class  $\Sigma$  and  $art(\Sigma)$  be the articles of this class. Let  $cov(F_j)$  be the set of articles covered by  $F_j$ . We select the pattern group satisfying  $\sum_{j=1}^K cov(F_j) = art(\Sigma)$ .

**Step 2: Selecting the Pattern Group According to Its Uniqueness**

Given a revision cycle pattern  $F_j$  of class  $\Sigma_i$ , its uniqueness (denoted as  $uni(F_j)$ ) is defined as  $\frac{cov(F_j)}{m}$ , where  $m$  is the total number of quality classes whose articles exhibit the pattern  $F_j$ . Intuitively, the larger the  $uni(F_j)$  is, the more discriminating the pattern is. Given one quality class, We select the pattern group with the largest average pattern uniqueness.

---

**Algorithm 3.** selectSeed

---

**Input:** A group of candidate medoids  $M_{init}$  of size  $Ak$ ;  
**Output:** reduced set of candidate medoids  $M$  of size  $Bk$

```

1  $M \leftarrow$  a random sample  $F_1 \in M_{init}$ ;
2 foreach  $F_i \in M_{init} \setminus M$  do
3   |  $gain(F_i) \leftarrow IOG(F_i, F_1)$ ;
4 end
5 for  $i \leftarrow 2$  to  $Bk$  do
6   | let  $F_i \in M_{init} \setminus M$  s.t.  $gain(F_i) \leftarrow \max\{gain(x) | x \in M_{init} \setminus M\}$ ;
7   |  $M \leftarrow M \cup F_i$ ;
8   | foreach  $x \in M_{init} \setminus M$  do
9     |  $gain(x) \leftarrow \min(gain(x), IOG(x, F_i))$ ;
10  | end
11 end
12 return  $M$ ;
```

---

### 4.2 Refining Quality Class into Quality Corpora

Given one quality class  $\Sigma$  and its mined revision cycle patterns  $\{F_1, F_2, \dots, F_M\}$ , we employ algorithm 2 refining quality class into quality corpora. The advantage is that it automatically generates the description of the clusters in terms of revision cycle patterns.

Given one quality class articles  $art(\Sigma)$ , the general principles for finding a group of corpora  $\{\Omega_1, \dots, \Omega_k\}$  is (1)  $\sum_{i=1}^k cov(\Omega_i) = art(\Sigma)$  (this has been ensured during pattern mining); and (2) the overlap between any two corpora of  $\Omega_i$  and  $\Omega_j$  (for  $i \neq j$ ), denoted as  $olap(\Omega_i, \Omega_j)$ , should be minimized. This aim is achieved by algorithm 2. The algorithm proceeds in two phases, namely initialization phase and iterative phase. In the initialization phase we try to find a superset of  $B.k$  medoids using greedy technique which is shown in algorithm 3. The principle is finding the medoids that maximize the coverage and minimize the overlap. The incremental overlap gain ( $IOG$ ) of pattern  $F_i$  over another pattern  $F_j$  is defined as

$$IOG(F_i, F_j) = \frac{(cov(F_i) \cup cov(F_j)) - cov(F_j)}{1 + olap(F_i, F_j)}. \tag{1}$$

In the second phase hill-climbing policy is employed to find a good set of medoids by iteration.

### 4.3 Quality Class Representation and Quality Rating

Now each quality class (denoted as  $\Sigma$ ) consists of a group of quality corpora. That is to say,

$$\Sigma = \{\Omega_1, \Omega_2, \dots, \Omega_W\} \tag{2}$$

where  $\Omega_i (1 \leq i \leq W)$  is a quality corpus. Every quality corpus consists of a group dominant revision cycle patterns, namely  $\Omega_i = (F_1, F_2, \dots, F_j, \dots, F_n)$ .

To measure how close an article  $P$  is to an quality class  $\Sigma_i$  ( $1 \leq i \leq 7$ ), which needs measuring the article’s Quality Similarity with quality Corpus  $\Omega_j \in \Sigma_i$ . This is measured as follows.

$$QSC(P, \Omega_j) = \frac{|fre(P) \cap \Omega_j|}{|fre(P) \cup \Omega_j|} \tag{3}$$

where  $fre(P)$  is the frequent revision cycle patterns exhibited in  $P$ ’ state sequence. Now quality similarity between an article  $P$  and one quality class  $\Sigma_i$  is

$$QS(P, \Sigma_i) = max\{QSC(P, \Omega_j) | \Omega_j \in \Sigma_i\}. \tag{4}$$

Given a set of quality class  $\Sigma_1, \Sigma_2, \dots, \Sigma_n$ , an article  $P$  belongs to quality class  $\Sigma_i$  ( $1 \leq i \leq n$ ) with the probability

$$f_i = \frac{\frac{1}{QS(P, \Sigma_i)}}{\sum_{k=1}^n \frac{1}{QS(P, \Sigma_k)}}. \tag{5}$$

## 5 Experiment Results

We collected a set of English articles from Wikipedia. As a large collaborative encyclopedia, there are approximately 3,321,000 articles (in the English version alone) and some articles were already manually evaluated in terms of their content quality. The data repository is available for downloading with complete revision history. We chose a set of English articles from computing category. We chose this set of articles because the articles have been assigned quality class labels according to Wikipedia editorial team’s quality grading scheme<sup>2</sup>. Table 1 shows the statistics of the dataset. Clearly, ST and SU classes occupy most of the dataset, nearly 81%, and FA, GA, B, C classes occupy the rest. Note that there are no A-Class articles in this dataset.

**Table 1.** Statistics of Computing Dataset

Quality Class	FA	A	GA	B	C	ST	SU
Total number of articles	16	0	55	768	755	3145	4000
Average number of revision pages	3512	0	3623	931	1012	119	62

### 5.1 Data Preprocessing

To extract the observations, data cleansing is done as follows. First, revision entries are extracted from the history view pages and are ordered into sequence by date. Second, reverting subsequence are removed. Third, algorithm 1 is applied to get a more compact observation sequence.

For each quality class’ sequence database, initial state distribution  $\Pi$  is initialized s.t.  $pr(\mathcal{B}) + pr(\mathcal{C}) + pr(\mathcal{F}) = 1$ . In other words, the state sequence always starts with  $\mathcal{B}$ ,  $\mathcal{C}$  or  $\mathcal{F}$  state. Initially, for each start state  $i$ , transition probability

<sup>2</sup> [http://en.wikipedia.org/wiki/Wikipedia:Version\\_1.0\\_Editorial\\_Team/Assessment](http://en.wikipedia.org/wiki/Wikipedia:Version_1.0_Editorial_Team/Assessment)



is evenly distributed among all destination states. So is for emission probability initialization. The learning proceeds until the likelihood difference of the database occurrence is within threshold  $\nabla = 0.2$  (in log form) in 4 consecutive iteration. Due to space we only report the transition probability of FA class in table 2. When the parameters were set, Viterbi algorithm was employed to transform observation sequence into revision state sequence.

**Table 2.** State Transition Probability for FA Class

Init. State	Destination State					
	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{F}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{A}$
$\mathcal{B}$	0.2041	0.034	0.0408	0.3673	0.1088	0.2449
$\mathcal{C}$	0.0606	0.0960	0.0657	0.5303	0.1061	0.1414
$\mathcal{F}$	0.0625	0.0938	0.0625	0.4063	0.2396	0.1354
$\mathcal{D}$	0.0550	0.1079	0.0296	0.5915	0.0974	0.1185
$\mathcal{E}$	0.0687	0.0773	0.1202	0.3906	0.1717	0.1717
$\mathcal{A}$	0.0653	0.0674	0.0211	0.2105	0.0947	0.5411

## 5.2 Evaluation and Discussion

To measure how well our probabilistic assessment covers the correct rating, we adapt  $p@n$  metric in information retrieval as such

$$p@n = \frac{\sum_{i=1}^N tag_i^n}{N}, \quad (6)$$

where  $N$  is the total number of articles. Here  $tag_i^n$  is 1 if the top  $n$  rating of article  $i$  covers the correct quality class, otherwise it is zero.

Besides, to show the error of probabilistic quality rating we define a Distribution Difference Error ( $DDE$ ) measure as follows

$$DDE = \frac{1}{N} \sum_{i=1}^N |pe_i| \quad (7)$$

where  $N$  is the total number of articles and  $|pe_i|$  is the absolute difference between the probability of the first rating and that of the correct rating.

To objectively report the performance of AQRH approach, we used 10-fold cross validation method. Our reported result is the average of ten runs. We report our results in terms of effectiveness, impact of clustering on performance and comparisons with previous work in detail.

**Effectiveness of Probabilistic Assessment Approach.** Candidate revision cycle patterns are generated by setting  $\alpha$  from 4 to 100 and  $\beta$  from 4 to 20 for each quality class. After pruning, the pattern group with the largest uniqueness is chosen for each quality class. Table 3 summaries the pattern selection result for each quality class on computing dataset.

To refine quality class into quality corpora, the number of clusters, namely  $k$ , is tuned by trial and error test as follows. First, clustering is performed by

**Table 3.** Summary of Pattern Group Selection

Quality class	FA	GA	B	C	ST	SU
Uniqueness of selected group	5.3	4.9	3.7	2.95	3	2.53
Number of patterns in selected group	893	1021	367	392	104	28

setting  $k$  from 2 through  $k_{max} = \sqrt{N}$ , where  $N$  is the total number of articles of training set. Second, the validity index is computed for each clustering as follows [16],

$$V = \frac{intra}{inter} \tag{8}$$

where *intra* is defined as

$$intra = \frac{1}{N} \sum_{i=1}^k \sum_{F_j \in \Omega_i} |1 - sim(F_j, m_i)| \tag{9}$$

and *inter* is defined as

$$inter = \min(|1 - sim(m_i, m_j)|), i = 1, 2, \dots, k - 1, j = i + 1, \dots, k. \tag{10}$$

Here  $m_i$  is the medoid of corpus  $\Omega_i$ . The validity index represents overall average compactness against separation of the partition. Finally, the  $k$  producing the smallest  $V$  is chosen.

**Table 4.**  $k$ ,  $p@n$  and *DDE* Result on Computing Dataset

Quality Class	FA	GA	B	C	ST	SU
k	4	6	25	23	48	65
p@1	0.98	0.95	0.96	0.93	0.91	0.85
p@2	1	1	0.99	0.98	0.96	0.89
p@3	1	1	1	1	0.98	0.94
p@4	1	1	1	1	1	0.97
p@{5-7}	1	1	1	1	1	1
DDE	0.014	0.019	0.025	0.04	0.03	0.037

Table 4 reports the  $k$  and  $p@n$  results for each quality class. From  $p@1$  result we note that our AQRH gives the better performance for FA, GA and B classes than that for other classes. Through analysis, we found that FA, GA and B quality class shows large number of dominant gapped revision cycle patterns which can effectively distinguish them from others. In contrast, it performs worse on ST and SU class, especially on SU class. This is mainly due to that the revision history of SU class is usually very short and it is difficult to extract the most effective patterns. Note that FA and GA’s patterns overlap each other, but our approach performs better on FA than on GA. This is mainly due to that the pattern overlap between GA and B is larger than that between FA and B.

Based on all  $p@n$  measure, we can observe that our approach can get a very accurate rating when  $n$  increase from 1 to 2. Although the  $p@n$  is not very accurate when  $n = 1$  due to the inherent ambiguities of revision cycle patterns, the  $p@n$  measure can get a very good rating when  $n = 2$ . Specifically, our approach can give all the correct rating for FA, GA classes and almost all correct rating for other classes. When  $n$  increase from 3 through 7, the  $p@n$  measure almost increase by zero except the ST and SU class. By reading  $DDE$  measure of the last row in table 4, we found it very small for all quality classes. In other words, even sometimes our approach may give a false rating, the probability for false rating and that for correct rating is very close. This confirm that our probabilistic assessment is preferable.

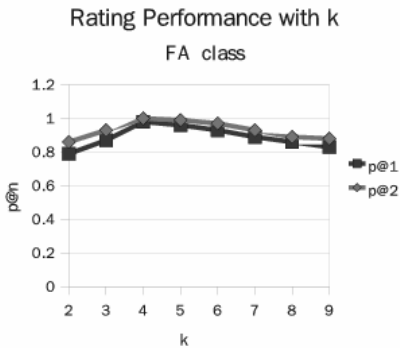


Fig. 1. Performance with k (FA class)

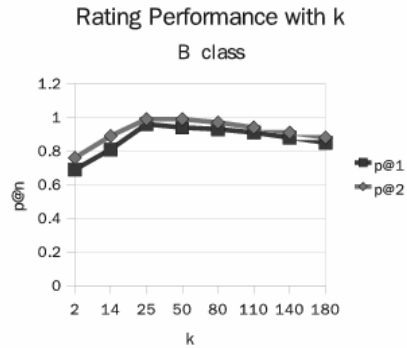


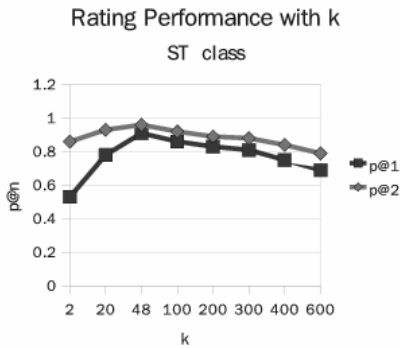
Fig. 2. Performance with k (B class)

**Impact of Clustering on Rating Performance** We observed that quality rating is sensitive to how many corpora are generated for each quality class. We only report the result of FA, B and ST classes in figure 1, 2 and 3 due to space. When the number of corpora, namely  $k$ , increases from 2 to  $\frac{N}{4}$  for one quality class, the  $k_{best}$  for other quality classes are fixed. The result shows that the  $p@n$  measure first increases then decreases when  $k$  increase from 2 to  $\frac{N}{4}$ . In other words, the rating accuracy first increases then decreases with the increase of the number of corpora per quality class.

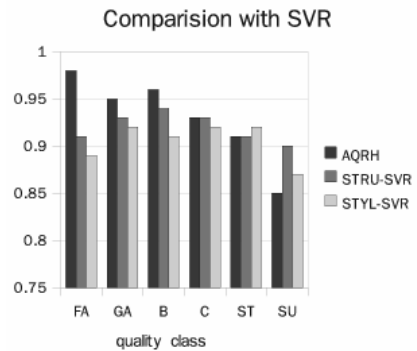
**Comparisons with Previous Works.** In this part we compare our method with the state-of-the-art work, namely SVR, which was proposed in literature 2. As the original implementation of the algorithms are not publicly available, we implemented it ourselves.

The SVR approach makes use of groups of features to train the model for quality rating. To be fair, we implement SVR approach using the group of features that were reported to perform best, namely structure feature group and style feature group. We use STRU-SVR and STYL-SVR to refer to them respectively. Figure 4 shows the performance comparison result between AQRH

and STRU-SVR, STYL-SVR. From the figure, we can observe that our method largely outperformed STYL-SVR and fairly outperformed STRU-SVR.



**Fig. 3.** Performance with k (ST class)



**Fig. 4.** Rating Performance Comparisons

## 6 Conclusion

In this paper we propose using revision history to probabilistically assess article quality. Specifically, we use frequent items mining to extract the revision cycle patterns. Then each quality class is characterized by a series of quality corpora, each of which represents a group of the revision cycle patterns which often co-occur. Finally, an article's quality is determined by comparing its state sequence with the quality corpora. Besides, a concrete probabilistic quality measure is given and thus quality can be objectively described in probabilistic sense.

As our future work, we plan to improve rating accuracy for some quality classes and to assess article quality in a more fine granularity.

## References

1. Dasu, T., Johnson, T., Muthukrishnan, S., Shkapenyuk, V.: Mining database structure; or, how to build a data quality browser. In: Proc. of SIGMOD 2002, pp. 240–251 (2002)
2. Dalip, D.H., Cristo, M., Calado, P.: Automatic quality assessment of content created collaboratively by web communities: A case study of wikipedia. In: Proc. of JCDL 2009, pp. 295–304 (2009)
3. Aebi, D., Perrochon, L.: Towards improving data quality. In: Proc. of the International Conference on Information Systems and Management of Data, pp. 273–281 (1993)
4. Wang, R.Y., Kon, H.B., Madnick, S.E.: Data quality requirements analysis and modeling. In: Proc. of the Ninth International Conference on Data Engineering, pp. 670–677 (1993)

5. Bouzeghoub, M., Peralta, V.: A framework for analysis of data freshness. In: Proc. of 2004 International Information Quality Conference on Information System, pp. 59–67 (2004)
6. Pernici, B., Scannapieco, M.: Data quality in web information systems. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 397–413. Springer, Heidelberg (2002)
7. Macdonald, N., Frase, L., Gingrich, P., Keenan, S.: The writer's workbench: computer aids for text analysis. *IEEE Transactions on Communications* 30(1), 105–110 (1982)
8. Foltz, P.W.: Supporting content-based feedback in on-line writing evaluation with *lsa*. *Interactive Learning Environments* 8(2), 111–127 (2000)
9. Rassbach, L., Pincok, T., Mingus, B.: Exploring the feasibility of automatically rating online article quality (2008)
10. Stvilia, B., Twidle, B., Smith, M.C.: Assessing information quality of a community-based encyclopedia. In: Proc. of the International Conference on Information Quality, pp. 442–454 (2005)
11. Zeng, H., Alhossaini, M.A., Ding, L.: Computing trust from revision history. In: Proc. of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services (2006)
12. Zeng, H., Alhossaini, M.A., Fikes, R., McGuinness, D.L: mining revision history to assess trustworthiness of article fragments. In: Proc. of International conference on Collaborative Computing: Networking, Applications and Worksharing, pp. 1–10 (2009)
13. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of IEEE*, 257–286 (1989)
14. Baum, L.E., Petrie, T., Soules, G., Weiss, N.: A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Statist.* 41(1), 164–171 (1970)
15. Ding, B., Lo, D., Han, J., Khoo, S.C.: Efficient mining of closed repetitive gapped subsequences from a sequence database. In: Proc. of 2009 ICDE, pp. 1024–1035 (2009)
16. Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13(8), 841–847 (1991)

# Propagation of Multi-granularity Annotations

Ryo Aoto, Toshiyuki Shimizu, and Masatoshi Yoshikawa

Kyoto University, Kyoto, Japan 606-8501

{aoto@db.soc., tshimizu@, yoshikawa@}i.kyoto-u.ac.jp

**Abstract.** Data origin or processing information and the metadata that is useful in understanding data can be associated with data by using annotation. Provenance knowledge preserved by annotation is managed by continuously propagating the annotations through the workflow. Models for explicitly associating annotations are generally used for annotation-based provenance management, and techniques for propagating annotations have been proposed. There is also a model for implicitly associating annotations – the annotations are associated with data with arbitrary granularity by using queries. We call the implicit model “multi-granularity annotation” model. Multi-granularity annotation enables flexible association of information. However, no provenance management methods using multi-granularity annotations have been reported. We have developed a method for propagating multi-granularity annotations. We define rules for annotation propagation for each relational algebra operation, and they are used to recalculate the scopes of annotations associated with data. We also addressed the loss of information needed to preserve annotation associations during data derivation and the lack of static data annotations by extending the operations and the association method. Experiments showed that our method requires less space usage and execution time than conventional annotation management methods.

**Keywords:** multi-granularity annotation, annotation propagation, provenance management.

## 1 Introduction

The importance of provenance management has been increasing [5, 8]. Provenance means *origin*, *source* or *lineage* and is typically used to represent information about the creation of scientific artifacts. Data provenance includes the processing history of data and gives information about the data used for a data derivation or the processing steps themselves. This information is indispensable for determining the quality of scientific results. Data provenance is generally regarded to be as important as the results themselves [13] because it ensures the reproducibility of data derivations and guarantees the reliability of data. Moreover, the dependence of the results on the derivation processes can be extracted by analysis of the provenance knowledge.

Many studies on provenance management have been conducted in the database field. Static workflow analysis [6] or annotations are typically used for provenance

management. Workflow analysis studies have generally addressed the derivation of data origin. The source data for a result are traced by analyzing the dataset and queries used for the data derivation [7,3,11]. In studies on annotation, various types of information useful for managing and understanding the data can be associated with data as metadata: origin, derivation, quality, deficit, and comment. By propagating this information to the results through queries, we can preserve the information needed for provenance management. In this paper, we introduce a technique using such propagation for annotation-based provenance management.

The word annotation generally represents the documents associated with a subset of data. Annotations are used for associating metadata with records in a database. There have been many studies on the management of annotations, and various techniques for the association of annotations have been proposed. In the typical methods, annotations are associated with data by a cell [4], a subset of a record [10], or a block of records [12]. Annotations are explicitly associated with a record in the methods proposed in literature [4,10]. In these explicit methods, the propagations are easily calculated since the annotations are directly associated with records. However, the records to which an annotation refers must be individually specified when the annotation is created, and the annotation must be re-associated whenever new records to which they refer are inserted. In the multi-granularity annotation method proposed by Srivastava et al. [12], annotations are implicitly associated with records that satisfy a specific property by using conditional expressions. The annotations can be flexibly associated with data across multiple records, whereas conventional methods associate annotations with a cell or a subset of a record.

As an example of multi-granularity annotations, consider the association with the meteorological data shown in Figure. 1. The data table has four attributes (observation point id, observation date, temperature, and humidity) and four annotations are associated with it. Annotation  $A_P$  is associated with the value 345001 in the POINTID column and it preserves information about the observation point (e.g., place name, longitude, or latitude). Annotation  $A_C$  is a comment associated with a value in the TEMP column. Annotation  $A_D$  is a comment associated with a pair of values in the TEMP and HUMID column whose values of temperature-humidity index are greater than 75. Annotation  $A_E$  represents the missing value due to an equipment error and is associated with the values of the HUMID column at specific date. These annotations are associated with specific records that satisfy the following conditions.

- $A_P$  :  $ID = 345001$
- $A_C$  :  $(TEMP < 25) \wedge (YEARMODA \text{ like } ' \_ \_ \_ -08 - \_ ')$
- $A_D$  :  $DC(TEMP, HUMID) > 75$
- $A_E$  :  $(ID = 345001) \wedge (YEARMODA = 2010-08-01)$

As this example shows, a method that exploits conditional expressions to specify the associations has two advantages. First, the annotations are stored at lower storage cost than with the explicit methods in which contents of annotations overlap when the annotations are associated with multiple records. The space

**TempHumid345001**

POINTID	YEARMODA	TEMP	HUMID
345001	2010-08-01	30.2	10
345001	2010-08-02	23.4	50
345001	2010-08-03	30.7	74
...	...	...	...

Annotations and their associations:

- AE: 'Equipment error' (points to HUMID)
- AP: Place Name, Longitude, Latitude (points to POINTID)
- AC: 'Cool' (points to TEMP)
- AD: 'Discomfort' (points to HUMID)

**Fig. 1.** Example associations of annotations with meteorological data

**OUT**

POINTID	YEARMODA	HUMID
345001	2010-08-01	10
345001	2010-08-03	74
...	...	...

Annotations and their associations:

- AE: 'Equipment error' (points to HUMID)
- AD: 'Discomfort' (points to HUMID)
- AP: Place Name, Longitude, Latitude (points to POINTID)

**Fig. 2.** Example output

usage for storing annotations is larger if the annotations are associated with multiple records such as annotation  $A_P$  in this example. In contrast, annotations are associated with a block of records by the using conditional expressions, and its contents are stored only once in multiple-granularity annotations. This approach reduces the required storage space. Second, annotations are automatically associated with the records that satisfy the specified condition. For example, if records added to the dataset for another point satisfy the condition for  $A_C$ , they will be automatically associated with  $A_C$ .

An important problem with using annotations is that they are lost during data derivation, so the results generated lack the annotations. It is desirable that the information preserved by annotations can be extracted from the results. While the information associated with the input data could be extracted by tracing the source of the results, tracing every data source would be impractical. Several *annotation propagation* techniques that address this problem have been proposed. The annotations associated with the source data are propagated to the results during data derivation. The metadata associated with the source data enable to be extracted at an arbitrary point of the workflow by annotation propagation. While techniques for propagating annotations have been proposed for explicit association [4,10], none have been reported for implicit association. Only an association technique has been proposed [12].

We have developed a method for propagating multi-granularity annotations. As an example of such propagation, consider the extraction of the humidity value from a record in which the temperature value is greater than 30. The output of a query usually contains only the selected data: the annotations are lost in the extraction. With multi-granularity annotation, the information associated with the input is propagated by recalculating the additional annotation conditions and re-associating the annotations with the result of the query, as shown in Figure. 2

In our development of this method, we considered static association in which annotations remain associated with the records to which they were initially associated in addition to dynamic association, because in some cases, annotations are associated with unexpected records as the result of a union operation since datasets with different contexts are combined. Moreover, we addressed the loss of information needed to preserve annotation associations during data derivation



by extending the algebraic operations used and the association method. In the developed method, the additional scope of the annotations to be propagated are automatically recalculated.

The composition of this paper is as follows. In Section 2, we briefly discuss related works. In Section 3, we describe our method for propagating multi-granularity annotations. In Section 4, we evaluate our method in terms of storage space and execution time. In Section 5, we summarize the key points and mention future work.

## 2 Related Work

As mentioned above, annotations are lost during data derivation. If the location of annotations were recalculated after derivation and associated with the output, they would be carried forward to the query results. Buneman et al. [4] were the first to propose technique for propagating annotations in a relational database. In their propagation model, annotations are associated with the records for a cell as shown in Figure. 3, and rules for propagating annotations are defined for each relational algebraic operation. Propagation is calculated on the basis of the location from which each value in the cell was copied (“where provenance” [3]). Bhagwat et al. [2] developed an annotation management system *DBNotes* based on the model of Buneman et al. [4]. They defined a query language pSQL, which is extension of SQL for the annotation propagation, and showed the algorithm that absorbs the differences among all equivalent formulations of a given query. Gerrts et al. [10] extended the location-based model so that annotations can be associated with an arbitrary set of attributes in a record as shown in Figure. 4. They defined *color algebra* for manipulating the annotations, which are represented as color blocks, and determined the meanings of annotations by categorizing them on the basis of whether they have a meaning as a set or not. Srivastava et al. [12] generalized the conventional annotation models into one in which annotations can be associated with arbitrary set of attributes and records as shown in Figure. 5. A query is used for representing the association of an annotation. The select-and-where clause of a query specifies the scope of the data with which an annotation will be associated. Srivastava experimentally showed that his method requires less storage space and has lower update cost for associating annotations than conventional annotation management systems [4,10].

As summarized in Table 1, several methods for propagating annotations explicitly associated with data have been reported but not for those associated using arbitrary granularity. We consider that, by applying the propagation discussed above to the multi-granularity annotation model, we can manage data provenance information more easily and at lower cost. Eltabakh et al [9] developed the method for storing annotations in relations efficiently and propagating them through queries. On the other hand, we developed the query-based propagation method by extending algebras for preserving the semantics of associations.

**Table 1.** Relationship with related work

	[42]	[10]	[12]	Our method
Granularity	for a cell of a record	for a subset of of a record	for a subset of multiple records	for a subset of multiple records
Propagation considered	YES	YES	NO	YES

	A	B	C
t <sub>1</sub>	1 a	2 a	3
t <sub>2</sub>	4 a	5 b	6 b

**Fig. 3.** Representation of annotations described by Buneman et al. [42]

	A	B	C
t <sub>1</sub>	1	2	3
t <sub>2</sub>	4	5	6

**Fig. 4.** Representation of annotations described by Geerts et al. [10]

	A	B	C
t <sub>1</sub>	1	2	3
t <sub>2</sub>	4	5	6

**Fig. 5.** Representation of annotations described by Srivastava et al. [12]

### 3 Propagation of Multi-granularity Annotations

In this section, we describe our representation of multi-granularity annotations and our method for propagating them.

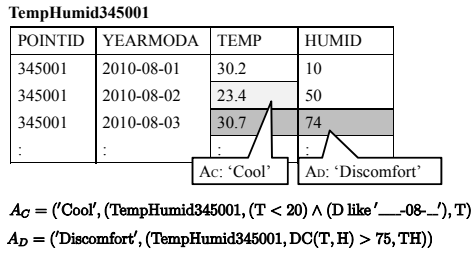
#### 3.1 Annotation Model

An annotation associated with an arbitrary portion of data is represented by both of the contents of annotation and an additional condition. The additional condition specifies the portion of the data with which the annotation is associated. It is represented by a triple: table name, selection conditional expression, and attributes list. The selection conditional expression is a logical formula specifying the scope of the records to which the annotations refers. The attributes list is a set of attributes with which the annotation is associated. By determining the table, expression and attributes, an annotation can be associated with data with arbitrary granularity. Annotation A is given by

$$A = (a, l) = (a, (R, T_c, \mathcal{A}))$$

where "a" represents the contents of the annotation and "l" represents the additional condition. As mentioned above, the l is represented by the triple- R, T, A (table name, selection expressional expression, attributes list). This annotation model is equal to the model formulated by Srivastava [12] at the concept level.

As an example, we reconsider the TempHumid345001 table and annotations  $A_C, A_D$  associated to the table in Figure. 6. We use the abbreviations I, D, T, and H for representing attributes POINTID, YEARMODA, TEMP, and HUMID.



**Fig. 6.** Example multi-granularity annotations

Annotations  $A_C$  and  $A_D$  are represented as

$$A_C = ('Cool', (\text{TempHumid345001}, (T < 20) \wedge (D \text{ like '---08----'}), T))$$

$$A_D = ('Discomfort', (\text{TempHumid345001}, DC(T, H) > 75, TH))$$

With this model, annotations can be flexibly associated with data on the basis of defined conditions. Moreover, annotations can be associated dynamically. If a record satisfying a condition is inserted, it is automatically associated with the annotation. That is, no maintenance for associating existing annotations with data in new records is needed. However, one may want to associate annotations with particular records that have been in the dataset since the annotations were created. For example, automatically associating an annotation that has meaning only in the initial dataset with data in new records may be contrary to the intent of the annotation creator. This problem is particularly likely in the union of datasets. Therefore, we also considered the static association of annotations.

### 3.2 Propagation Model

Our proposed propagation model is based on the representation of multi-granularity annotations described in the preceding section.

To propagate multi-granularity annotations, we need to know to which portion of the data the propagated annotations should be re-associated. We thus defined propagation rules for recalculating the scope of multi-granularity annotations. These rules represent the transformation of the additional scope of an annotation at each propagation step. Annotations are propagated by recalculating the additional scope of each annotation in accordance with the rules. We define propagation rules for every basic operation of relational algebra: selection, projection, join, union, difference, and rename. The rules for each operation are represented such that the left side is an input annotation and the right side is a propagated annotation.

– **Projection:**  $\pi_B(R)$

$$(a, (R, T_c, \mathcal{A})) \rightarrow (a, (\pi_B(R), T_c, \mathcal{A} \cap B))$$

This expression alters the attributes list of additional conditions in accordance with the induced changes in the projection operation attributes. The

**Project**

POINTID	YEARMODA	TEMP	HUMID
345001	2010-08-01	30.2	10
345001	2010-08-02	23.4	50
345001	2010-08-03	30.7	74
:	:	:	:

$$A_C = ('Cool', (Project, (T < 20) \wedge (D \text{ like } '_{-}08-_{-}'), T))$$

$$A_D = ('Discomfort', (Project, DC(T, H) > 75, T))$$

**Fig. 7.** Propagation result following projection

**Select**

POINTID	YEARMODA	TEMP	HUMID
345001	2010-08-01	30.2	10
345001	2010-08-03	30.7	74
:	:	:	:

$$A_D = ('Discomfort', (Select, DC(T, H) > 75, TH))$$

**Fig. 8.** Propagation result following selection

modified attributes list is the product of attributes list A of the annotation and parameter B of the projection. If an attribute included in the selection conditional expression is deleted by this modification, the information for associating annotations is lost. The solution is to preserve the attributes constituting the conditional expressions of the annotations in addition to the attributes projected by the projection operator. The preserved attributes are treated as a system-valid column, invisible to users.

As an example, consider the projection operation projecting POINTID, YEARMODA, and TEMP from the TempHumid345001 table in Figure. 6. In conventional projection, only the attributes assigned by the parameter are projected. Here, the HUMID attribute is also projected to preserve the values used for the association of the annotations. The results are shown in Figure. 7.

– **Selection:**  $\sigma_c(R)$

$$(a, (R, T_c, \mathcal{A})) \rightarrow (a, (\sigma_c(R), T_c, \mathcal{A}))$$

The selection conditional expression  $T_c$  of the input is applied directly to the output of the operation. Annotations are propagated without altering the selection conditional expressions. That is because, any tuple that does not satisfy the select-predicates does not appear in the select operation result and any tuple is newly inserted.

As an example, consider a selection operation that selects the TEMP data with a value greater than 30 from the table in Figure. 6. The tuples with a YEARMODA value of 2010-08-02 or 2010-08-03 are the target tuples with which annotations are associated. The results are shown in Figure. 8.

– **Join:**  $R_1 \bowtie R_2$

$$(a_1, (R_1, T_{c_1}, \mathcal{A}_1)) \rightarrow (a_1, (R_1 \bowtie R_2, T_{c_1}, \mathcal{A}_1))$$

$$(a_2, (R_2, T_{c_2}, \mathcal{A}_2)) \rightarrow (a_2, (R_1 \bowtie R_2, T_{c_2}, \mathcal{A}_2))$$

Annotations are propagated without any modification of the selection conditional expressions. This is because any tuple that does not satisfy the join-predicates does not appear in the join operation result.

**PRESS345001**

POINTID	YEARMODA	PRESS
345001	2010-08-01	1001
345001	2010-08-02	1007
345001	2010-08-03	9999
:	:	AE: 'Equipment error'

$A_E = ('Equipment\ error', (PRESS345001, (D = 2010-08-03) \wedge (I = 345001), P))$

**Fig. 9.** Input table for join

**Join**

POINTID	YEARMODA	TEMP	HUMID	PRESS
345001	2010-08-01	30.2	10	1001
345001	2010-08-02	23.4	50	1007
345001	2010-08-03	30.7	74	9999
:	:	:	:	AE

$A_C = ('Cool', (Join, (T < 20) \wedge (D\ like\ '---08-_', T)))$

$A_D = ('Discomfort', (Join, DC(T, H) > 75, TH))$

$A_E = ('Equipment\ error', (Join, (D = 2010-08-03) \wedge (I = 345001), P))$

**Fig. 10.** Propagation result following join

As an example, consider the join operation for the table in Figure. 6 and the PRESS345001 table in Figure. 9, which has the same values for POINTID and YEARMODA plus PRESS column. A data value in the PRESS345001 table is associated with an annotation for "equipment error". The selection condition is passed to the output directly following the joining of these tables, leading to the propagation of annotations. The results are shown in Figure. 10

– **Union:**  $R_1 \cup R_2$

$$\begin{aligned}
 (a_1, (R_1, T_{c_1}, \mathcal{A})) &\rightarrow (a_1, (R_1 \cup R_2, T_{c_1}, \mathcal{A})) \\
 (a_2, (R_2, T_{c_2}, \mathcal{A})) &\rightarrow (a_2, (R_1 \cup R_2, T_{c_2}, \mathcal{A}))
 \end{aligned}$$

Annotations can be propagated without any modification of an additional condition if they are dynamically associated with data. Annotations with equivalent attribute lists and contents can be integrated as follows.

$$\begin{aligned}
 (a, (R_1, T_{c_1}, \mathcal{A})) \\
 (a, (R_2, T_{c_2}, \mathcal{A})) &\rightarrow (a, (R_1 \cup R_2, T_{c_1} \vee T_{c_2}, \mathcal{A}))
 \end{aligned}$$

If records that newly appear in the output following the union satisfy the selection conditional expression of an annotation, they are automatically associated with the annotation. This dynamic association is useful, because the annotations are easily maintained by automatic association. However, users should be able to determine whether new records should be associated with existing annotations. We thus need to also consider the use of static association in which annotations are associated with only the records that existed when the annotations were created.

For static association to be used the records that were initially associated with an annotation need to be distinguishable at later time. To enable this, we modify the data table and annotation table when the annotations are created. A system-valid attribute (E) that is invisible to users is added to the data table, and the name of the data table is stored to it when generating an annotation. In addition, a predicate is added to the selection conditional expression that defines the value of E. For example, if annotation  $(a, (R, T_c, \mathcal{A}))$

TempHumid486002

POINTID	YEARMODA	TEMP	HUMID
486002	2010-08-01	22.1	60
486002	2010-08-02	20.3	68
486002	2010-08-03	19.2	0
:	:	:	:

$$A_E = (\text{'Equipment error', (TempHumid486002, (D = 2010-08-03) \wedge (I = 486002), H)})$$

Fig. 11. Input table for union

Union

POINTID	YEARMODA	TEMP	HUMID
345001	2010-08-01	30.2	10
345001	2010-08-02	23.4	50
345001	2010-08-03	30.7	74
:	:	:	:
486002	2010-08-01	22.1	60
486002	2010-08-02	20.3	68
486002	2010-08-03	19.2	0
:	:	:	:

$$A_C = (\text{'Cool', (Union, (T < 20) \wedge (D like '___-08-'), T)})$$

$$A_D = (\text{'Discomfort', (Union, DC(T, H) > 75, TH)})$$

$$A_E = (\text{'Equipment error', (Union, (D = 2010-08-03) \wedge (I = 486002), H)})$$

Fig. 12. Propagation result following union

is generated for data table R, the selection conditional expression for it is rewritten as  $(a, (R, T_c \wedge (E = ' R'), \mathcal{A}))$ . The records initially associated with an annotation can then be selected by modifying the annotation table and the data table when generating the annotation and examining whether the record satisfies the added condition when referring to the annotation. Note that static association is needed when new records are inserted by union operation. However, annotation tables and data tables need to be modified beforehand by adding the predicates indicating when the annotations were created.

As an example, consider the union operation for the table in Figure. 6 and the table in Figure. 11, which contains values observed for an other point. If the annotations are associated dynamically, the result is as shown in Figure. 12. The annotations are propagated to the output, and the new records that satisfy the additional condition of the annotations are also associated with them. The new records with TEMP values satisfying the additional condition of the annotations  $A_C$  associated with the value in the TempHumid345001 table are also associated with the annotation. However, if annotation  $A_C$  is a comment that has meaning only for a specific point, the dynamic association may be contrary to the creator's intent. To associate annotation  $A_C$  with the records in the TempHumid345001 table that were initially associated to it, we preserve the name of the data table and add a predicate to the selection conditional expression, as shown in Figure 13. The new predicate is examined to determine whether the records were initially in the TempHumid345001 when the annotation is referred.

- **Rename** :  $\delta_\theta(R)$

$$(a, (R, T_c, \mathcal{A})) \rightarrow (a, (\delta_\theta(R), \theta(T_c), \theta(\mathcal{A})))$$

We alter the name of the attributes in the selection conditional expressions and attribute lists in accordance with the parameters of the rename operation.

**Union**

POINTID	YEARMODA	TEMP	HUMID	E
345001	2010-08-01	30.2	10	345001
345001	2010-08-02	23.4	50	345001
345001	2010-08-03	30.7	74	345001
:	:	:	:	:
486002	2010-08-01	22.1	60	486002
486002	2010-08-02	20.3	68	486002
486002	2010-08-03	19.2	0	486002
:	:	:	:	:

$A_C = ('Cool', (Union, (T < 20) \wedge (D \text{ like } ' \_ \_ \_ 08 \_ \_ ') \wedge (E = 345001))$   
 $A_D = ('Discomfort', (Union, (DC(T, H) > 75) \wedge (E = 345001), TH))$   
 $A_E = ('Equipment error', (Union, (D = 2010-08-03) \wedge (I = 486002) \wedge (E = 486002), H))$

**Fig. 13.** Static association of annotations

**Rename**

POINTID	YEARMODA	MEAN_TEMP	HUMID
345001	2010-08-01	30.2	10
345001	2010-08-02	23.4	50
345001	2010-08-03	30.7	74
:	:	:	:

$A_C = ('Cool', (Rename, (T_M < 20) \wedge (D \text{ like } ' \_ \_ \_ 08 \_ \_ '), T_M))$   
 $A_D = ('Discomfort', (Rename, DC(T_M, H) > 75, T_M H))$

**Fig. 14.** Propagation result following rename

POINTID	YEARMODA	TEMP	HUMID
345001	2010-08-01	30.2	10
345001	2010-08-03	30.7	74

**Fig. 15.** Input table for difference

**Difference**

POINTID	YEARMODA	TEMP	HUMID
345001	2010-08-03	30.7	74
:	:	:	:

$A_D = ('Discomfort', (Difference, DC(T, H) > 75, TH))$

**Fig. 16.** Propagation result following difference

As an example, consider an operation that renames TEMP in Figure. 6 to MEAN\_TEMP . Following this change to the column name, the corresponding attribute name of the selection conditional expressions and the attribute lists for the additional conditions are modified. As a result, the annotations are propagated to the output as shown in Figure. 14

– **Difference** :  $R_1 - R_2$

$$(a, (R_1, T_{c_1}, \mathcal{A})) \rightarrow (a, (R_1 - R_2, T_{c_1}, \mathcal{A}))$$

Without any modification of the selection conditional expressions and by directly applying it to the output, we can propagate annotations. This is because the tuples deleted by the difference operation do not appear in the query execution result.

As an example, consider the difference operation for the table in Figure. 6 and the table in Figure. 15. By applying the selection conditional expressions of the input annotations to the tuple remaining after the difference operation, we can propagate the annotations as shown in Figure 16.

The combination of modified algebra can also construct more complicated queries represented by relational algebra (e.g., outer-join, anti-join, etc.).

## 4 Experimental Evaluation

We evaluated our method for propagating multi-granularity annotations by comparing its performance with that of two conventional annotation management systems [2,10]. We used space usage and query execution time with annotation propagation as the metrics.

For our evaluation, we implemented a prototype of our annotation management system on the MySQL relational DBMS, running on a windows-based PC (Windows 7 OS, 64-bit IntelCPU, Core i7 architecture, 1.2-GHz clock, 4-GB RAM). Our system uses two auxiliary tables to store the annotations in a relational database for each data table as shown in Figure. 17: *annotation table* and *attribute list table*. The annotation table contains the additional conditions and the annotation contents, and the attribute list table contains the correspondences between the attribute names in the data table and the values of the attribute ID columns of the annotation table. The additional condition for an annotation is determined by the values in the attribute id and selection condition columns in the annotation table. The attribute ID columns are Boolean attributes: the value is true if the annotation is associated with the corresponding attribute in the data table; otherwise it is false. The selection condition column contains the attributes used to keep the values that specify the records associated with the annotations. Their values are represented as logical expressions as shown in Figure. 17. To deal with modifications to attribute names in rename and projection operations, we present the selection conditional expressions as attribute IDs instead of the attribute names in the attribute list table. Modifications of attribute names in a data table are applied to the values of the attribute ID columns in the attribute list table. By creating these auxiliary tables, we can associate the annotations with the data with arbitrary granularity. The propagations are calculated using the propagation rules defined in the previous section. The annotations are propagated by executing a query against the data tables and applying the rewritten query to the annotation tables and attribute list tables. For example, if a query contained a projection or rename operation, the attribute list table would be modified. Similarly, attribute list tables are combined for propagation following a union operation. As shown in Figure. 17, our system represents static associations of annotations by using tables that stores the names of the table with which each annotations are initially associated. When annotations are referred, the data tables and the tables storing table names are joined. Some annotations may become invalid (i.e., they are no longer associated with a record) following a selection, join, or difference operation. These invalid annotations can be detected later by data validation and deleted.

The two annotation management systems used for comparison DBNotes [2] and MONDRIAN [10], were recently proposed. We exclude the system proposed in [12] from comparison because our system stores annotations in the same way as [12] and it does not offer the mechanism to propagate annotations. DBNotes stores annotations in an extra column added to each attribute column in the data table, as shown in Figure. 18(a). If more than one annotation is associated with a record, the record is replicated and the other annotations are stored in



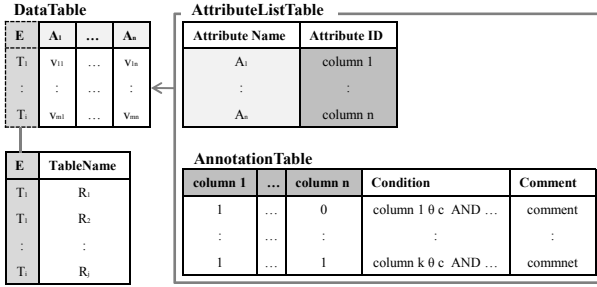


Fig. 17. Data structure for annotations

**Data Table with Annotations**

A	A'	B	B'
1	a	2	a
3	b	4	b
3	c	4	-

extra columns

(a) DBNotes

**Data Table**

A	B	ID
1	2	1
3	4	2

**Annotation Table**

ID	A'	B'	Comment
1	1	1	a
2	1	1	b
2	1	0	c

(b) MONDRIAN

Fig. 18. Data structure for conventional systems

the replicant. Since multiple annotations can be associated with the same record, there may be record duplication. In Figure. 18(a), annotations 'a', 'b', and 'c' are associated with  $(A, B) = (1, 2), (3, 4)$ . Annotation 'a' is associated to  $(1, 2)$ , and 'b' and 'c' are associated with  $(3, 4)$ . The tuple  $(3, 4)$  is multiple recorded because more than one annotation is associated with it, as shown in Figure. 18(a). In DBNotes, annotations are explicitly associated with records, so they are automatically propagated along with the query result. In MONDRIAN, annotations are associated with every record and, as shown in Figure. 18(b), stored in an annotation table dissociated from the data tables. The same annotations in the table in Figure. 18(a) are associated with the table in Figure. 18(b). The annotation tables have Boolean columns corresponding to the attributes in the data tables, and the columns store the annotation contents. If an annotation is associated with an attribute of a record in a data table, the value of the corresponding Boolean column is true, otherwise it is false. The difference between MONDRIAN and DBNotes is that MONDRIAN can associate an annotation with an arbitrary set of record attributes. In the original MONDRIAN system, annotations are associated with records by storing the clones of data tables in annotation tables. However, for fair comparison, we preserved the correspondence between annotations and records by using IDs in our implementation of MONDRIAN, as shown in Figure 18(b). MONDRIAN executes queries joining data tables and annotation tables. Since it explicitly associates annotations with records, annotations are automatically propagated to query result.

As a dataset for our evaluation we used the 10-GB TPC-H database [1], and we used the customer table and the supplier table of it. We created 5000 couple of an additional condition and a sentence of 50 random characters as annotations associated with the dataset. We associated a portion of these to 5% of the records are associated with some annotations. Similarly, we associated the annotations to 25, 50, 75, and 90% of the dataset. Since the queries generated by TPC-H contain aggregations, and our system is unable to handle the aggregations, we executed our own queries: **Select query**; **SP suery**; **SPJ query**.

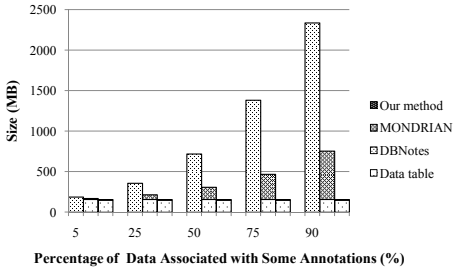


Fig. 19. Space for storing annotations associated with original dataset

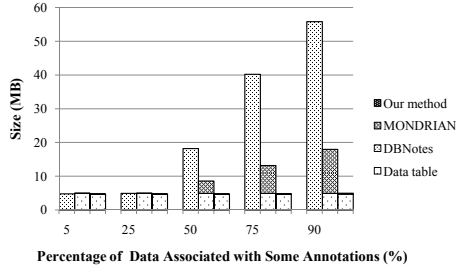


Fig. 20. Space for storing annotations associated with results of Select query

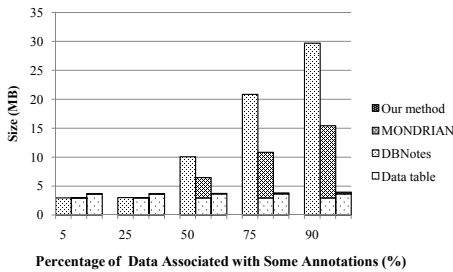


Fig. 21. Space for storing annotations associated with results of SP query

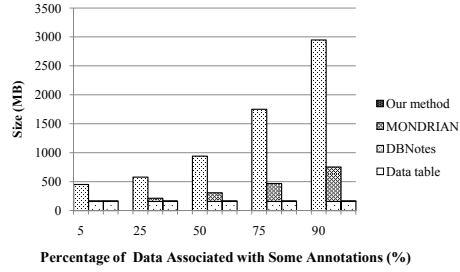


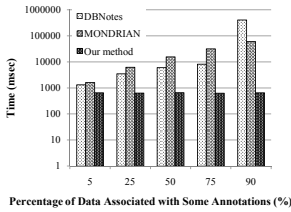
Fig. 22. Space for storing annotations associated with results of SPJ query

### 4.1 Storage Space

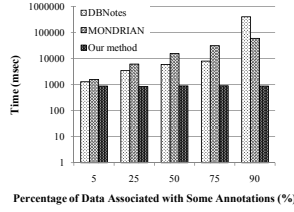
For a fair comparison of our system with DBNotes in terms of storage space, we measured the space used by DBNotes with the size of the data table in which annotations are stored. For measuring MONDRIAN and our system, we used the total size of data tables and annotation tables.

The space required for associating the annotations with the original data table is shown in Figure. 19. That associated with the query results shown in Figure. 20, Figure. 21 and Figure. 22. The data table size in Figures. 19 to 22 is the total size of the data tables: and the space required by MONDRIAN and our system is the total size of the data tables and the annotation tables. As our method associates annotations as modeled by Srivastava et al. [12], the superiority of our method evident in Figure. 19 has been previously shown [12].

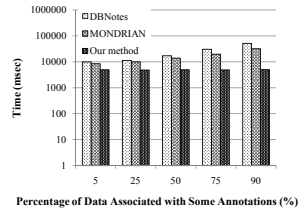
The results show that our method stored annotations in less space than the two other methods in almost all cases. This was due to the differences in storage mechanisms. In DBNotes, annotations are associated with every cell in the target records, so annotation content is duplicated for records and attributes. Similarly, in MONDRIAN, annotation content is stored for associated records. This means that the more records with which annotation is associated, the greater the



**Fig. 23.** Propagation time of select query



**Fig. 24.** Propagation time of SP query



**Fig. 25.** Propagation time of SPJ query

amount of duplicate content storage, which means increased storage space. Since our system stores annotation content only once in a separate table, it requires the least storage space.

In the case the percentage of data associated with annotations is 5 or 25%, the space required by our method is larger than that of the other two methods, as shown in Figure 21. This is attributed to the peculiarity of our propagation rule: the attributes included in the selection conditional expressions are additionally projected to the result, which is not the case with the other two methods. The storage space needed for output derived by queries including projection tended to be larger than with the other two systems. The original SP query propagated four of the seven attributes while our system rewrote the query to preserve the values of the attributes included in the selection conditional expressions, resulting in the propagation of five attributes. As a result, the data tables derived by the query increased in size. Therefore, the storage space required with our method was sometimes larger than with the other two methods.

## 4.2 Query Execution Time

For our comparison of query execution times, we measured the query execution time of DBNotes as the time for querying the data table with which the annotations are associated. For MONDRIAN and our system, we used the total time for querying a data table and transforming the annotations.

As shown in Figures 23, 24, and 25, the times with our system were the shortest in all cases. The execution time for DBNotes and MONDRIAN varied with the amount of annotations while those for our system were virtually constant. This is attributed to the differences in the data structure and the annotation propagation method. In the other two systems, the greater the number of annotations associated with records, the greater the amount of stored contents due to content duplication. On the other hand, in our method, the amount of content stored for an annotation is independent of the number of records associated with that annotation, so propagation basically consists of annotation table replication. Therefore, there was no significant increase in the execution time of our system.

## 5 Conclusions

The method we have developed for annotation management propagates annotations associated with multi-granularity data to the query output. We defined propagation rules for calculating the transformation of the additional scope of an input annotation for every operation of relational algebra. Our experiments showed that our method is better than two typical annotation management systems in terms of storage space and propagation execution time. Two main conclusions were obtained from this work.

- An annotation associated with data with arbitrary granularity can be propagated through a query by using the defined propagation rules.
- The space required for storing annotations and the propagation execution time can be reduced by using multi-granularity annotations.

We addressed propagation only for the basic operations of relational algebra. We plan to address it for aggregation operations or updates as well. Furthermore, we plan to develop a propagation method considering the semantics of annotations.

## References

1. Tpc-h benchmark, <http://www.tpc.org/tpch/>
2. Bhagwat, D., Chiticariu, L., Tan, W.C., Vijayvardiya, G.: An annotation management system for relational databases. In: VLDB. pp. 900–911 (2004)
3. Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: ICDT. pp. 316–330 (2001)
4. Buneman, P., Khanna, S., Tan, W.C.: On propagation of deletions and annotations through views. In: PODS. pp. 150–158 (2002)
5. Buneman, P., Tan, W.C.: Provenance in databases. In: SIGMOD. pp. 1171–1173 (2007)
6. Cheney, J., Chiticariu, L., Tan, W.C.: Provenance in databases: Why, how, and where. *Foundations and Trends in Databases* 1(4), 379–474 (2009)
7. Cui, Y., Widom, J., Wiener, J.L.: Tracing the lineage of view data in a warehousing environment. *ACM TODS* 25(2), 179–227 (2000)
8. Davidson, S.B., Boulakia, S.C., Eyal, A., Ludäscher, B., McPhillips, T.M., Bowers, S., Anand, M.K., Freire, J.: Provenance in scientific workflow systems. *IEEE Data Eng. Bull.* 30(4), 44–50 (2007)
9. Eltabakh, M.Y., Aref, W.G., Elmagarmid, A.K., Ouzzani, M., Silva, Y.N.: Supporting annotations on relations. In: EDBT. pp. 379–390 (2009)
10. Geerts, F., Kementsietsidis, A., Milano, D.: Mondrian: Annotating and querying databases through colors and blocks. In: ICDE. p. 82 (2006)
11. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: PODS. pp. 31–40 (2007)
12. Srivastava, D., Velegrakis, Y.: Intensional associations between data and metadata. In: SIGMOD. pp. 401–412 (2007)
13. Tan, W.C.: Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull.* 30(4), 3–12 (2007)

# Author Index

- Akbarinia, Reza I-140  
Amann, Bernd II-203  
Aoto, Ryo II-589  
Apers, Peter M.G. II-118  
Aravogliadis, Pantelis II-1  
Asano, Yasuhito I-341  
Ashrafi, Mafruz Zaman I-187  
Asiki, Athanasia II-527
- Badr, Mehdi I-379  
Balke, Wolf-Tilo II-350  
Banek, Marko II-439  
Barbosa, Denilson II-502  
Barhamgi, Mahmoud I-202  
Béchet, Nicolas II-154  
Beierle, Christoph I-27  
Bellahsene, Zohra II-396  
Ben Saad, Myriam I-394  
Bhattacharya, Arnab II-493  
Bi, Yaxin II-219  
Bisbal, Jesus II-59  
Böhm, Christian I-349  
Bouchou, Béatrice I-94  
Bouillot, Flavien II-154  
Bowers, Shawn I-526  
Bressan, Stéphane I-448  
Bringas, Pablo G. II-519  
Bringay, Sandra II-154  
Brut, Mihaela II-249  
Buche, Patrice I-511  
Busemann, Claas II-311
- Campos, Andre M. II-303  
Canuto, Anne M.P. II-303  
Cao, Huiping I-526  
Cao, Jinli I-425  
Cappellari, Paolo II-366, II-411  
Capra, Licia I-542  
Caragea, Doina I-217  
Ceci, Michelangelo II-97  
Che, Dunren II-420  
Chen, Baichen I-156  
Chen, Chunan II-136  
Chen, Qiming II-162
- Cheng, Jingwei II-447  
Chiu, David II-381  
Chow, Randy I-78  
Chundi, Parvathi I-110  
Clemmer, Aaron I-288  
Codreanu, Dana II-249  
Coletta, Remi II-396  
Collard, Martine II-559  
Conlan, Owen II-319, II-334  
Corrales, Fabian II-381  
Creus Tomàs, Jordi II-203  
Curé, Olivier I-481
- Dabringer, Claus II-144  
Dang, Tran Khanh I-280  
Davies, Stephen I-288  
Dédzoé, William Kokou I-140  
Delgado, Jaime II-234  
Deng, Ke I-270  
Deng, Xiaotie II-279  
de Souza Mendes, Leonardo II-511  
De Virgilio, Roberto II-366  
Dibie-Barthélemy, Juliette I-511  
Ding, Zhiming I-270, II-464  
Dong, Jiawei II-279  
Dou, Dejing II-74  
Dray, Gérard I-457  
Duan, Qiyang II-128  
Duc, Chan Le I-481  
Dumitrescu, Stefan II-249  
Duthil, Benjamin I-457  
Dutta, Sourav II-493
- Eder, Johann II-144  
Engelbrecht, Gerhard II-59
- Fang, Yuan I-187  
Fayn, Jocelyne I-202  
Fegaras, Leonidas II-17  
Ferrarotti, Flavio I-125  
Fialho, Sergio V. II-303  
Finthammer, Marc I-27  
Flouvat, Frédéric II-107  
Frangi, Alejandro F. II-59  
Frasincar, Flavius I-440

- Gançarski, Stéphane I-394  
 Ghedira, Chirine I-202  
 Gonçalves, Luiz M.G. II-303  
 Gruenwald, Le I-496  
 Guo, Xi I-47  
 Gürel, Meltem II-334
- Halfeld Ferrari, Mirian I-94  
 Hampson, Cormac II-319, II-334  
 Han, Jingyu II-574  
 Härder, Theo II-33  
 Hartmann, Sven I-125  
 Hecht, Robin I-481  
 Heendaliya, Lasanthi I-247  
 Hilali-Jaghdam, Inès II-90  
 Hogenboom, Alexander I-440  
 Hogenboom, Frederik I-440  
 Hoppen, Martin I-262  
 Hou, Wen-Chi II-420  
 Hsu, Meichun II-162  
 Hsu, Wynne I-232  
 Huang, Maolin II-43  
 Huang, Sheng II-128  
 Huq, Mohammad Rezwani II-118  
 Hurson, Ali I-247
- Ishikawa, Yoshiharu I-47
- Jen, Tao-Yuan II-90  
 Jiang, Dawei II-574  
 Jurić, Damir II-439
- Karydis, Ioannis I-62  
 Kashyap, Shrikant I-232  
 Kawamoto, Junpei I-341  
 Kaymak, Uzay I-440  
 Kern-Isberner, Gabriele I-27  
 Kheffi, Rania I-511  
 Kołaczowski, Piotr II-475  
 Kouba, Zdeněk II-188  
 Koziris, Nectarios II-527  
 Křemen, Petr II-188  
 Küng, Josef I-280
- Ladwig, Günter I-303, II-171  
 Lamarre, Philippe I-140  
 Lamolle, Myriam I-481  
 Laurent, Dominique II-90  
 Le, Van Bao Tran I-125  
 Leclère, Michel I-466
- Lee, Mong Li I-232  
 Li, Jiang II-43  
 Li, Xiao I-78  
 Li, Xiaodong II-279  
 Li, Xiaou II-544  
 Liang, Weifa I-156  
 Lima, Maria Adriana Vidigal I-94  
 Lin, Dan I-247  
 Link, Sebastian I-125  
 Liu, Haishan II-74  
 Liu, Siqi II-51  
 Liu, Weimo II-136  
 Loglisci, Corrado II-97  
 Luo, Cheng II-420
- Ma, Z.M. II-447  
 Maccioni, Antonio II-366  
 Malerba, Donato II-97  
 Mami, Imene II-396  
 Manolopoulos, Yannis I-62  
 Manzat, Ana-Maria II-249  
 Mao, Dingding II-136  
 Maris, Marinus II-456  
 Marques, Eduardo Zaroni II-511  
 Medjahed, Brahim I-202  
 Min, Geyong I-156  
 Missikoff, Michele II-294  
 Mohamed, Khalil Ben I-466  
 Montmain, Jacky I-457  
 Motomura, Tetsutaro I-410  
 Moyna, Niall II-411  
 Mrissa, Michael I-202  
 Mugnier, Marie-Laure I-466
- Natschläger, Christine II-264  
 Ng, See Kiong I-187  
 Nguyen, Khanh I-425  
 Nicklas, Daniela II-311  
 Nieves, Javier II-519  
 Nin, Jordi II-234  
 Nobari, Sadegh I-448
- Oswald, Annahita I-349  
 Otagiri, Kenichi I-364  
 Ou, Xinming I-217
- Patro, Sunanda I-172  
 Pires, Carlos Eduardo II-502  
 Plantié, Michel I-457  
 Poncelet, Pascal I-457, II-154

- Pratap Singh, Aditya I-320  
 Proietti, Maurizio II-294  
 Pudi, Vikram I-320
- Qin, Han II-74
- Riazati, Dariush II-428  
 Ribeiro, Leonardo Andrade II-33  
 Richter, Christian I-349  
 Roantree, Mark II-366, II-411  
 Roche, Mathieu I-457, II-154  
 Rodriguez, Lisbeth II-544  
 Roßmann, Jürgen I-262  
 Rybiński, Henryk II-475
- Sais, Fatiha I-511  
 Samoladas, Vasilis II-485  
 Sandberg, Jacobijn II-456  
 Santos, Igor II-519  
 Saraiva, Márcio II-502  
 Sawin, Jason II-381  
 Schildhauer, Mark P. I-526  
 Schluse, Michael I-262  
 Schouten, Kim I-440  
 Sedes, Florence II-249  
 Selke, Joachim II-350  
 Selmaoui-Folcher, Nazha II-107  
 Shestakov, Denis I-331  
 Shi, Jie II-411  
 Shimizu, Toshiyuki I-410, II-589  
 Shubhankar, Kumar I-320  
 Signoretti, Alberto II-303  
 Sioutas, Spyros I-62  
 Skočir, Zoran II-439  
 Smith, Fabrizio II-294  
 Stattner, Erick II-559  
 Subramaniam, Mahadevan I-110  
 Sun, Weiwei II-136
- Tajiri, Ricardo Hideyuki II-511  
 Tang, Ruiming I-448  
 Tawaramoto, Kazuki I-341  
 Tbahriti, Salah-Eddine I-202  
 Teisseire, Maguelonne II-154  
 Teitsma, Marten II-456  
 Teja, B. Palvali II-493  
 Thalheim, Bernhard I-12  
 Theodoridis, Yannis I-62  
 Thimm, Matthias I-27
- Thom, James A. II-428  
 Tobin, Crionna II-411  
 Toran, Pere II-234  
 Tous, Ruben II-234  
 Tran, Thanh I-303, II-171  
 Travers, Nicolas II-203  
 Troussel, François I-457  
 Truong, Anh Tuan I-280  
 Tsatsanifos, George II-485  
 Tsichlas, Kostas I-62  
 Tsoumakos, Dimitrios II-527
- Ugarte-Pedrero, Xabier II-519
- Valduriez, Patrick I-1, I-140  
 van der Meer, Otto I-440  
 Vassalos, Vasilis II-1  
 Vidot, Nicolas II-559  
 Vieira, Priscilla II-502  
 Villa-Uriol, Mari-Cruz II-59  
 Vodislav, Dan I-379, II-203
- Wackersreuther, Bianca I-349  
 Wackersreuther, Peter I-349  
 Wagner, Andreas I-303, II-171  
 Wang, Chao II-279  
 Wang, Chuandong II-574  
 Wang, Feng II-279  
 Wang, Guoren II-51  
 Wang, Junhu II-43  
 Wang, Peng II-128  
 Wang, Wei I-172, II-128  
 Waspe, Ralf I-262  
 Watanabe, Yousuke I-364  
 Weerakoon, R.M. Aruna I-110  
 Wielinga, Bob II-456  
 Wombacher, Andreas II-118  
 Wu, Huayu I-448  
 Wu, MingXi II-128  
 Wu, Shengli II-219  
 Wu, Yi I-364
- Xavier-Junior, João C. II-303  
 Xing, Zhaowen I-496
- Yahia, Sadok Ben II-90  
 Yan, Li II-447  
 Yokota, Haruo I-364

Yoshikawa, Masatoshi I-341, I-410,  
II-589  
Yu, Feng II-420

Zanardi, Valentina I-542  
Zarpelão, Bruno Bogaz II-511  
Zeng, Xiaoqin II-219

Zhang, Bin II-162  
Zhang, Fu II-447  
Zhang, Su I-217  
Zheng, Baihua II-136  
Zhu, Qiang II-420  
Zhu, Shanfeng II-279