Weiyi Meng Ling Feng
Stéphane Bressan Werner Winiwarter
Wei Song (Eds.)

LNCS 7825

Database Systems for Advanced Applications

18th International Conference, DASFAA 2013
Wuhan, China, April 2013
Proceedings, Part I

Springer

Lecture Notes in Computer Science 7825

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Weiyi Meng Ling Feng
Stéphane Bressan Werner Winiwarter
Wei Song (Eds.)

Database Systems for Advanced Applications

18th International Conference, DASFAA 2013
Wuhan, China, April 22-25, 2013
Proceedings, Part I

Springer

Volume Editors

Weiyi Meng
Binghamton University. Department of Computer Science
Binghamton, NY 13902, USA
E-mail: meng@binghamton.edu

Ling Feng
Tsinghua University, Department of Computer Science and Technology
100084 Beijing, China
E-mail: fengling@tsinghua.edu.cn

Stéphane Bressan
National University of Singapore, Department of Computer Science
117417 Singapore
E-mail: steph@nus.edu.sg

Werner Winiwarter
University of Vienna, Research Group Data Analytics and Computing
1090 Vienna, Austria
E-mail: werner.winiwarter@univie.ac.at

Wei Song
Wuhan University, School of Computer Science
430072 Wuhan, China
E-mail: songwei@whu.edu.cn

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-37486-9 e-ISBN 978-3-642-37487-6
DOI 10.1007/978-3-642-37487-6
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013934238
CR Subject Classification (1998): H.2-5, C.2, J.1, J.3

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

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

Printed on acid-free paper

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

Preface

It is our great pleasure to present to you the proceedings of the 18th International Conference on Database Systems for Advanced Applications (DASFAA 2013), which was held in Wuhan, China, in April 2013. DASFAA is a well-established international conference series that provides a forum for technical presentations and discussions among researchers, developers, and users from academia, business, and industry in the general areas of database systems, Web information systems, and their applications.

The call for papers attracted 208 submissions of research papers from 28 countries (based on the affiliation of the first author). After a comprehensive review process, the Program Committee selected 51 regular research papers and 10 short research papers for presentation. The acceptance rate for regular research papers is less than 25%. The conference program also included the presentations of three industrial papers selected by the Industrial Committee chaired by Haixun Wang and Haruo Yokota, and nine demo presentations selected from 19 submissions by the Demo Committee chaired by Hong Gao and Jianliang Xu.

The proceedings also include the extended abstracts of the two invited keynote lectures by internationally known researchers, Katsumi Tanaka (Kyoto University, Japan) and Peter M.G. Apers (University of Twente, The Netherlands), whose topics are on "Can We Predict User Intents from Queries? Intent Discovery for Web Search" and "Data Overload: What Can We Do?", respectively. In addition, an invited paper contributed by the authors of the DASFAA 10-year Best Paper Award winner for the year 2013, Chen Li, Sharad Mehrotra, and Liang Jin, is included. The title of this paper is "Record Linkage: A 10-Year Retrospective." The Tutorial Chairs, Jian Pei and Ge Yu, organized four tutorials given by leading experts on a wide range of topics. The titles and speakers of these tutorials are "Behavior-Driven Social Network Mining and Analysis" by Ee-Peng Lim, Feida Zhu, and Freddy Chua, "Understanding Short Texts" by Haixun Wang, "Managing the Wisdom of Crowds on Social Media Services" by Lei Chen, and "Ranking Multi-valued Objects in a Multi-dimensional Space" by Wenjie Zhang, Ying Zhang, and Xuemin Lin. The Panel Chairs, Aoying Zhou and Jeffrey Xu Yu, organized a stimulating panel on big data research. The panel was chaired by Xiaoyang Sean Wang. This rich and attractive conference program of DASFAA 2013 is published in two volumes of Springer's *Lecture Notes in Computer Science* series.

Beyond the main conference, Bonghee Hong, Xiaofeng Meng, and Lei Chen, who chaired the Workshop Committee, put together three exciting workshops (International DASFAA Workshop on Big Data Management and Analytics, International Workshop on Social Networks and Social Web Mining, and International Workshop on Semantic Computing and Personalization). The

workshop papers are included in a separate volume of proceedings also published by Springer in its *Lecture Notes in Computer Science* series.

DASFAA 2013 was primarily sponsored and hosted by Wuhan University of China. It also received sponsorship from the National Natural Science Foundation of China (NSFC), the Database Society of the China Computer Federation (CCF DBS), and the State Key Laboratory of Software Engineering of China (SKLSE). We are grateful to these sponsors for their support and contribution, which were essential in making DASFAA 2013 successful.

The conference would not have been possible without the support and hard work of many colleagues. We would like to express our gratitude to Honorary Conference Chairs, Lizhu Zhou and Yanxiang He, for their valuable advice on all aspects of organizing the conference. Our special thanks also go to the DASFAA Steering Committee for their leadership and encouragement. We are also grateful to the following individuals for their contributions to making the conference a success: the General Co-chairs, Jianzhong Li, Zhiyong Peng and Qing Li, Publicity Co-chairs, Jun Yang, Xiaoyong Du and Satoshi Oyama, Local Arrangements Committee Chair, Tieyun Qian, Finance Co-chair, Howard Leung and Liwei Wang, Web Chair, Liang Hong, Best Paper Committee Co-chairs, Changjie Tang, Hiroyuki Kitagawa and Sang-goo Lee, Registration Chair, Yunwei Peng, Steering Committee Liaison, Rao Kotagiri, APWEB Liaison, Xueming Lin, WAIM Liaison, Guoren Wang, WISE Liaison, Yanchun Zhang, and CCF DBS Liaison, Zhanhuai Li.

Our heartfelt thanks go to all the Program Committee members and external reviewers for reviewing all submitted manuscripts carefully and timely. We also thank all authors for submitting their papers to this conference. Finally, we thank all other individuals and volunteers who helped make the conference program attractive and the conference successful.

April 2013

Weiyi Meng
Ling Feng
Stéphane Bressan
Werner Winiwarter
Wei Song

Organization

Honorary Conference Co-chairs

Lizhu Zhou	Tsinghua University, China
Yanxiang He	Wuhan University, China

Conference General Co-chairs

Jianzhong Li	Harbin Institute of Technology, China
Zhiyong Peng	Wuhan University, China
Qing Li	City University of Hong Kong, China

Program Committee Co-chairs

Weiyi Meng	Binghamton University, USA
Ling Feng	Tsinghua University, China
Stéphane Bressan	National University of Singapore, Singapore

Workshop Co-chairs

Bonghee Hong	Pusan National University, South Korea
Xiaofeng Meng	Renmin University, China
Lei Chen	Hong Kong University of Science and Technology, China

Tutorial Co-chairs

Ge Yu	Northeastern University, China
Jian Pei	Simon Fraser University, Canada

Panel Co-chairs

Aoying Zhou	East China Normal University, China
Jeffery Xu Yu	City University of Hong Kong, China

Demo Co-chairs

Hong Gao	Harbin Institute of Technology, China
Jianliang Xu	Hong Kong Baptist University, China

Industrial Co-chairs

Haixun Wang	Microsoft Research Asia, China
Haruo Yokota	Tokyo Institute of Technology, Japan

Best Paper Committee Co-chairs

Changjie Tang	Sichuan University, China
Hiroyuki Kitagawa	University of Tsukuba, Japan
Sang-goo Lee	Seoul National University, South Korea

Publicity Co-chairs

Jun Yang	Duke University, USA
Xiaoyong Du	Renmin University, China
Satoshi Oyama	Hokkaido University, Japan

Publication Co-chairs

Werner Winiwarter	University of Vienna, Austria
Wei Song	Wuhan University, China

Local Arrangements Chair

Tieyun Qian	Wuhan University, China

Finance Co-chairs

Howard Leung	City University of Hong Kong, China
Liwei Wang	Wuhan University, China

Registration Chair

Yuwei Peng	Wuhan University, China

Web Chair

Liang Hong	Wuhan University, China

Steering Committee Liaison

Rao Kotagiri	University of Melbourne, Australia

WAIM Liaison

Guoren Wang	Northeastern University, China

APWEB Liaison

Xueming Lin	University of New South Wales, Australia

WISE Liaison

Yanchun Zhang	Victoria University, Australia

CCF DBS Liaison

Zhanhuai Li	Northwestern Polytechnical University, China

Program Committees

Research Track

Toshiyuki Amagasa	University of Tsukuba, Japan
Masayoshi Aritsugi	Kumamoto University, Japan
Zhifeng Bao	National University of Singapore, Singapore
Ladjel Bellatreche	Poitiers University, France
Boualem Benatallah	University of New South Wales, Australia
Sourav S Bhowmick	Nanyang Technological University, Singapore
Chee Yong Chan	National University of Singapore, Singapore
Jae Woo Chang	Chonbuk National University, South Korea
Ming-Syan Chen	National Taiwan University, Taiwan
Hong Cheng	Chinese University of Hong Kong, China
James Cheng	Nanyang Technological University, Singapore
Reynold Cheng	University of Hong Kong, China
Byron Choi	Hong Kong Baptist University, China
Yon Dohn Chung	Korea University, South Korea
Gao Cong	Nanyang Technological University, Singapore
Bin Cui	Peking University, China
Alfredo Cuzzocrea	Institute of High Performance Computing and Networking of the Italian National Research Council, Italy
Gill Dobbie	University of Auckland, New Zealand
Eduard C. Dragut	Purdue University, USA
Xiaoyong Du	Renmin University, China
Jianhua Feng	Tsinghua University, China
Jianlin Feng	Sun Yat-Sen University, China
Yunjun Gao	Zhejiang University, China
Wook-Shin Han	Kyung-Pook National University, South Korea
Takahiro Hara	Osaka University, Japan
Bingsheng He	Nanyang Technological University, Singapore
Wynne Hsu	National University of Singapore, Singapore
Haibo Hu	Hong Kong Baptist University, China

Yoshiharu Ishikawa	Nagoya University, Japan
Adam Jatowt	Kyoto University, Japan
Yiping Ke	Institute of High Performance Computing, A*STAR, Singapore
Sang Wook Kim	Hanyang University, South Korea
Young-Kuk Kim	Chungnam National University, South Korea
Hiroyuki Kitagawa	University of Tsukuba, Japan
Hady W. Lauw	Singapore Management University, Singapore
Mong Li Lee	National University of Singapore, Singapore
Sang-goo Lee	Seoul National University, South Korea
Wang-Chien Lee	Pennsylvania State University, USA
Hong-va Leong	Hong Kong Polytechnic University, China
Cuiping Li	Renmin University, China
Guohui Li	Huazhong University of Science and Technology, China
Xiang Li	Nanjing University, China
Xuemin Lin	University of New South Wales, Australia
Jan Lindstrom	IBM Helsinki Lab, Finland
Chengfei Liu	Swinburne University of Technology, Australia
Eric Lo	Hong Kong Polytechnic University, China
Jiaheng Lu	Renmin University, China
Nikos Mamoulis	University of Hong Kong, China
Shicong Meng	IBM Thomas J. Watson Research Center, USA
Xiaofeng Meng	Renmin University, China
Yang-Sae Moon	Kangwon National University, South Korea
Yasuhiko Morimoto	Hiroshima University, Japan
Miyuki Nakano	University of Tokyo, Japan
Vincent T. Y. Ng	Hong Kong Polytechnic University, China
Wilfred Ng	Hong Kong University of Science and Technology, China
Katayama Norio	National Institute of Informatics, Japan
Makoto Onizuka	NTT Cyber Space Laboratories, NTT Corporation, Japan
Sang Hyun Park	Yonsei Universiy, South Korea
Uwe Röhm	University of Sydney, Australia
Ning Ruan	Kent State University, USA
Markus Schneider	University of Florida, USA
Heng Tao Shen	University of Queensland, Australia
Hyoseop Shin	Konkuk University, South Korea
Atsuhiro Takasu	National Institute of Informatics, Japan
Kian-Lee Tan	National University of Singapore, Singapore
Changjie Tang	Sichuan University, China
Jie Tang	Tsinghua University, China
Yong Tang	South China Normal University, China
David Taniar	Monash University, Australia
Vincent S. Tseng	National Cheng Kung University, Taiwan

Vasilis Vassalos	Athens University of Economics and Business, Greece
Guoren Wang	Northeastern University, China
Jianyong Wang	Tsinghua University, China
John Wang	Griffith University, Australia
Wei Wang	University of New South Wales, Australia
Chi-Wing Wong	Hong Kong University of Science and Technology, China
Huayu Wu	Singapore's Institute for Infocomm Research (I2R), Singapore
Xiaokui Xiao	Nanyang Technological University, Singapore
Jianliang Xu	Hong Kong Baptist University, China
Man-Lung Yiu	Hong Kong Polytechnic University, China
Haruo Yokota	Tokyo Institute of Technology, Japan
Jae Soo Yoo	Chungbuk National University, South Korea
Ge Yu	Northeastern University, China
Jeffrey X. Yu	Chinese University of Hong Kong, China
Qi Yu	Rochester Institute of Technology, USA
Zhongfei Zhang	Binghamton University, USA
Rui Zhang	University of Melbourne, Australia
Wenjie Zhang	The University of New South Wales, Australia
Yanchun Zhang	Victoria University, Australia
Baihua Zheng	Singapore Management University, Singapore
Kai Zheng	University of Queensland, Australia
Aoying Zhou	East China Normal University, China
Lei Zou	Peking University, China

Industrial Track

Bin Yao	Shanghai Jiaotong University, China
Chiemi Watanabe	Ochanomizu University, Japan
Jun Miyazaki	Nara Institute of Science and Technology, Japan
Kun-Ta Chuang	National Cheng Kung University, South Korea
Seung-won Hwang	POSTECH, South Korea
Wexing Liang	Dalian University of Technology, China
Ying Yan	Microsoft, USA

Demo Track

Aixin Sun	Nanyang Technological University, Singapore
Chaokun Wang	Tsinghua University, China
Christoph Lofi	National Institute of Informatics, Japan
De-Nian Yang	Institute of Information Science, Academia Sinica, Taiwan

Feida Zhu	Singapore Management University, Singapore
Feifei Li	University of Utah, USA
Guoliang Li	Tsinghua University, China
Ilaria Bartolini	University of Bologna, Italy
Jianliang Xu	Hong Kong Baptist University, China
Jin-ho Kim	Kangwon National University, South Korea
Lipyeow Lim	University of Hawaii at Manoa, USA
Peiquan Jin	USTC, China
Roger Zimmermann	National University of Singapore, Singapore
Shuigeng Zhou	Fudan University, China
Weining Qian	East China Normal University, China
Wen-Chih Peng	National Chiao Tung University, Taiwan
Yaokai Feng	Kyushu University, Japan
Yin Yang	Advanced Digital Sciences Center, Singapore
Zhanhuai Li	Northwestern Polytechnical University, China
Zhaonian Zou	Harbin Institute of Technology, China

External Reviewers

Shafiq Alam	Sheng Li	Wei Tan
Duck-Ho Bae	Yingming Li	Ba Quan Truong
Sebastian Bre	Yong Li	Jan Vosecky
Xin Cao	Bangyong Liang	Guoping Wang
Alvin Chan	Bo Liu	Liaoruo Wang
Chen Chen	Cheng Long	Lu Wang
Lisi Chen	Yifei Lu	Yousuke Watanabe
Shumo Chu	Lydia M	Chuan Xiao
Xiang Ci	Youzhong Ma	Yi Xu
Zhi Dou	Silviu Maniu	Zhiqiang Xu
Juan Du	Jason Meng	Kefeng Xuan
Qiong Fang	Sofian Maabout	Da Yan
Wei Feng	Takeshi Misihma	ByoungJu Yang
Lizhen Fu	Luyi Mo	Xuan Yang
Xi Guo	Jaeseok Myung	Liang Yao
Zhouzhou He	Sungchan Park	Jongheum Yeon
Jin Huang	Peng Peng	Jianhua Yin
Min-Hee Jang	Yun Peng	Wei Zhang
Di Jiang	Yinian Qi	Xiaojian Zhang
Yexi Jiang	Jianbin Qin	Yutao Zhang
Akimitsu Kanzaki	Chuitian Rong	Geng Zhao
Romans Kaspeovics	Wei Shen	Pin Zhao
Selma Khouri	Hiroaki Shiokawa	Xueyi Zhao
Sang-Chul Lee	Matthew Sladescu	Zhou Zhao
Sangkeun Lee	Zhenhua Song	Rui Zhou
Jianxin Li	Yifang Sun	Xiaoling Zhou
Lu Li	Jian Tan	Qijun Zhu

Table of Contents – Part I

Nearest Neighbor Search

Index

Query Analysis

XML Data Management

Privacy Protection

Uncertain Data Management

Table of Contents – Part II

Graph Data Management I

Physical Design

Knowledge Management

Temporal Data Management

Social Networks II

Query Processing II

Graph Data Management II

Data Mining

Applications

Industrial Papers

Demo Papers I: Data Mining

Demo Papers II: Database Applications

Data Overload: What Can We Do?

Peter Apers

CTIT, University of Twente, The Netherlands
p.m.g.apers@utwente.nl, db.cs.utwente.nl

ICT is changing the world. It is changing both individuals and our society. One very intriguing change is that everybody has become both a consumer and a producer.

Before, the production of books and music went through a whole procedure to guarantee quality. Nowadays everyone is putting Youtube videos on the internet, regardless of the quality.

People no longer accept to pay perceived high prices. Many companies and organizations have to rethink their business model. All traditional institutions have to rethink what their added value is.

Also the way we behave is changing. Many people no longer read a book or watch a movie without continuously checking their mail or social media.

Furthermore, Google gives us the feeling that we have access to all the information in the world, resulting in delaying decisions because obtaining more data may lead to perceived better decisions.

All this results in a number of issues that require attention:

- the amount of data is increasing at a tremendous speed, making us actually close to blind;
- the average quality of the data is going down, many data is contradicting each other, can we still trust the data on the internet?
- is it possible to extract reliable knowledge from the Internet.

During the presentation we will discuss a couple of research projects from our group. The research goal of the group is to design algorithms give to provide high quality and highly relevant information along the following lines:

Filtering or searching in a decentralized way and for specific groups. The idea is to involve producers of data in the search. The challenge is to achieve high quality results do in a time-efficient way. Furthermore, children search in a complete different way than adults.

Disclosing semi-structered data, for example from the Deep Web or from logs. Many reliable data sources are hidden in databases hidden behind web services. The challenge is to include these in a simple and uniform way.

Providing transparency by keeping track of how the data presented is obtained. More and more decisions are taken based on data of which the way it is obtained is not clear. Data provenance keeps track on how data is obtained. The challenge is to do this in a memory-efficient way.

Extracting knowledge from large data sets from various sources. The challenges are to link data from different sources based on content, location, time etc and to do it in a time-efficient way.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, p. 1, 2013.

Can We Predict User Intents from Queries?

- Intent Discovery for Web Search -

Katsumi Tanaka

Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo, Kyoto 606-8501, Japan
tanaka@dl.kuis.kyoto-u.ac.jp
http://www.dl.kuis.kyoto-u.ac.jp

Abstract. Although Web search engine technologies have made a great progress in recent years, they are still suffering from the low search performance (precision and recall) because of the following reasons:

(1) Queries for search engines are mostly limited to keywords or short natural language sentences, and

(2) Most search engines use traditional "keyword-in-document" information retrieval models.

Obviously, a user's search intent is not easily expressed by a set of keyword terms. A same keyword-query is formulated and executed by many users, but its search intents (e.g. what information are the "really relevant" answers for the users) are different from users. Also, the traditional "keyword-in-document" IR model assumes that query keywords (and/or related keywords) are contained in the target documents (Web pages). For example, it makes difficult to search for documents (Web pages) whose reputation are specified in user queries.

Search intent discovery is a hot research area in Web search, such as search query classification (informational, navigational and transactional queries), search result diversification, and query recommendation.

In this talk, after a brief survey on the research of search intent discovery and query type classification, we introduce a new framework on search intent discovery and intent-based Web search. In our framework, search-intents are roughly classified into four types: (1) content-related intents (topic-relevance, diversity, comprehensibility, concreteness etc.), (2) task-related intents (search for doing some actions), (3) "social" intents (popularity, typicality, novelty/unexpectedness etc.), and (4) aggregation-based intents (such as retrieving the most expensive Kyoto foods).

Then, we survey our research activities to discover "search-intent types" for user search queries. The proposing methods are based on the usages of ontological knowledge, user behavior data analysis, knowledge extracted from CQA corpus & ads, and "relevance" feedback by intent-based page features.

Keywords: Web search and mining, intent discovery, social data, user behavior analysis.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, p. 2, 2013.

Record Linkage: A 10-Year Retrospective

Chen Li[1], Sharad Mehrotra[1], and Liang Jin[2]

[1] Department of Computer Science, UC Irvine, CA 92697
[2] Microsoft Corporation

Abstract. We describe how we wrote the DASFAA 2003 paper titled "Efficient Record Linkage in Large Data Sets" that received the DASFAA 2013 ten-year best paper award, and the followup research after the paper.

Keywords: Record Linkage, Approximate String Search, Flamingo Package.

1 Paper Background

In a nutshell, our DASFAA 2003 [JLM03] paper was a successful collaboration of two faculty with different research interests to work together on an interesting problem, with the contribution of a first-year PhD student at UC Irvine.

Chen's Research: In October 2001, after graduating from Stanford University, Chen Li joined the faculty of Computer Science at UC Irvine as an assistant professor. To help setup his research agenda as a junior faculty, he started to talk to people at the UC Irvine medical school to get some sense about their data management problems. One of the problems they mentioned was how to link patient records from different information sources that can represent the same real-world person. In particular, each year they obtained records about patients from different databases, and had to link these records with their existing records in a database in order to do additional analysis. Due to the complexity of the problem, these doctors hired contractors to do this linkage, thus had to spend a significant amount of resources. An efficient and effective solution will be very appealing to these practitioners.

Chen's previous research was in the field of data integration, i.e., integrating information from heterogeneous sources. He worked on topics related to dealing with limited query capabilities of sources and answering queries using views. Given the connection between data integration and record linkage, he decided to get deeper into this research area, and quickly realized that the problem was much more complicated than it looked. There are many technical challenges related to accuracy and efficiency. For instance, what is a good function to measure the similarity between two records? How to use domain-specific knowledge to increase the confidence of matching? How to efficiently identify candidate pairs of records, especially when the data set is large? Among these challenges, he decided to work on the issues related to efficiency. He also started a research project called Flamingo (http://flamingo.ics.uci.edu/) to study problems.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 3–12, 2013.

Sharad's Research: At the time, one of the areas in which Sharad was working on was content-based multimedia retrieval. He, along with his students, had engineered a *Multimedia Analysis and Retrieval System* (MARS) which was amongst the first systems to explore relevance feedback and query refinement in multimedia search. A major contribution of MARS was scalable multidimensional indexing and support for similarity queries. This included data structures such as hybrid tree for scalable range and KNN queries over highly multidimensional spaces, mechanisms to map multimedia objects (e.g., time series) with domain specific distance measures into multidimensional spaces which preserved the original distance metric, and methods for dimensionality reduction, etc. This prior research proved vital for coming up with the approach to address efficient record linkage.

At the time the dominant record linkage approach, that overcame the quadratic nature of the problem, was merge/purge. Merge/purge exploited the fact that often one can identify a set of keys such that if data is sorted based on the keys, duplicate records would be close in the sort order for at least one such key. Merge/purge effectively reduced the record linkage to that of sorting which could be implemented much more efficiently. Nonetheless, the approach had limitation – how should such keys be identified, how should the appropriate window size for merge be chosen? Sharad's prior work on distance preserving dimensionality reduction provided a useful clue to overcome the challenges of designing a scalable record linkage solution: one could map records to a multidimensional space in ways that the mapped points will preserve the distance based on the domain-specific measures (such as edit distance). If we could do such a mapping, multidimensional indexing and similarity joins could be used to realize record linkage.

The Coming Together: Chen and Sharad started collaborating on the problem with the help of Liang Jin, who was a new student who had joined the PhD program in the same year. The three authors started technical discussions on this topic. We decided to formulate the problem as finding pairs of records from a large data set that are similar enough, where we use common functions such as edit distance and Jaccard. One technical difficulty is that strings using these functions do not have a total order. Instead, they are in a metric space. Given our previous knowledge in multi-dimensional indexing and embedding techniques, we came out with of idea of trying the technique called "FastMap" by Christos Faloutsos and King-Ip Lin. This technique maps objects in a metric space to a high-dimensional Euclidean space while preserving their pairwise distances as much as possible. Since string similarity functions we considered are metric spaces, we can first use this technique to embed the strings into a Euclidean space, then use existing high-dimensional indexes such as R-tree or hybrid tree to do similarity queries.

After identifying the direction, the next question was whether this embedding method can preserve pairwise string similarity well. The first experimental results, as shown in Figure 4 in the original paper, showed a very promising evidence. In particular, most similar pairs of strings are still very similar to

each other in the new Euclidean space, most dissimilar strings are different from each other in the new space, and we can find a good distance threshold to separate these two classes of string pairs. We further studied how to use a multi-dimensional structure (R-tree in particular) to index the mapped objects, and use the structure to answer queries. We also studied how to deal with the case where the records have multiple attributes, and how to efficiently answer a similarity query with multiple attributes.

2 Followup Work on Approximate String Queries

The DASFAA 2003 paper was Chen's first work in the field of data cleaning. Ever since then he led a research team to study various problems in the context of the Flamingo project. The following are some of the techniques developed in the project:

- How to improve nearest-neighbor search using histograms [JKL04].
- How to support approximate queries with predicates of both numerical conditions and string conditions [JKLT05].
- How to estimate the selectivity of fuzzy string predicates [JL05, JLV08].
- How to relax SQL queries with join and selection conditions [KLTV06].
- How to improve the performance of approximate string queries using grams with variable lengths [LWY07, YWL08].
- How to use inverted lists and filtering techniques to do approximate string queries [LLL08].
- How to do compression of inverted lists in approximate string search [BJLL09].
- How to do efficient top-k queries in approximate string search [VL09].
- How to do spatial approximate string queries [AL10, ABL10]
- How to use disk-based indexing to do approximate string queries [BLC11].
- How to do approximate string joins using Hadoop [VCL10].
- How to do interactive fuzzy search [JLLF09].

Realizing the importance of making research techniques available by releasing their source code, Chen led the team to start building a C++ package that included the techniques published in these paper. The first technique included in the package was the StringMap algorithm described in the DASFAA 2003 paper [JLM03]. Over the next eight years, the package had gradually become more stable and powerful, thanks to the feedback from many colleagues and users.

In addition, Chen started working on the area of doing powerful search by supporting instant fuzzy search, in the context of the ipubmed project[1]. He also started a company called SRCH2 (http://www.srch2.com) to commercialize the research results. He wrote an article at the ACM SIGMOD BLOG[2] to describe the experiences of research commercialization. SRCH2 has developed a search

[1] `http://ipubmed.ics.uci.edu`
[2] `http://wp.sigmod.org/?page_id=52`

engine (built from the ground up in C++) targeting enterprises that want to enable a Google-like search interface for their customers. It offers a solution similar to Lucene and Sphinx Search, but with more powerful features such as instant search, error correction, geo support, real-time updates, and customizable ranking. Currently its first products are developed and it has paying customers.

3 Followup Work on Overcoming the Quality Curse in Data Cleaning

Following the work described in our DASFAA 2003 paper, our exploration on data cleaning shifted to addressing the data quality challenge at a deeper level[3]. At the time, most conventional record linkage techniques were based on comparing similarity between records , which, in turn, was computed using a combination of similarities at the level of individual attributes/features. Attribute similarities were domain-specific based on edit distances, Jaro metric, N-gram based distances, cosine similarity, or special techniques suited for comparing names, etc. Often, techniques such as agglomerative merging, partitioning, relational clustering were applied as a post-processing step after computing similarities in order to finally group the same records. While data management research primarily focused on techniques to overcome the quadratic complexity of comparing records (e.g., the merge/purge, and our DASFAA paper being examples), the related machine learning research focused on techniques to learn similarity functions at the attribute level, at the record level, and on building appropriate classifiers to determine if indeed two records were identical. Multiple proposals included use of EM algorithm [Winkler 99], decision trees, hidden markov models, support vector machines, conditional random fields, etc.

One of our observations, at the time, was that such similarity-based computations exhibit fundamental limitations. Let us illustrate this in the context of the related problem of entity resolution where the goal is to determine if two references to entities co-refer; that is, they refer to the same real-world entity. Consider for instance, the following two text segments: *"Photo collection of Michael Carey from Beijing China 2007 Sigmod trip"* and *"Mike Carey, research interests: data management, Professor UC Irvine"* – these two entities could be records in two different databases being merged and the corresponding record linkage problem being whether the two entities refer to the same person. If we simply use feature based matching (features being concepts /words in the vicinity of the reference), we would never be able to reconcile the two references to be the same person, which, in this example, is indeed the case. Approaches based on feature-matching could also make the reverse mistake by wrongfully merging records corresponding to different entities. To see this, consider another two text segments: *"S. Mehrotra has joined the faculty of Univ. of Illinois. He received his PhD from UT, Austin, and a bachelors from India"* and *"S. Mehrotra, Ph.D.*

[3] The work described in this section is a result of a decade long collaboration between one of the authors with Dmitri Kalashnikov.

$(A_1, `DaveWhite', `Intel')$
$(A_2, `DonWhite', `CMU')$
$(A_3, SusanGrey', `MIT')$
$(A_4, `JohnBlack', `MIT')$
$(A_5, `JoeBrown', unknown)$
$(A_6, `LizPink', unknown)$

Fig. 1. *Author* records

$(P_1, `Databases \ldots', `JohnBlack', `DonWhite')$
$(P_2, `Multimedia \ldots', `SueGrey', `\mathbf{D.\ White}')$
$(P_3, `Title3 \ldots', `DaveWhite')$
$(P_4, `Title5 \ldots', `DonWhite', `JoeBrown')$
$(P_5, `Title6 \ldots', `JoeBrown', `LizPink')$
$(P_6, `Title7 \ldots', `LizPink', `\mathbf{D.\ White}')$

Fig. 2. *Publication* records

from University of Illinois is visiting UT Austin to give a talk on prefetching on multiprocessor machines. He received his bachelors from India"[4]. While reading the two segments, we can straightaway tell that the references belong to two different individuals, if an automated technique based on feature-based similarity were to be used (where features correspond to say words in the neighborhood of the entity reference), the system may mistakenly decide that these are indeed the same person. While one could argue that we can overcome the problem by extracting semantically meaningful features (e.g., using NLP), the point is not whether we can design mechanisms suited for this instance. Rather, it is to highlight that similarity-based methods that just compare the records based on features can make mistakes both by wrongly merging records/entities that are distinct and also by splitting the same entity into different group.

Deep Dive into Data. Amongst the first methods we tried to overcome the quality-challenge of traditional similarity based methods was to explore "additional semantic information hidden in the data in the form of relationships. This led us to an approach we named *Relationship Based Data Cleaning* (RelDC). Let us illustrate RelDC using an example from the publication domain. Consider a set of author and publication records shown in Figures 1 and 2 above.

Our goal is to match the authors in the publication table to the right author in the author table. Existing feature based similarity methods would allow us to identify that Sue Grey in paper P_2 corresponds to author A_3. But what about D. White in P_2 and in P_6 - they could match either A_1 (Dave White) or A_2 (Don White)? If we look more closely at data, we discover that Don White has co-authored a paper (P_1) with John Black who is at MIT, while the author Dave White does not have any co-authored papers with authors at MIT. We can use this observation to disambiguate between the two authors. In particular, since the co-author of D. White in P_2 is Susan Grey of MIT, there is a higher likelihood that the author D. White in P_2 is Don White. The reason is that the data suggests a connection between author Don White with MIT and an absence of it between Dave White and MIT. Second, we observe that author Don White has co-authored a paper (P_4) with Joe Brown who in turn has co-authored a paper with Liz Pink. In contrast, author Dave White has not co-authored any papers with either Liz Pink or Joe Brown. Since Liz Pink is a co-author of P_6,

[4] There were indeed two different S. Mehrotra at Illinois at the same time, one, who is a co-author of this paper, who was a professor and the other a Ph.D. student specializing in architecture.

there is a higher likelihood that D. White in P_6 refers to author Don White compared to author Dave White.

RelDC captured the above intuition that relationships help disambiguate references into a general principle by modeling the database as a graph wherein entities corresponded to nodes and edges to relationships between the nodes. The main hypothesis of RelDC was that if we model the data as an instantiated ER graph, the nodes in the graph that are more connected (and hence have stronger relationships) are more likely to co-refer to the same real-world entity. The hypothesis captured the intuition that existing relationships (e.g., if two individuals have co-authored a paper) promotes new additional relationships (e.g., they may co-author again) and such a relationship analysis can be used as evidence to disambiguate.

Of course, the devil was in the detail. How do we measure strength of connectivity amongst nodes in a graph? Not all relationships are equally important evidences for disambiguation, how do we discover relative importance automatically? How do we self-tune the approach? Can we design a general purpose approach that exploits relationships for disambiguation which is domain-independent? How do we integrate such relationship evidences with other evidences such as feature-based similarity? Are there specific domains where relationships work better for disambiguation compared to others? How do we make such analysis efficient given the (potentially) large size of the data.

Stella Chen's Ph.D. thesis and work in [NTKM13, NTKM12, NTKMY12] and [NTCKM09, KCMNT08, KNTM08, NTKM07, KCNT$^+$09] explored the above challenges building a general purpose data cleaning framework (which we called Graph Disambiguation Framework (GDF) that exploited relationship analysis. Her experimental results showed significant improvements in performance on a variety of benchmark problems.

Looking Outside the Box. The second approach we explored to address the quality-curse was to look *outside the box* for evidences to merge (or split) records. We were by no means the first to explore such an idea, there was already research on using focused crawling, ontologies, Wikipedia, taxonomies as external data sources to help reconcile records in a database. But to the best of our knowledge, we were the first to exploit search engine statistics (learnt through web queries) to learn correlations amongst entities which significantly helped in the disambiguation decisions. The basic intuition was straightforward and let us illustrate it in the context of a *people search* task on the internet.

People Search Challenge: Imagine a search engine that allows users to specify names of individuals as queries and clusters the results based on entities. Such a system can support an interface wherein information about different namesakes can be shown to the user enabling the user to choose which particular entity he/she meant to retrieve information about. There are significant advantages of such a technology, e.g., imagine searching for Bill Clinton, no not the ex-president, but perhaps an old acquaintance who happens to share his name with the president (a famous person). It is very difficult to search for such individuals

using toady's search technology since most results obtained would belong the president. Instead, a clustered result would group all web pages of the president together into a single entry enabling information about other entities to be visible on the first page. The challenge in building such a technology is disambiguating references on web pages to decide which Bill Clinton it corresponds to. We refer to this as the *People search* challenge.

Now let us illustrate how correlations learnt from web search engine statistics can help disambiguate references on the web page. Imagine a query looking for Jeff Ullman on the web. Consider two returned pages: one page contains concepts/entities such as Jennifer Widom, data management, recursive queries and another which refers to entities such as Renee Miller and VLDB 2007. At first sight, the entities occurring on the two pages are distinct and do not offer evidence that the two pages refer to the same Jeff Ullman. However, statistics about the number of pages in which entities Jennifer Widom and VLDB 2007 both co-occur and likewise Data management and Renee Miller co-occur can provide strong evidence that quite likely the two Jeff Ullman references are to the same person. The challenge, of course, include

- What type of queries should be submitted to the search engine to learn information about interrelationships amongst the contextual entities on the web pages returned as part of the original people search query?
- How should such statistics be used as evidence and how should such evidence be combined with other features to make robust decisions?
- How do we handle situations when the web page contains very little contextual information causing it not to cluster with other pages – this is a common phenomena in people search.
- Querying search engine statistics on the web is extremely time consuming which is further complicated with API restrictions imposed by most search engines. What techniques can be used to limit the number of queries so that execution time can be controlled?
- Response time of search engines are outside the control of end-users, thereby making the approach vulnerable to the vagaries of the search engine. What can be done to support some degree of predictability to the amount f time the approach takes?

Rabia Nuray's Ph.D. dissertation [KMC05, CKM05, KM06, CKM07, CKM09] addressed the above challenges in a sequence of papers culminating in a people search engine prototype which we refer to as *Web-people search engine* (WEST) which added a layer of disambiguation atop results of a search engine such as google. The WEST system turned out to be a useful experience, if for no other reason, but as a source of entertainment to our visitors at UCI when they would play with the system to see if it could enable them to find web pages of acquaintances who might not necessarily have a significant web presence, or to check if WEST would cluster all the right pages for friends/colleagues they knew well.

4 What's Next?

The paper, so far, has described how we came up with the DASFAA 2003 paper and provided an overview of our related work since then. We conclude the paper with our view of where the area of data quality is headed. First, we note that the area of record linkage (and data quality in general) has continued to receive a large amount of research attention over the past decade resulting in a large number of new innovative ideas and techniques which address the problem from variety of perspectives from techniques to exploring additional sources and types of information to achieve high quality – e.g., analyzing relationships, domain constraints, ontologies, encyclopedia, general external datasets, issuing web queries, and so forth. Equally active has been approaches to gain efficiency as exemplified by approaches such as Stanford's Swoosh project.

In our view, however, significant challenges lie ahead and the coming years are going to witness a whole set of new technologies related to the area. To highlight one such new challenge, we note that data cleaning technologies, traditionally designed to improve quality of data in back-end data warehouses, are now fast emerging as a vital and important component of real-time information access. As the Web evolves towards supporting interactive analytics and basic search migrates from simple keyword retrieval to retrieval based on semantically richer concepts (e.g., entities) extracted from web pages, the need for *on-the-fly* cleaning techniques that can help alleviate data quality challenges (e.g., errors, ambiguities, incompleteness, etc.) in such automatically generated content is rapidly increasing. While efficiency has always been an important component of data cleaning research, we also need finer (more aggressive) control over how data is cleaned and what the costs of cleaning are. Furthermore, since in such applications, data cleaning is embedded in the information access/processing pipelines, the result of cleaning need to be conveyed in the form comprehensible to end recipient (viz., individual users/analysis code).

References

[ABL10] Alsubaiee, S.M., Behm, A., Li, C.: Supporting location-based approximate-keyword queries. In: ACM SIGSPATIAL GIS (2010)

[AL10] Alsubaiee, S., Li, C.: Fuzzy keyword search on spatial data. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010, Part II. LNCS, vol. 5982, pp. 464–467. Springer, Heidelberg (2010)

[BJLL09] Behm, A., Ji, S., Li, C., Lu, J.: Space-constrained gram-based indexing for efficient approximate string search. In: ICDE (2009)

[BLC11] Behm, A., Li, C., Carey, M.J.: Answering approximate string queries on large data sets using external memory. In: ICDE, pp. 888–899 (2011)

[CKM05] Chen, Z., Kalashnikov, D.V., Mehrotra, S.: Exploiting relationships for object consolidation. In: IQIS, pp. 47–58 (2005)

[CKM07] Chen, Z., Kalashnikov, D.V., Mehrotra, S.: Adaptive graphical approach to entity resolution. In: JCDL, pp. 204–213 (2007)

[CKM09] Chen, Z., Kalashnikov, D.V., Mehrotra, S.: Exploiting context analysis for combining multiple entity resolution systems. In: SIGMOD Conference, pp. 207–218 (2009)

[JKL04] Jin, L., Koudas, N., Li, C.: Nnh: Improving performance of nearest-neighbor searches using histograms. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 385–402. Springer, Heidelberg (2004)

[JKLT05] Jin, L., Koudas, N., Li, C., Tung, A.K.H.: Indexing mixed types for approximate retrieval. In: VLDB, pp. 793–804 (2005)

[JL05] Jin, L., Li, C.: Selectivity estimation for fuzzy string predicates in large data sets. In: VLDB, pp. 397–408 (2005)

[JLLF09] Ji, S., Li, G., Li, C., Feng, J.: Efficient interactive fuzzy keyword search. In: WWW, pp. 371–380 (2009)

[JLM03] Jin, L., Li, C., Mehrotra, S.: Efficient record linkage in large data sets. In: DASFAA, pp. 137–146 (2003)

[JLV08] Jin, L., Li, C., Vernica, R.: SEPIA: estimating selectivities of approximate string predicates in large databases. VLDB J. 17(5), 1213–1229 (2008)

[KCMNT08] Kalashnikov, D.V., Chen, Z., Mehrotra, S., Nuray-Turan, R.: Web people search via connection analysis. IEEE Trans. Knowl. Data Eng. 20(11), 1550–1565 (2008)

[KCNT+09] Kalashnikov, D.V., Chen, Z., Nuray-Turan, R., Mehrotra, S., Zhang, Z.: West: Modern technologies for web people search. In: ICDE, pp. 1487–1490 (2009)

[KLTV06] Koudas, N., Li, C., Tung, A.K.H., Vernica, R.: Relaxing join and selection queries. In: VLDB, pp. 199–210 (2006)

[KM06] Kalashnikov, D.V., Mehrotra, S.: Domain-independent data cleaning via analysis of entity-relationship graph. ACM Trans. Database Syst. 31(2), 716–767 (2006)

[KMC05] Kalashnikov, D.V., Mehrotra, S., Chen, Z.: Exploiting relationships for domain-independent data cleaning. In: SDM (2005)

[KNTM08] Kalashnikov, D.V., Nuray-Turan, R., Mehrotra, S.: Towards breaking the quality curse.: a web-querying approach to web people search. In: SIGIR, pp. 27–34 (2008)

[LLL08] Li, C., Lu, J., Lu, Y.: Efficient merging and filtering algorithms for approximate string searches. In: ICDE, pp. 257–266 (2008)

[LWY07] Li, C., Wang, B., Yang, X.: VGRAM: Improving performance of approximate queries on string collections using variable-length grams. In: VLDB, pp. 303–314 (2007)

[NTCKM09] Nuray-Turan, R., Chen, Z., Kalashnikov, D.V., Mehrotra, S.: Exploiting web querying for web people search in weps2. In: WePS (2009)

[NTKM07] Nuray-Turan, R., Kalashnikov, D.V., Mehrotra, S.: Self-tuning in graph-based reference disambiguation. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 325–336. Springer, Heidelberg (2007)

[NTKM12] Nuray-Turan, R., Kalashnikov, D.V., Mehrotra, S.: Exploiting web querying for web people search. ACM Trans. Database Syst. 37(1), 7 (2012)

[NTKM13] Nuray-Turan, R., Kalashnikov, D.V., Mehrotra, S.: Adaptive connection strength models for relationship-based entity resolution. ACM JDIQ (2013)

[NTKMY12] Nuray-Turan, R., Kalashnikov, D.V., Mehrotra, S., Yu, Y.: Attribute and object selection queries on objects with probabilistic attributes. ACM Trans. Database Syst. 37(1), 3 (2012)

[VCL10] Vernica, R., Carey, M., Li, C.: Efficient parallel set-similarity joins using MapReduce. In: SIGMOD Conference (2010)

[VL09] Vernica, R., Li, C.: Efficient top-k algorithms for fuzzy search in string collections. In: KEYS, pp. 9–14 (2009)

[YWL08] Yang, X., Wang, B., Li, C.: Cost-based variable-length-gram selection for string collections to support approximate queries efficiently. In: SIGMOD Conference (2008)

Finding Rising Stars in Social Networks

Ali Daud[1], Rashid Abbasi[1], and Faqir Muhammad[2]

[1] Department of Computer Science and Software Engineering, Sector H-10, International Islamic University, Islamabad 44000, Pakistan
[2] Department of Business Administration, Sector E-9, Air University, Islamabad 44000, Pakistan
ali.daud@iiu.edu.pk, {abbasi_20150,aioufsd}@yahoo.com

Abstract. This paper addresses the problem of finding rising stars in academic social networks. Rising stars are the authors which have low research profile in the beginning of their career but may become prominent contributors in the future. An effort for finding rising stars named PubRank is proposed, which considers mutual influence and static ranking of conferences or journals. In this work an improvement of PubRank is proposed by considering authors' contribution based mutual influence and dynamic publication venue scores. Experimental results show that proposed enhancements are useful and better rising stars are found by our proposed methods in terms of average citations based performance evaluation. Effect of parameter alpha and damping factor is also studied in detail.

Keywords: Rising stars, author contribution, dynamic publication venue score, PageRank, Academic Social Networks.

1 Introduction

Academic social networks are made up of co-author and citation based relationships between authors and research papers, respectively. Co-author means the authors writing paper together and citation based relationships occur when one paper cites other papers or is cited by other papers. Academic social network analysis has many interesting research tasks such as expert finding [7], author interest finding [6] citation recommendations [8] name disambiguation [17] and rising star finding [20]. This work is focused on finding rising stars. The motivation is to find new born researchers with abilities to become stars or experts in future. All those persons, who may not be at the top at the moment or are not experienced, but are capable to be at the top position in their respective fields in near future, are referred to as Rising Stars. Finding rising stars is very useful for appointing young faculty members to increase research productivity of department, finding reviewers for conferences and journals which can provide reviews on time and making them members of different academic committees to get benefit from their dynamic and energetic behavior.

An effort made for finding rising stars considers mutual influence and static (a ranking list of publication venues) importance named PubRank [20]. The idea was if a

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 13–24, 2013.

junior author influences/collaborates with a well known researcher and the publication venue is of high rank (1,2,3 where 1 is higher level, 2 is normal level and 3 is low level) he has bright chances to become a star in future. There were two major problems with the existing method (1) the authors did not consider the author contribution oriented mutual influence in PubRank which is very important when one wants to calculate the influence of one author on another. Here, contribution means the order in which the authors appears in the paper such as first author, second author and so on, with first author is usually considered the main contributor of that work. The junior author who influences/collaborates with well known researchers as main contributor of work has more chances than that of a junior author simply influencing/collaborating with well known researchers and (2) using static rankings is not practical as quality of work published in publication venues changes every year so as the ranking of publication venues and old static ranking lists available on the web does not provide latest rankings of publication venues. Due to aforementioned reasons we are motivated to propose StarRank algorithm which overcomes the limitations of PubRank in easy way. Our proposed method considers author contribution based mutual influence of authors on each other's in terms of order in which authors appears in the paper as well as latest (dynamic) scores of rankings for publication venues which is calculated using entropy. Our intuition to use entropy is based on the fact that the venues which are stricter in accepting papers to their areas of research are of higher level and has less entropy as compared to the venues which are not very strict in accepting papers to their areas of research and has higher entropy. Here one thing needs to be made clear that usage of entropy for scoring publication venues is workable for conferences/journals but not for workshops as they accept topic specific papers but they do not need to be of high quality because they are not finished or top level papers mostly.

Our hypothesis is supported with the detailed experimentation which shows that our proposed StarRank outperformed existing method clearly for rising star finding task. The effect of algorithm parameters is also studied in details to find their suitable values for rising star finding task.

The major contributions of this work are (1) contribution oriented co-author weight (2) entropy based dynamic publication venue score (3) unification of contribution oriented weight and dynamic publication venue score (4) and experimental evaluation of our proposed method on the real world dataset of DBLP.

The rest of the work is arranged as follows. Section 2 provides the literature review of tasks performed in academic social networks followed by the applications of page rank in these networks. Section 3 provides the existing method with the detailed approach proposed by us for finding rising stars. Section 4 provides dataset description, performance evaluation procedure, parameter settings with results and discussions in different scenarios and section 5 finally provides the concluding remarks.

Several concepts are used interchangeably in this paper such as academic social networks and co-author networks, conferences/journals and publication venues, papers, research papers and publications etc.

2 Related Work

2.1 Tasks in Academic Social Networks

Author interest finding focuses on who have interest in writing on some topics [6]. Authors based on their areas of research chose one or more topics to work on. Expert finding addresses the task of finding authors which are well known in their area of research [7]. Online publication databases like DBLP and CiteSeer provide very useful information in which names are inconsistent, which is called named entity disambiguation task. Name entity disambiguation task have two major challenges which are (1) name sharing and (2) name variation [17].

Finding Association aims at discovering the relationships between nodes or people in social networks. Email networks also provide associations between the senders and receivers in several ways [1].With the emergence and rapid explosion of social applications and media, such as instant messaging (e.g. MSN, Yahoo, Skype) and blogs (e.g., Blogger, Live Journal) finding and quantifying the social influence of actors on each other is significant [18].The research has become too much planned and sophisticated these days. There are several challenging factors need to be considered, e.g. how to find that if someone is expert of a topic either he can be good advisor or not? [19].

The competition between researchers has become very challenging these days so they want paper writing a quick process, so finding accurate citations quickly is important [8]. Community mining in co-author networks is important problem, where authors are connected to each other by co-authorships or paper citations and thus can be modeled as interaction graphs by considering semantics-based generalized topic models [9].

2.2 Applications of Page Rank

PageRank is a very useful algorithm for ranking pages or important entity finding in graphs. TextRank [15] was proposed for extracting keywords, key phrases and sentences from the documents and comparable results with supervised learning algorithms are achieved. A weighted directed model AuthorRank for co-authors network is proposed by Xiaoming et al., [13]. The importance of an individual author and its popularity is weighted through prestige. People tag resources in the web according to their understanding of those resources results in developing social tagging systems which have emerged quickly on the web. FolkRank [11] algorithm is proposed to rank users, tags and resources on the basis of undirected links between them. An important area of research in Bioinformatics is biological network analysis. Personalized PageRank [12] is proposed to find important proteins in protein networks. Finding rising star is investigated through mutual influence and static ranking of conferences/journals. Mutual influence did not consider author contributions and static rankings usage is not correct as rankings of conferences/journals keeps on changing so are dynamic, which motivated us to propose StarRank [20].

3 Finding Rising Stars

In this section, before describing our (1) author contribution based, (2) dynamic publication quality based, and (3) composite StarRank approaches, we briefly introduce related existing approach PubRank [20] for rising star finding.

3.1 PubRank

PubRank method [20] was proposed to find rinsing stars from academic social networks. It can be used to find authors which can be future experts. They considered two main points (1) Mutual influence among the researchers in term of co-authorships and (2) track record of researcher's publications in terms of publishing in different level of publication venues.

Firstly, a graph is considered in which nodes describe authors and edges describe co-author relationships for calculating mutual influence. The main idea was that if a junior researcher can collaborate with expert senior researchers or can be able to influence their work he has bright chances to be a future expert. A novel link weighting strategy is proposed. When authors (A_k,A_l) are co-author in any article to calculate the weight of author $A_k$they put the weight A_k= (A_k,A_l) as fraction of A_l author which is co-author with A_k. Moreover, the weight of A_l = (A_l ,A_k) is fraction of A_k. This weighting strategy is based on the intuition that a junior researcher will influence its senior researcher less and senior will influence more as he has more publications, which reduces the junior researcher fraction of co-authored work. The following example explains how the influence is calculated.

Suppose we have two authors K with 4 papers and L with 3 papers. If they have co-authored two papers with each other, the weight with which they influence each other is calculated as:

$$W(A_l, A_k) = \frac{(A_l,A_k)}{PA_k} = \frac{2}{4} = 0.4\,, W(A_k, A_l) = \frac{(A_k,A_l)}{PA_l} = \frac{2}{3} = 0.66 \quad (2)$$

Here, PA_l and PA_k are the total number of papers written by authors L and K. (A_l,A_k) is the number of co-authored papers between authors L and K. The weight $W(A_l,A_k)$ with which author A_l influences author A_k is less as compared to the weight $W(A_k,A_l)$ with which author A_k influences author A_l. As the number of papers written by author A_k are 5 which are more in number as compared to A_l, So, A_k is a senior and influences A_l more.

Secondly, prestige of publication venue is considered for calculating the track record of researcher's publications. The reputation of a researchers work can by judged by number of citations his papers have which is biased towards earlier publications as publication needs time to be cited by other papers. Rising starts cannot have many highly cited papers. So the publication venue levels in which they have published the papers are considered. The main idea behind this intuition is that if a researcher is publishing in high level venues in the beginning of his career he has bright chances to be an expert in future. A static ranking scheme available on web [14] is

used with following ranks. Rank 1 (premium), Rank 2 (leading), Rank 3 (reputable) and Rank 4 (unranked). The publication quality score for each researcher is calculated by using the following equation.

$$\lambda(A_i) = \frac{1}{|p|} * \sum_{i=1}^{p} \frac{1}{\propto^{r(pub)-1}} \tag{3}$$

where, $\lambda(A_i)$ is publication quality score of author A_i all publications the larger the better,pub_i is i^{th} publication, r (pub) is publication rank of the paper and $\propto$ value is between (o<$\propto$<1) which is damping factor so low rank publications can have low scores.

Finally, mutual influence among the researchers and track record of researcher's publications is hybridized in PubRank as follows.

$$pubRank(A_i) = \frac{1-d}{n} + d * \sum_{j=1}^{|a|} \frac{W(A_j,A_i)*\lambda(A_i)*pubRank(A_j)}{\sum_{k=1}^{|a|} W(A_k,A_j)*\lambda(A_k)} \tag{4}$$

where, n is total number of scientist, $W(A_j, A_i)$ is weight for authors influencing author A_i,$\lambda(A_i)$ is publication quality score of author $p_i$and $pubRank(A_j)$ is the PubRank of authors linking to author A_i.

3.2 StarRank

In this section, StarRank is proposed by us to find rinsing stars. Based on the intuition of Sekercioglu [5] of quantifying coauthor contribution in a research paper we proposed author contribution based mutual influence calculation method among the researchers. We also proposed a dynamic way of calculating the scores for publication venues in comparison to simply using a static list(s), which provided out dated ranking [20]. Finally both are combined to propose Composite StarRank.

3.2.1 Author Contribution Based StarRank (AC StarRank)

A graph is considered in which nodes describe authors and edges describe co-author relationships for calculating mutual influence in PubRank [20]. In addition to this information the order in which authors are appeared in the papers is also considered with first author as maximum contributor. Less contributed author score would be less and more contributed author score should be more based as discussed by Sekercioglu [5]. When a paper has more than one co-author in that case equal contribution score given to all of them is unfair. We have proposed a novel link weighting strategy based on author order based contributions. For example, a paper has 4 authors and if an author appears at number one, he will have more contribution as compared to author appeared at number four. The main idea is that if a junior researcher can collaborate with senior researchers and can influence their work as a main contributor such as by appearing at number one or two in the paper he has bright chances to become future expert.

Suppose we have four authors K, L, M, N; K with 4 papers, L with 3 papers, M with 4 and N with 3 papers with each other. The order in which they appear in the paper is given in the following. The author K and L co-authored and M and N co-authored the papers with each other highlighted as bold face letters in the following table.

Table 1. Authors with their papers and order of appearance

Authors	Paper No (order of appearance)
K	**1(1), 2(3),** 3(2), 4(1)
L	**1(2), 2(2),** 3(1)
M	**1(1), 2(3),** 3(2), 4(1)
N	**1(3), 2(4),** 3(4)

One can see that L is junior researcher in co-authorship of K and L as L has 3 papers and K has 4 papers and N is junior researcher in co-authorship of M and N as N has 3 papers and M has 4 papers. In this co-authorship scenario L has 1 paper as first author and 2 papers as second author while N has 1 paper as third author and 2 papers as fourth author.

$$ACW(A_l, A_k) = \frac{(\sum AC_l + \sum AC_k)}{\sum PAC_k} = \frac{(0.5+0.5)+(1+0.33)}{1+0.33+0.5+1} = 0.823 \quad (5)$$

$$ACW(A_n, A_m) = \frac{(\sum AC_n + \sum AC_m)}{\sum PAC_m} = \frac{(0.33+0.25)+(1+0.33)}{1+0.33+0.5+1} = 0.67$$

Here, $\sum PAC_k$and $\sum PAC_m$is the total contribution of author K and M in all papers written by them. $\sum AC_l + \sum AC_k$ is the authors L and K contributions of co-authored papers. For example, author L has co-author paper 1 with author K as second author 2(2) so its contribution is ½ =0.5, which we get from 1/R where R is the rank of author [5] in the co-authored papers.

The author contribution weight $ACW(A_l, A_k)$ with which author A_l influences A_k is 0.823 which the contribution weight $ACW(A_n, A_m)$ with author A_n influences A_m is 0.67 which is less, as in the co-authored papers L is at number 2, while N is number three in one paper and number four in other paper. They both (L, N) have same number of papers and co-authored papers as well as their seniors have same number of papers but L was able to more influence his senior researcher due to more contribution in the work he has co-authored.

3.2.2 Dynamic Publication Venue Based StarRank (DPV StarRank)

In this section dynamic publication venue based StarRank score is calculated using entropy. We have calculated entropy of publication venue. The entropy is a measure of disorder in Physics and less disorder means the systems is better. The same phenomenon is workable here in this work as high level venues has low entropy while the

non high level venues will have higher entropy. Here, it is necessary to mention that using entropy enables us to calculate venues influence for the existing and new coming venues.

In case we use existing online list of venues levels there can be different problems. Such as (1) the list may not or usually do not contains all venues and (2) new coming venues will be added later or may be missed on some occasions. Entropy of venues is calculated by using the following standard equation in which w_i is the probability of $word_i$ in a venue *v*. The title words of papers published in a venue are used for calculating the entropy of venues. The lesser the entropy is better as it corroborates that the venue has less disorder due to only accepting papers on its specific areas mentioned in call for paper page.

$$Entropy\ (v) = \_\sum_{i=1}^{m} w_i log_2(w_i) \tag{6}$$

The publication quality score for each researcher is calculated by using the following equation.

$$\lambda(dpq_i) = \frac{1}{|p|} * \sum_{i=1}^{p} \frac{1}{\propto \text{Entropy of Venue}} \tag{7}$$

where, $\lambda(dpq_i)$ is publication quality score of authorA_i all publications the larger the better,entropy of venue is publication rank of the paper and $\propto$ value is between (o<∝<1) which is damping factor so low rank publications can have low scores.

3.2.3 Composite StarRank

In this section composite StarRank method is provided which calculates the rank of author according to author contribution based mutual influence and dynamic publication venue based scores.

$$StarRank(p_i) = \frac{1-d}{n} + d * \sum_{j=1}^{|a|} \frac{ACW(p_i,p_j)*\lambda(dpq)*starRank(p_j)}{\sum_{k=1}^{|v|} ACW(p_k,p_j)*\lambda(dpq)} \tag{8}$$

where, $StarRank(p_i)$ is hybridized score for authors with the higher the score the author is considered rising star.

4 Experiments

4.1 Dataset

The dataset is taken from Digital Bibliography and Library Project DBLP [21]. The data of 1996-2000 is used to predict rising stars. The title of paper, author name and conference/journal where papers have been published are considered as data variables. Stop words are removed and lower casing is performed as standard text preprocessing steps.

4.2 Performance Measurement

The ground truth ranking of rising stars is not available. The authors ranked top using StarRank and their standings are checked later in 2012 to verify if they have realized their predicted potentials or not for the performance evaluation of our proposed methods. The number of papers and citations of papers are averaged for top ranked authors for all methods using arnetminer [2]. If an author have high number citations for his papers he is usually considered better, while high number of papers can be useful but results and discussions explains that even an author A having less number of papers than author B can have more number of citations due to the high quality of his work. PubRank [20] which determines the rising stars from academic social network is used as existing method.

4.3 Parameter Settings

The value of alpha used in our experiments is 0.5 and damping factor value is 0.85. The detailed study of selecting these values for finding rising stars is provided in the sections 4.4.3 and 4.4.4.

4.4 Results and Discussions

4.4.1 Rising Stars

Top ten rising stars found using composite StarRank are shown in Table 2. We have visited their web pages by searching their names in Google. One can see that all of them received at least 3937 or higher number of citations for their published papers. All the top ranked authors are working on key posts now in famous IT companies as well as top ranked universities.

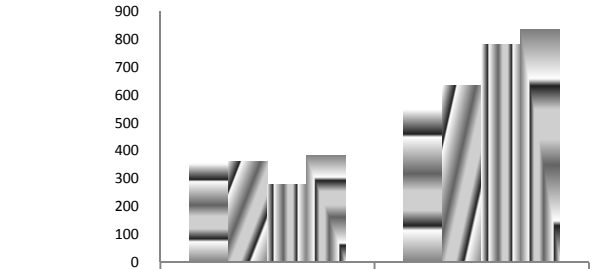
Table 2. Top Ten Predicted Rising Stars from StarRank

Author	Position	Citation
Wei Ying Ma	Principal Researcher, Research Area Manager, Microsoft Research Asia	14355
Philip S. yu	Professor and Wexler Chair in Information Technology, Department of Computer Science, University of Illinois Chicago	28429
Jiawei Han	Professor, Department of Computer Science, University of Illinois at Urbana-Champaign	46654
Zheng Chen	Senior Researcher , Microsoft Research Asia	3937
Divesh Srivastava	AT&T Labs, Inc.	11520
Wei Wang	Professor, University of North Carolina at Chapel Hill	9873
Hsinchun Chen	Professor and Director, Management Information Systems Department Eller College of Management The University of Arizona	8161
Erik d. Demaine	Associate Professor, Massachusetts Inst. Tech., Lab. for Computer Science	7361
Bertram ludaumlscher	Professor of Computer Science, Computer Science Department Stanford University	29544
Lee Tan	Provost's Chair Professor, School of Computing	6824

4.4.2 Comparative Study

Top ten authors are found for all the methods. The average papers and citations of top ten authors are shown in Figure 1. For PubRank [20] the average number of papers and citations for top ten authors are 353.2 and 546.5 which are less in number as compared to our proposed three methods except for average number of papers of top

ten authors for DPV StarRank which is not important. One can see that even the average number of papers for DPV StarRank is less as compared to existing PubRank [20] and one of our proposed method AC StarRank but DPV StarRank has more number of citations for top ten authors which is usually used criterion for evaluating the quality of research work published. This shows that even papers are less in number but if they are published in high level venues they are cited more which shows their popularity. It is clear from the Figure 1 that our proposed methods outperformed existing PubRank method in terms of average citation count for top ten authors.



	Papers	Citations
PubRank	353.2	546.5
AC StarRank	362	632.55
DPV StarRank	278.3	780.45
Composite StarRank	385.2	833.25

Fig. 1. Overall performance comparison

4.4.3 Effect of Alpha Parameter

Alpha is commonly used to measure the internal consistency or reliability of a psychometric test score and can take value between 0 or 1. We always observe in terms of type I errors alpha, which are always small (0.1, 0.05, .01). The smaller alpha value gets the firm proof that the alternative is correct, because the probability of type I error is reduced, but in some case high value of alpha causes high variance [16]. We calculate the rank of author using the different values of alpha 0.1, 0.2, 0.3, and up to 0.9 shown in Figure 2. When we set the alpha value 0.2 in all methods little bit change is observed in author rank, on value 0.3 author rank score is also increased a bit. For 0.4 and 0.5 value of alpha, there is maximum number of average citations and are stable. Consequently, the value of alpha used by us for all methods is 0.5.

4.4.4 Effect of Damping Factor

Damping factor value of 0.85 is usually used for ranking pages on Web [3]. Google itself uses this value because it is easy to get refined results. High damping factor means low dampened and PageRank grows higher. The StarRank is calculated for different values of damping factor to see its effect on rising star finding in terms of average citations of top ten authors. The citations of authors gradually increases from lower to high value of damping factor and one can see from Figure 3 that for 0.7, 0.8 and 0.85 maximum average citations are gained. In this paper we also used damping

factor value of 0.85 though one can use 0.7 and 0.8 too. We study the effect of damping factor on average citations only as they are most important factor to judge the quality of research. As average number of papers is not a very important factor which can be considered to judge the quality of research for an author.

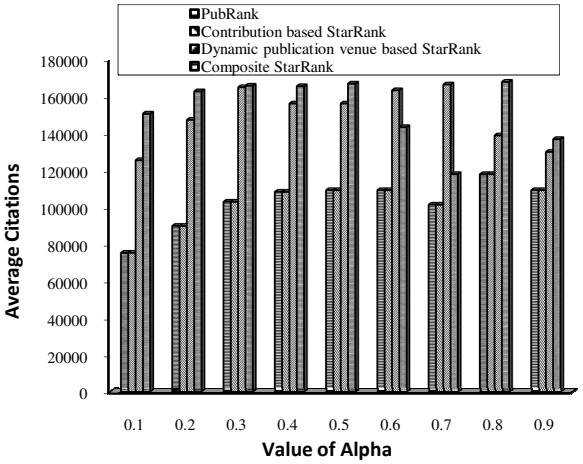


Fig. 2. Average citation on different values of alpha

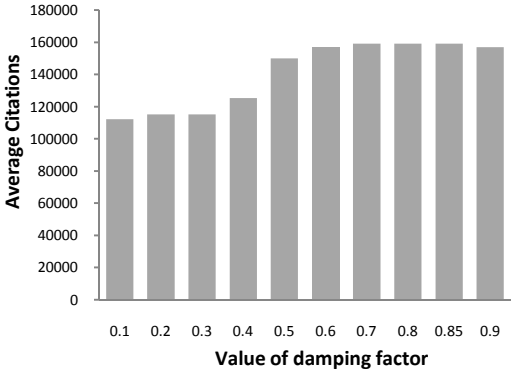


Fig. 3. Effect of damping factor in terms of Average Citations

5 Conclusions and Future Directions

This paper addressed the problem of finding rising stars based on authors' contribution oriented mutual influence and dynamic publication venue scores. From the results we can conclude that mutual influence based author contributions are important and helped in improved results for AC StarRank method. We can also conclude that when

dynamic publication venue scores are used instead of static publication venues scores, improved rising star finding results are obtained. Results have shown that dynamic publication venue scores are more useful as compared to author contribution oriented mutual influence though when both of them are hybridized more improved results are obtained. One can say that workshops are more typical venues but papers rejected from conferences are usually presented in them or incomplete works are presented in them for improvement. In future we plan to consider discriminative model for predicting rising stars as they are recently used to better predict experts for futures [10].

Acknowledgments. The work is supported by Higher Education Commission (HEC), Islamabad, Pakistan.

References

1. Adamic, L., Adar, E.: How to Search a Social Network. Social Networks 27, 187–203 (2005)
2. ArnetMiner, Academic Researcher Social Network Search, http://www.arnetminer.org/
3. Boldi, B.P., Santini, M., Sebastiano, V.: PageRank as a Function of the Damping Factor. In: Proc. of the 14th International Conference on World Wide Web, pp. 557–566 (2005)
4. Brin, S., Page, L.: The Anatomy of a Large-Scale Hyper Textual Web Search Engine. In: Proc. of International Conference on World Wide Web, pp. 107–117 (1998)
5. Sekercioglu, C.H.: Quantifying Coauthor Contributions. Science 322(5900), 416–417 (2008)
6. Daud, A.: Using Time Topic Modeling for Semantics-Based Dynamic Research Interest Finding. Knowledge-Based Systems (KBS) 26, 154–163 (2012)
7. Daud, A., Li, J., Zhou, L., Muhammad, F.: Temporal Expert Finding through Generalized Time Topic Modeling. Knowledge-Based Systems (KBS) 23(6), 615–625 (2010)
8. Daud, A., Shaikh, M.A., Rajpar, A.H.: Scientific Reference Mining using Semantic Information through Topic Modeling. Mehran University Research Journal of Engineering and Technology 28(2), 253–262 (2009)
9. Daud, A., Muhammad, F.: Group Topic Modeling for Academic Knowledge Discovery. Journal of Applied Intelligence 36(4), 876–886 (2012)
10. Fang, Y., Si, L., Mathur, A.P.: Discriminative Models of Integrating Document Evidence and Document-Candidate Associations for Expert Search. In: Proc. of SIGIR (2010)
11. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information Retrieval in Folksonomies: Search and Ranking. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 411–426. Springer, Heidelberg (2006)
12. Ivn, G., Grolmusz, V.: When the Web Meets the Cell: Using Personalized PageRank for Analyzing Protein Interaction Networks. Bioinformatics 27(3), 405–407 (2011)
13. Liu, X., Bollen, J., Nelson, M.L., Sompel, H.V.D.: Co-authorship Networks in the Digital Library Research Community. Information Processing and Management 41(6), 1462–1480 (2005)
14. Long, P.M., Lee, T.K., Jaffar, J.: Benchmarking Research Performance in Department of Computer Science, School of Computing, National University of Singapore (1999), http://www.comp.nus.edu.sg/~tankl/bench.html

15. Mihalcea, R., Tarau, P.: TextRank: Bringing order into Text. In: Proc. of the International Conference on Empirical Methods in Natural Language Processing, EMNLP, pp. 404–411 (2004)
16. Mohsen, T., Reg, D.: Making sense of Cronbach's alpha. International Journal of Medical Education 2, 245–246 (2011)
17. Shu, L., Long, B., Meng, W.: A Latent Topic Model for Complete Entity Resolution. In: Proc. of the International Conference on Data Engineering, ICDE (2009)
18. Tang, J., Sun, J., Wang, C., et al.: Social Influence Analysis in Large-scale Networks. In: Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD, pp. 807–816 (2009)
19. Wang, C., Han, J., Jia, Y., Zhang, D., Yu, Y., Tang, J., Guo, J.: Mining Advisor Advisee Relationships from Research Publication Networks. In: Proc. of 16th International Conference on Knowledge Discovery and Data Mining, SIGKDD (2010)
20. Li, X.-L., Foo, C.S., Tew, K.L., Ng, S.-K.: Searching for Rising Stars in Bibliography Networks. In: Zhou, X., Yokota, H., Deng, K., Liu, Q. (eds.) DASFAA 2009. LNCS, vol. 5463, pp. 288–292. Springer, Heidelberg (2009)
21. DBLP bibliography database, `http://www.informatik.uni-trier.de/~ley/db/`

Expertise Ranking of Users in QA Community

Yuanzhe Cai and Sharma Chakravarthy

IT Laboratory & Department of Computer Science and Engineering
The University of Texas at Arlington, Arlington, TX 76019
yuanzhe.cai@mavs.uta.edu, sharma@cse.uta.edu

Abstract. Community Question Answering services (CQAs) have become ubiquitous, and are widely used. Hence, it would be beneficial if we can mine useful inferences from these data sets to improve these services. For example, if we can infer or identify expertise of users' from these data sets, we can route questions to the right people. With the identification of expertise, number of experts needed to cover a set of topics (in a CQA service) can also be optimized. This paper addresses the problem of inferring expertise.

Current approaches infer expertise using traditional link-based methods such as PageRank or HITS, and others (e.g., number of answers given by a user or Z_score). Although an ask-answer graph can be generated for a CQA data set based on the ask-answer paradigm (who answers whose questions), this graph is different, in its semantics, from the web graphs. We posit that both graph structure and domain information related to an answerer (e.g., answer quality) is critical for inferring the expertise of users. Based on this observation, we propose the *ExpertRank* framework to compute users' expertise. We establish that the information used has a bearing on the accuracy of results. We present our algorithm along with extensive experimental analysis that indicates superiority of our approach as compared to other link-based methods.

1 Introduction

Community Question Answering services (CQAs) strive to provide users with meaningful information using the *ask-answer* paradigm. These communities allow users to post questions and other users to answer them. When a user posts a question using a CQA service, different approaches are used to find appropriate users to answer the question. Current communities mainly use the following approaches.

Questioner-Based Approach: In this approach, the user who asks the question is responsible for choosing an appropriate expert to answer his/her question. For example, AllExpert (`http://www.allexpert.com/`) provides a number of statistics for each expert in their list including number of questions answered, publications, awards received, and honors in relevant areas. After a questioner browses this information, s/he can direct his/her question to one of the experts.

Answerer-Based Approach: This approach allows answerers to select questions that are of interest and answer them. Users post their questions to the community as a whole and answerers choose the ones to answer. This method is

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 25–40, 2013.

mainly used in many different CQAs, such as Yahoo! Answers (`http://answers.yahoo.com`), Stack Overflow (`http://stackoverflow.com/`), etc.

However, the above-mentioned approaches have a number of drawbacks. In the questioner-based approach, there are a large number of users and hence it is impractical to expect questioners to find an expert by browsing users' profiles. Although the other approach encourages various users to answer questions, this method ignores answerer's quality. If it is possible to assess the expertise of users in an acceptable manner, one can identify a small set of users to answer a question. The questioner will not only be relieved of this burden but is also likely to receive better answers. This will also result in lesser number of message exchanges.

Although both information retrieval [12] and (extended) link-based methods [20,9] have been used for discovering experts from CQAs, they do not seem to be appropriate for this problem. Similarity score does not represent quality. Link-based methods assume transitivity (if B can answer A's question and C can answer B's question, C's expertise rank is boosted because C is able to answer a question of someone who has some expertise). This assumption may not be valid when questions are asked on diverse topics and there is considerable overlap among the users answering questions. Another, more important and critical, observation is that the link-based methods do not use the quality of contents associated with a link. We strongly believe that not using contents is not an option for QA services as the very notion of an expert depends upon the quality of answers given by that user.

Consider the following short example to illustrate the above observations. Table 1 shows a few questions and some of their answers from the Stack Overflow service for "C" language. Figure 1 shows the ask-answer graph for Table 1 using the ask-answer paradigm. A directed edge is drawn from user u_1 to user u_2 if user u_2 answered one or more questions of u_1. Table 2 shows the ranking result for this graph using the PageRank (PR) algorithm where user B and D have the same PageRank score (expertise score), but user B's expertise should be higher than user D as reflected by the voted score [1].

Table 1. A Sample from Stack Overflow

User	Votes	Content
Questioner A		In C arrays why is this true? a[5] == 5[a]
Answerer B	330	Because a[5] will evaluate to: *(a + 5) and 5[a] will evaluate to: *(5 + a)
Answerer D	-6	You can search the result on the Google.
Questioner B		What is the best tool for creating an Excel Spreadsheet with C#?
Answerer C	144	You can use a library called Excel Library. It's a free, open source library posted on Google Code.
Questioner E		How can Inheritance be modeled using C?
Answerer A	0	See also: http://stackoverflow.com/questions/351733/can-you-write-object-oriented-code-in-c

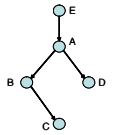


Fig. 1. ask-answer graph

Table 2. PR score

Node	PageRank
A	0.21
B	**0.20**
C	0.28
D	**0.20**
E	0.11

[1] In CQAs, the voted score of an answer is the *sum* of all votes received for that answer. A user can give a +1 for a good answer and a -1 otherwise.

Different service-specific information can be extracted from CQAs to identify answer quality. The voting information (shown in Table 1) is one piece of information. Another feature that represents answer quality can be "the length of answer" used in [1]. In addition, we can also use question/answer similarity score to approximate the answer quality of a user.

Contributions:

- We propose a framework using which CQA service-specific information that represents quality can be incorporated into the ask-answer graph. We analyze domain information from several data sets and indicate what can be beneficially used.
- We present the *ExpertRank* approach which is based on the Katz index algorithm [10] to measure users' expertise and rank them. A tunable parameter α (attenuation factor in Katz index) is used to control the transitivity aspect of the ask-answer graph.
- Extensive experimental analysis is performed on multiple, diverse data sets to show how the proposed algorithm and the domain information provide more accurate results than traditional link-based and other approaches.

Road Map: Section 2 presents related work. Section 3 defines the problem and motivates our approach. Section 4 describes our contributions along with the ExpertRank algorithm and alternative ways of using domain information. Section 5 shows extensive experimental results on three diverse data sets and their analysis. Section 6 has conclusions.

2 Related Work

Some IR techniques (e.g., similarity score) have been applied to CQA data sets to identify users' expertise. Efforts by [2,12] build a term-based expertise profile for each user and rank expertise based on the relevance scores of their profiles for a question by using traditional IR models. The notion of quality is not captured by similarity. Zhang et al. [20] propose four methods to identify expertise (or rank users in expertise order): number of answers given by a user (#Answers), Z_score measure, PageRank, and HITS. Note that none of these methods use the contents of questions or answers. The first two does not really capture quality as they are not based on content but only on the number of questions and answers. Traditional PageRank [15] and HITS [11] are also used again without considering answer quality and hence the accuracy of these link-based algorithms is not good. Campbell et al. [4] and Dom et al. [5] use the ask-answer paradigm for the exchange of emails in a group/community. They use HITS and in-degree methods, respectively, to rank users' expertise. They apply link-based algorithms to both a synthetic graph and a small email graph to rank correspondents according to their out-degree of expertise on subjects of interest. Again quality of answers is not taken into consideration. Rode et al. [17] also use traditional link-based rank approaches to rank entities and model them as a graph-based relevance propagation framework, but their work focuses on ranking entities (and not users) and on the XML/Web service system, not in CQAs.

Liu et al. [13] use TrueSkill [6] and SVM models [14] to rank users' expertise score using competition-based approach. They assume that given a question, its best answerer b has a higher expertise level than its asker a and other answerers. Thus, they extract pairwise comparison for each question as the training data set using $|A|$ as the number of answers for each question. They show that their approach is better than the link-based approaches. Our work is different in that we are extending the link-based approach with different types of quality information to rank the expertise of users. Our future work includes using the same data set to compare these two alternatives.

Other work on expertise analysis is not directly related to ours. For example, Pal et al. [16] extract six features from CQAs by considering the user's motivation and ability to help other users and build learning models, such as SVM and DTree, to predict *potential* experts. We want to use quality information rather than motivation and ability to help. Others, such as [21], also use the link mining approach for ranking experts in the paper-author network, but out work is different in that we focus on the question-answer networks and ask-answer graphs.

3 Problem Statement

This paper focuses on the general problem of mining expertise of users' from a CQA data set and rank them. As there are many paradigms used in CQA services, different types of domain (or service-specific) information are available. This paper evaluates the effectiveness and utility of quality as domain information for the purpose of ranking.

Given a CQA data set consisting of users $u_1, u_2, u_3, ..., u_n$ along with questions, answers, and available domain (or service-specific) information (e.g., votes), our goal is to infer users' expertise and rank them. Ranking of users will facilitate: i) providing a list of ranked experts for a topic category, ii) directing questions to a small set of experts, and iii) CQA services to optimize on the number of experts needed. We would like to point out that expertise rank order is quite subjective and can vary with respect to the evaluator. Furthermore, no real expertise rank exists in the CQA data set itself. Thus, it is really difficult to find a standard (or a baseline) to compare the evaluated rank order. As a result, one has to resort to manual evaluation as is commonly done by researchers on this topic [20,4].

3.1 Motivation for Our Approach

Traditional web page graphs capture citation or reference relationships. A web designer who is creating a page is likely to choose the best page (according to him/her) as a reference from that page. Hence, in a web reference graph, the number of times a web page is referenced can be used to evaluate the web page's quality. The same intuition is not true for an ask-answer graph as *any user*can answer any question and there is *no guarantee on quality!*

Characteristics of Ask-Answer Graphs: In contrast, an ask-answer graph does not have the same intuition as the web reference graph. First, in CQAs,

any answerer, no matter good or not, can give an answer to a question. Thus, in an ask-answer graph, questioners, typically, cannot choose the best answerer. In Figure 1, recall that the direction of the edge is from the questioner to the answerer. Since, in an ask-answer graph neither asker nor answerer ensure the quality of the links, we cannot directly apply these linked-based rank algorithms to this ask-answer graph. Thus, we propose approaches that include quality aspect into an ask-answer graph using domain information. We propose four different approaches to include quality information for links in an ask-answer graph using domain specific information in CQAs. Second, the ask-answer relationship is also different from the web page reference relationship. Using a URL, we can directly extract web page reference relationship. However, we cannot directly extract the ask-answer relationship from the QA data set, because between two users there may be a number of ask-answer relationships (each with varying quality). Therefore, combining/aggregating these relationships together to describe the ask-answer relationship becomes an important problem. Based on our observation, we believe that for an ask-answer graph we cannot directly use traditional link-based ranking algorithm to identify users' expertise.

4 ExpertRank Framework

Based on our earlier observations, the ask-answer graph derived from a CQA data set is enhanced. First, a weight is associated with each edge (or link) using domain information that reflects quality of answers corresponding to that edge. As an edge in an ask-answer graph is likely to reflect more than one answer given by a user, it is not sufficient to use the number of answers given by a user as it can be misleading. Qualitative value of all the answers need to be aggregated to reflect the edge semantics. Otherwise, some spam users who give *bad or irrelevant* answers will reach a high authority score. Since an edge from user u_i to user u_j in an ask-answer graph indicates that user u_j answered one or more questions of user u_i, the weight of the edge between users u_i and u_j takes into account: the quality of user u_j's each answer for u_i's question, and the fraction of u_i's questions answered by u_j. Second, the transitivity relationship used in most of the link-based approaches also need to be adjusted for each data set. We want to derive this transitivity relationship from the data set itself as this is likely to be specific to a data set. In this paper, we adjust the variable α (called the attenuation factor in the Katz index) to reflect this transitive relationship. We believe that this is important for these applications.

In the following sections, we discuss the use of different types of domain (or service-specific) information, their relevance, and the computation of edge weights for an ask-answer graph.

4.1 Voted Score Approach (VS)

A voted score (available in many CQA services such as Yahoo! Answers) for an answer is the cumulative score of that answer as judged by users in the community. A score, given by a user for an answer, is either a +1, or -1 reflecting positive, or negative answer quality. A voted score is meant to reflect the quality

of an answer. The voted score seems to be a better indication of the quality of an answer (especially if a good number of high quality users (In Stack Overflow only those user whose reputation is more than 15 can vote for an answer.) take part in the voting process as shown in Table 1).

We compute the answer score as

$$A\_Score(u_i, u_j, q_k) = VS(u_i, u_j, q_k)$$

where $A\_Score(u_i, u_j, q_k)$ is the answer score for u_j's answer to u_i's question q_k and $VS(u_i, u_j, q_k)$ is the voted score of u_j's answer for u_i's question q_k. We still need to combine quality scores from different answers given by the same user. Since this step is common to all the approaches it is discussed in Section 4.5 after describing the approaches.

4.2 Ranked Answer Approach (RA)

We need an approach for services that may not have a voted score or where the voted score may not be reliable. Many services mentioned earlier do not have a special voting phase. In order to overcome the limitations of voted score as a quality measure and have a safer alternative for arriving at the answer quality, we use an approach for ranking answers for quality as described below[2].

We use a method along the lines proposed in Shah et al. [19] for obtaining answer quality by extracting a set of features from the data set. Specifically, we use the approach proposed in [3] for this purpose. This approach uses a learning to rank approach (RankSVM [8]) and temporal features which have been shown to [3] estimate answer quality better. We use the score obtained for each answer as its quality score.

$$A\_Score(u_i, u_j, q_k) = RankSVM(u_i, u_j, q_k)$$

where $RankSVM(u_i, u_j, q_k)$ describes the ranking score of u_j's answer for u_i's question q_k. The accuracy of ranked answer approach is mainly decided by the training data set and the features used. We will discuss this in section 5.

4.3 Information Retrieval Approach (Sim)

Traditionally, cosine similarity value is used in document retrieval and search. Although we believe that this is not a very good measure for answer quality, we use this to demonstrate the effect of answer content information for CQA data sets. If the similarity between an answer and a question is low, we interpret this answer having low quality since this answer is unrelated to the question. For example, since in Table 1 user D gives a "bad" answer for user A's question, the similarity score between D's answer and A's question is 0. After removing stop words from a question and its answers in CQAs and stemming them, we build a question vector (a term vector extracted from each question) and an answer vector (a term vector extracted from each answer for that question) and then use VSM [18] to calculate cosine similarity between question and answer.

[2] Note that any approach for ranking quality of answers can be used here; we are using one that we are familiar with.

4.4 Hybrid Approach (Hyb)

In order to improve accuracy, it is possible to combine multiple answer quality scores into a composite score. Since we believe that the similarity score is not as good as the others for measuring quality (see [3]), we propose to combine the voted score with the ranked score. We use a parameter γ to adjust the contribution of these two scores. We use the following formula to calculate the hybrid answer quality score. In Section 5, we discuss the choice of γ for real data sets.

$$A\_Score(u_i, u_j, q_k) = \gamma VS(u_i, u_j, q_k) + (1 - \gamma) RankSVM(u_i, u_j, q_k)$$

4.5 Computing Edge Weights

Since voted score which is an aggregate varies significantly from answer to answer, it needs to be normalized. We normalize the voted score using the minimum and maximum values of the scores for *each* answer as follows:

$$minA\_Score(u_i, q_k) = min\{A\_Score(u_i, u_j, q_k) | j = 1, ..., n\}$$
$$maxA\_Score(u_i, q_k) = max\{A\_Score(u_i, u_j, q_k) | j = 1, ..., n\}$$

and the normalized A_Score is computed as

$$NA\_Score(u_i, u_j, q_k) = \frac{\frac{A\_Score(u_i,u_j,q_k) - minA\_Score(u_i,q_k)}{maxA\_Score(u_i,q_k) - minA\_Score(u_i,q_k)} + \epsilon}{1 + \epsilon}$$

which is in the range $[\frac{\epsilon}{1+\epsilon}, 1]$. ϵ is used to adjust the lower bound of normalized values. We do not want to set the normalized value for the answer receiving the lowest voted score to 0 because these answers have been assessed for quality as opposed to answers that have not been voted upon (which receive a normalized score of 0). Thus, if an answer receives the lowest voted score, its normalized quality score is equal to $\frac{\epsilon}{1+\epsilon}$. In our experiments, ϵ is set to 0.1. ϵ value greater than 0.1 or 0.15 does not make sense as it is a compensatory value. We use the average score of all answers as the answer quality score between two users. That is,

$$A\_Quality(u_i, u_j) = \frac{\sum_{j=1}^{|QA(u_i,u_j)|} NA\_Score(u_i, u_j, q_k)}{|QA(u_i, u_j)|}$$

where $A\_Quality(u_i, u_j)$ describes the answer quality (of an edge from u_i to u_j) and $|QA(u_i, u_j)|$ is the number of u_i's questions answered by u_j. Usually, if $A\_Quality(u_i, u_j)$ is high, user u_j has been able to answer user u_i's questions well.

In addition to answer quality, the fraction of u_i's questions answered by u_j captures whether or not user u_j is familiar with user u_i's questions. Hence, u_j's quality of answers need to be tempered by the fraction of u_i's questions answered by u_j and is calculated as

$$Q\_Factor(u_i, u_j) = \frac{|QA(u_i, u_j)|}{|Ques(u_i)|}$$

where $|QA(u_i, u_j)|$ is the number of u_i's questions answered by u_j and $|Ques(u_j)|$ is the total number of user u_i's questions. $Q\_Factor(u_i, u_j)$ is in the range

of $[0, 1]$. The quality of u_j's answers to u_i's questions combines these two factors. Thus, the weight of the edge between u_i and u_j is computed as:

$$QA\_Quality(u_i, u_j) = A\_Quality(u_i, u_j) \times Q\_Factor(u_i, u_j)$$

where $QA\_Quality(u_i, u_j)$ captures the quality of u_j's to answer u_i's questions (i.e., weight of the edge from u_i to u_j). As the edge in this graph captures the quality of answers to question, we term this graph a weighted ask-answer graph. Users and relationships in CQAs are modeled as a directed graph $G = (V, E)$, where a node in V represents a user in this QA community and a directed edge $< u_i, u_j >$ from u_i to u_j indicates that u_j answered one or more of u_i's questions. The weight of the edge $< u_i, u_j >$ captures the quality of u_j's answers u_i's questions. For a user u_i in this graph, we denote $Q(u_i)$ and $A(u_i)$, respectively, as the set of questioners (in-neighbors) and answerers (out-neighbors). The k^{th} questioner for user u_i are denoted as $Q_k(u_i)$, for $1 \leq k \leq |Q(u_i)|$, and individual k^{th} answerers of user u_i are denoted as $A_k(u_i)$, for $1 \leq k \leq |A(u_i)|$.

4.6 ExpertRank Algorithm

The expertise rank of a user u_i ($ER(u_i)$) is the sum of user u_i's quality score with respect to u_i's questioners. Thus, we have our iterative equations to compute $ER(u_i)$ as:

$ER_k(u_i)$ gives the expertise score for user u_i on the k^{th} iteration and we successively compute $ER_{k+1}(*)$ based on $ER_k(*)$. We start with $ER_0(*)$ where each $ER_0(*)$ is equal to 0, which is the lower bound on the actual expertise score $ER(u_i)$:

$$ER_0(u_i) = 0$$

To compute $ER_{k+1}(u_i)$ from $ER_k(*)$, we use the following equation:

$$ER_{k+1}(u_i) = \sum_{j=1}^{|Q(u_i)|} QA\_Quality(Q_j(u_i), u_i) + \alpha \sum_{j=1}^{|Q(u_i)|} ER_k(Q_j(u_i))$$

For iteration $k + 1$, we update user u_i's expertise scores of his/her neighbors from the previous iteration k.

Algorithm 1. *ExpertRank*

Require:
User Weighted Matrix U;
Ensure:
Expertise Rank Vector, ER;
1: $ER_0 \leftarrow 0$;
2: Do $k = 0$ to Max-Iteration K
3: For each element $ER(u_i)$
4: $ER_{k+1}(u_i) = \sum_{j=1}^{|Q(u_i)|} U(Q_j(u_i), u_i) + \alpha \sum_{j=1}^{|Q(u_i)|} ER_k(Q_j(u_i))$;
5: End For
6: Normalize(ER);
7: If ($max(ER_{k+1}(u_i) - ER_k(u_i)) < \varrho$) go to line 9;
8: End Do
9: **return** ER;

Algorithm 1 outlines *ExpertRank* computation. It takes one argument U. In line 1, *ExpertRank* algorithm initializes variables and sets all the user's expertise rank score as 0. Lines 2-8 implements iterative equation to calculate each user's expertise score. Line 6 is used to *normalize* the user's expertise score in each iteration. Lines 2 and 7 are used to stop this iterative algorithm. Although the convergence of iterative expertise rank algorithm can be guaranteed in theory [10], practically a tolerance factor ϱ is used to control the number of iterations performed. It is recommended to set $\varrho = 0.001$, same as the one used in PageRank [15]. The terminating condition of the iterative algorithm is:

$$max(ER_{k+1}(u_i) - ER_k(u_i)) < \varrho$$

The algorithm stops when the maximal change of rate of expertise rank score between two consecutive iterations for all the users is smaller than the threshold ϱ. In all of our experiments we have seen rapid convergence with the relative expertise ranking score stabilizing in 40 iterations. Hence, we have fixed the number of iterations (k) to be 40.

we also analyze the time and space complexity of this algorithm. Because the weighted ask-answer graph is a very sparse matrix, we need to store only the edges for the weighted graph; Therefore, the space required is $O(e)$, where e is the number of edges in this graph. Let d be the average of $|Q(u_i)|$ over all the users u_i. The time complexity of this algorithm is $O(kdn)$, since in each iteration, expertise rank score of u_i is updated with values from this user's questioners. n is the number of nodes in the graph. As d is the average of $|Q(u_i)|$ over all the users u_i, it can be treated as a constant as it is not likely to increase with n.

4.7 Discussion of α

Recall that α describes the transfer probability in the QA community. If α is small, ExpertRank will only consider local information (a smaller scope of graph); if α is large, ExpertRank will consider global information (a larger scope of graph). For example, Figure 2 is a sample ask-answer weighted graph and Table 3 shows ExpertRank's results for different values of α. It is clear from the graph that users d and f answer questions, and e further answers questions of d and f. If the entire graph is considered, e should come out with the highest expertise score followed by d and e. If only local information is used, d and e should come out as equal experts. This translates to the values of α as follows. If α is 0.5, node e receives the highest expertise score. However, if α is 0.05, node d and f receive the highest expertise score. Table 3 also shows ranking results of PageRank and In-Degree. PageRank considers global graph information to rank each node; In-Degree just considers its neighbor nodes to rank the node. In this example, when α is 0.5 (large), ExpertRank has the same rank order as PageRank; when α is 0.05 (small), ExpertRank has the same rank order as In-Degree. This example clearly shows that ExpertRank's results shift from local to global with the increase of α.

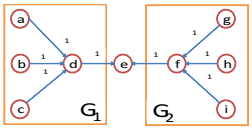


Fig. 2. A Sample Ask-Answer Graph

Table 3. Results of Rank Algorithms for Figure 2

Nodes	ER(0.5)	ER(0.1)	ER(0.05)	PageRank	In-Degree	HITS
a	0.18	0.30	0.32	0.05	0	0
b	0.18	0.30	0.32	0.05	0	0
c	0.18	0.30	0.32	0.05	0	0
d	**0.45**	**0.39**	**0.37**	**0.18**	**3**	**0.61**
e	**0.63**	**0.38**	**0.35**	**0.34**	**2**	**0**
f	**0.45**	**0.39**	**0.37**	**0.18**	**3**	**0.61**
f	0.18	0.30	0.32	0.05	0	0
g	0.18	0.30	0.32	0.05	0	0
h	0.18	0.30	0.32	0.05	0	0

[1] ER(0.5) means that α in ExpertRank is 0.5.
[2] ϱ in all the algorithms is 0.001.

5 Experimental Analysis

We use different data sets to test our approaches (See Table 4).

Stack Overflow (SO) Data Set: This service focuses on computer programming topics. Unlike other traditional QA services, SO allows a user to modify other user's answers. As a result, the average number of answers for each question is only 2.36. We consider the first user who posts the answer as the answerer, because, in most cases, the first user is likely to provide a larger contribution than others. Each question is marked with a topic tag (e.g., "C"). We use questions marked as "C" ("Oracle") as SO-C (SO-O) data set.

Turbo Tax (TT) Data Set: TT service discusses tax-related issues. This community enrolles many experts to answer questions, so most of the users are mainly questioners. Thus, the average number of answers for each question is only about 1.11. We choose questions between 2009.1-2009.4 for our experiments as this community is very active in that period.

Table 4. Data set Characteristics

Data set	#Questions	#Answers	Avg (V)	#Qs (1A)
SO-C	25,942	91,615	2.6	5,576
SO-O	8,644	21,879	1.6	2,811
TT	232,411	257,113	1.3	215,163

[1] Avg (V) is the average number of votes for each answer.
[2] #Qs (1A) is #questions having 1 answer.

Table 5. Characteristics of 50 Random Users

Data set	#Questions	#Answers
SO-C	134	1,492
SO-O	246	2,408
TT	17	23,453

5.1 Evaluation Method

For these studies, baseline or a standard with which to compare results is extremely important. For some scenarios it is easy to find or derive a baseline. Similarity is one such standard that is widely used in information retrieval. We believe that similarity is not a very good measure for our problem. Our problem for finding a standard for comparison is exacerbated by the fact that the notion of expertise itself can be quite subjective. Hence our choice of data sets to areas where that issue is minimized. There is no user expertise rank information in the

data sets nor can it be derived from the data sets. Hence, as has been done by other researchers (e.g, [20]), we have used human experts to manually evaluate the expertise of users in each data set. Due to the large number of users (as can be see from Table 4) it is impossible to manually rate all users in the data set. Hence, we randomly choose 50 users in each data for manual evaluation. We only choose users who have answered at least 10 questions to ensure enough content for manual evaluation. Five levels of expertise (again commonly used in these studies) as shown in Table 5 were used.

Table 6. Five Levels of expertise rating

Level	Meaning	Description
5	Top Expert	Knows core theory and advanced topics
4	Professional Expert	Can answer most questions and knows one or more sub topics well.
3	General Expert	Knows some advanced concepts in these topics.
2	Learner	Knows some basic concepts
1	New Recruit	Just starting to learn these topics

We have used two independent experts who are very familiar with the "C language" and the "Oracle database" from the computer science department to evaluate the two SO data sets. We have used two experts from the business department to evaluate the "Turbo Tax" data set. *None of these experts take part or are associated with this research.* After each expert evaluated the data sets independently, for sanity check, we used Kendall's τ [7] score to compare these two users' rank lists. The Kendall's τ distance between two raters is 0.741 for SO-C, 0.761 for SO-Oracle, and 0.711 for TT. In order to maintain consistency of evaluation, we remove users from our evaluation whose score differs by more than 1 level in Table 6. Based on this criteria, we have removed 3 users from the SO-C, 2 users from SO-O, and 5 users from TT data set. After this, the Kendall's τ has improved to 0.793 for SO-C, 0.783 for SO-O), and 0.788 for TT. As we add the rating of two raters, there are a total of 10 categories.

The metric used for comparison is also important. In the information retrieval area, researchers use a number of measures to evaluate the rank list's accuracy; one of them is the DCG (Discounted Cumulative Gain) score [7]. Intuitively, the DCG score evaluation method penalizes experts with a higher rank if they appear lower in the list. Hence, this evaluation metric matches well with our application requirement[3]. Since DCG score is not between 0 and 1, we use the Normalized DCG (or NDCG) [7] to evaluate the ranked list. If NDCG@n is large, this algorithm's rank order will match well with the standard; If NDCG@n is small, this algorithm's rank order does not match well with the standard.

Methods for Comparison: In the literature, four methods have been used for predicting the expertise of users. In this paper, we have proposed four ExpertRank-based approaches – ER (VS), ER (RA), ER (Sim), and ER (Hyb) – for doing the same. Our analysis will compare these eight methods for accuracy:

– HITS: Jurczyk et al. [9] use the HITS authority score to identify users' expertise.

[3] Kendall's τ is a measure for the list where as NDCG can be calculated for various positions.

- PageRank: Zhang et al. [20] use the PageRank score as the expertise score. In our experiments, the parameter d (the damping factor) is 0.85 (the highest accuracy).
- #Answers: Zhang et al. [20] use the number of questions answered (or number of answers) as users' expertise score.
- Z_Score: Considering both the number of questions and answers, Zhang et al. [20] use Z_score to identify users' expertise.
- Four approaches in this paper: ER (VS), ER (RA), ER (Sim), and ER (Hyb).

Intuitive Analysis: Our premise is that there is a need for quality information in addition to structure to predict expertise. We expect both ER (VS) and ER (RA) approaches to do better than methods that do not use the above information (PageRank and HITS). Z_score and #Answers have been shown to be better than the link-based methods in [20]. Our experiments indicate the same. Intuitively, we do not expect the similarity approach to do better than any of our approaches as merely the similarity information is not sufficient to infer answer quality. Finally, we expect the ER (Hyb) to do much better than all others as it combines two different and meaningful quality information.

5.2 Experimental Results

Parameter α affects the accuracy of ExpertRank score directly. This parameter is application dependent. In order to study the effect of this parameter, for the SO-C data set, we compared accuracy with the standard by changing the α value and found as expected that the accuracy is high for the α value of 0.1. We have used this value for all data sets. This also indicates that although we choose the questions from a special domain, the transitivity of the ask-answer graph is low. In other words, questions for each user cover a large scope. Eventually, this parameter need to be tied to the characteristics of the data set.

Analysis of ER (RA): The accuracy of ER (RA) is mainly decided by the training data set and the list of features used. In order to study the effect of the training data set[4], we chose the training data set in two ways. The first choice included 1000 questions each of which has more than 5 answers. In the second alternative, we randomly chose 1000 questions without imposing any constraints. We show the results of these alternatives as ER (RA-5A) and ER (RA-Ran) respectively. Figures 3a, 3b, and 3c show accuracy results of ER (RA-5A) and ER (RA-Ran). Intuitively, more answers provide better quality values and hence should perform better than random selection. Indeed ER (RA-5A) shows better NDCG curve as compared to ER (RA-Ran) method for all the data sets. The reason can be explained as follows. Some of these CQAs data sets have a lot of questions that have only one answer (see Table 4). RankSVM cannot build a very good ranking model for these questions because the rank is the comparison for at least two elements. Because ER (RA-Ran) randomly chooses questions from the CQAs, training data set will contains a lot of questions which have

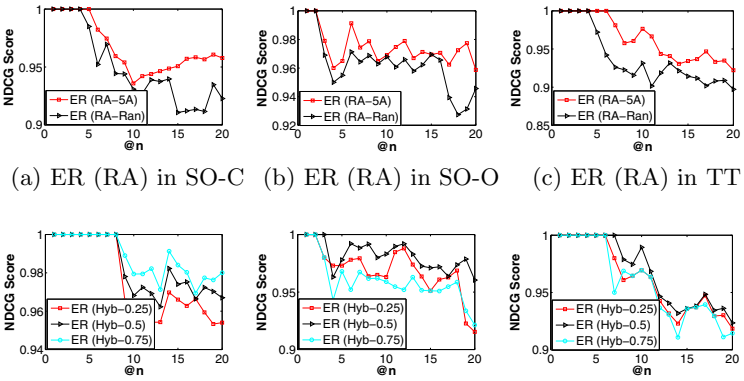
[4] Due to space constraints, we are not discussing alternate choices for features in this paper.

only one answer; Therefore, the ER (RA-Ran) model receives a lower accuracy than ER (RA-5A). Moreover, Figures3a, 3b, and 3c clearly indicate that in the TT data set the accuracy improvement from ER (RA-5A) to ER (RA-Ran) is much higher than the other two data sets. This is because in the TT data set more than 90% of questions has one answer. As ER (RA-5A) is better in general, we only compare ER (RA-5A) with the other algorithms.

Parameter γ in ER (Hyb): ER (Hyb) uses γ to combine ER (VS) and ER (RA). Figures 3d, 3e, and 3f show the accuracy for different γ values (0.25, 0.5, and 0.75). When $\gamma = 0.75$, ER (Hyb) reaches the highest accuracy in SO-C; When $\gamma = 0.5$, both SO-O and TT data sets ER (Hyb) reach the highest accuracy. To understand this, take a look at the average number of votes for each answer in the SO-C data set which is 2.6. In contrast, in SO-O and TT data sets the average number of votes for each answer is 1.6 and 1.3 respectively. (see Table 4). More voting results in a better quality assessment and hence weighing it higher is more useful for the hybrid approach. Based on this analysis, γ can be chosen based on the characteristics of the data set which are easy to compute. A similar analysis can be performed for the accuracy of the features and the training data set to further determine the value of γ in ER (Hyb).

Accuracy Analysis: In these experiments, an NDCG score of 1 is desirable for as many values of n as possible. A score of 1 up to an n indicates that the predicted expertise matches the baseline completely up to n experts. Hence, the goal is to improve the n value for which the NDCG score is 1. Even improvements by a small n is significant when identifying top k experts. For example, extending the NDCG score of 1 from @2 to @3 is an improvement of 50%. We have obtained an improvement of 40% for the TT data set and 160% for the SO-C data set. For the SO-O data set, our approach has achieved an NDCG score of 1 for @3 as compared to none of the earlier approaches even reaching an NDCG score of 1 for any n.

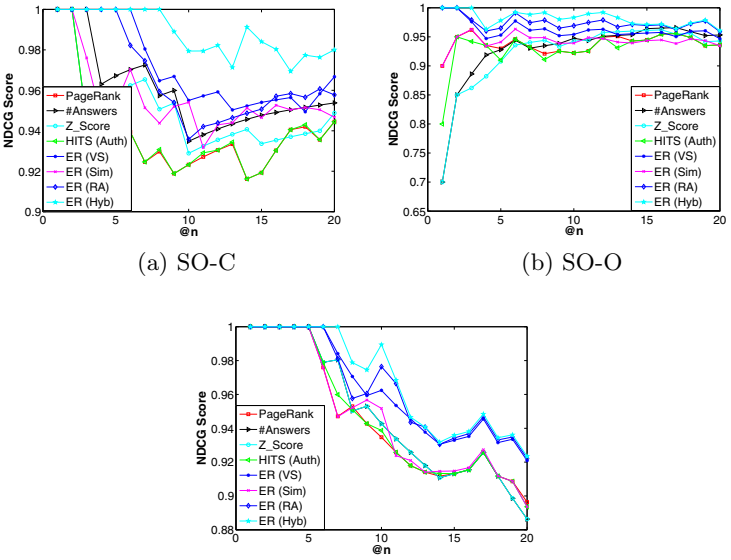
Figures 4a, 4b, and 4c show the comparison of 8 approaches for all three data sets. Our experimental results clearly indicate: (i) in general, our approaches (ER (Sim), ER (VS), ER (RA), ER (Hyb)) have better NDCG scores (for more n) as compared to the other four methods proposed in the literature. We believe this is due to the inclusion of answer quality and α, (ii) #Answers and Z_Score methods have better NDCG curves than PageRank and HITS. The reason can be explained as the transitivity relationship in the QA community is much weaker than web page graphs, (iii) Z_score reaches similar accuracy as #Answers. In the QA community a user who answers a lot of questions is likely to ask few questions. In our test data sets these 50 users answer more than 10 questions and they ask very few questions (See Table 5). Thus, Z_score and #Answers have similar NDCG curves, (iv) PageRank and HITS have similar results because both of these two algorithms consider the transitive property as we discussed in section 2, (v) ER (Sim) receives the lowest accuracy compared with the other three approaches, namely ER (VS), ER (RA) and ER (Hyb). The reason can be explained as similarity score only identifies the most related answer for a question



(a) ER (RA) in SO-C (b) ER (RA) in SO-O (c) ER (RA) in TT

(d) ER (Hyb) in SO-C (e) ER (Hyb) in SO-O (f) ER (Hyb) in TT

Fig. 3. NDCG score for ER (RA) and ER (Hyb)



(a) SO-C

(b) SO-O

(c) TT

Fig. 4. NDCG score for SO-C, SO-O and TT data set

but does not discriminate quality, (vi) in SO-C data set ER (VS) is better than ER (RA); in SO-O and TT data set ER (RA) is better than ER(VS). The reason is that in SO-C data set each answer receives enough votes and hence ER (VS) is much more effective in identifying the users' expertise, (vii) in SO-O data set, only ER-based approaches show a NDCG score of 1, and finally (viii) as expected, ER (Hyb) reaches the highest accuracy in *all* three data sets because this method combines both ER (VS) and ER (RA) together with the right value for γ ($\gamma = 0.75$ for SO-C and $\gamma = 0.5$ for SO-O and TT).

In summary, out initial hypothesis that both structure and answer quality are needed for these applications is borne out by the experimental results. Further, we have shown how to use alternative and available domain information beneficially. We have tied the values of weights to data set characteristics so that they can be determined easily.

6 Conclusions

We have analyzed the ask-answer graphs generated from a CQAs and highlighted subtle, but important differences with the conventional web graphs. Based on these differences, we have proposed the use of domain (or service-specific) information for mining expertise rank from CQA data sets. We have proposed the ExpertRank framework and several approaches to predict the expertise level of users with good accuracy by considering both answer quality and graph structure. We have demonstrated the effectiveness of our approach by comparing them with traditional link-based approaches.

References

1. Adamic, L.A., Zhang, J., Bakshy, E., Ackerman, M.S.: Knowledge Sharing and Yahoo Answers: Everyone Knows Something. In: Proceeding of the 17th International Conference on World Wide Web, pp. 665–674. ACM (2008)
2. Balog, K., Azzopardi, L., de Rijke, M.: Formal Models for Expert Finding in Enterprise Corpora. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 43–50. ACM (2006)
3. Cai, Y., Chakravarthy, S.: Predicting Answer Quality in Q/A Social Networks: Using Temporal Features. Technical report, University of Texas at Arlington, The Texas (2010), `http://itlab.uta.edu/ITLABWEB/Students/yuanzhe/TR/CSE-2012-8.pdf`
4. Campbell, C.S., Maglio, P.P., Cozzi, A., Dom, B.: Expertise Identification Using Email Communications. In: Proceedings of the Twelfth International Conference on Information and Knowledge Management, pp. 528–531. ACM (2003)
5. Dom, B., Eiron, I., Cozzi, A., Zhang, Y.: Graph-based Ranking Algorithms for E-mail Expertise Analysis. In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 42–48. ACM (2003)
6. Herbrich, R., Minka, T., Graepel, T.: Trueskill^{TM}: A Bayesian Skill Rating System. Advances in Neural Information Processing Systems 19, 569 (2007)
7. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems 22(1), 5–53 (2004)
8. Joachims, T.: Optimizing Search Engines Using Clickthrough Data. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 133–142. ACM (2002)
9. Jurczyk, P., Agichtein, E.: Discovering Authorities in Question Answer Communities by Using Link Analysis. In: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, pp. 919–922. ACM (2007)

10. Katz, L.: A New Status Index Derived from Sociometric Analysis. Psychometrika 18(1), 39–43 (1953)
11. Kleinberg, J.M.: Authoritative Sources in a Hyperlinked Environment. Journal of the ACM 46(5), 604–632 (1999)
12. Littlepage, G.E., Mueller, A.L.: Recognition and Utilization of Expertise in Problem-solving Groups: Expert Characteristics and Behavior. Group Dynamics: Theory, Research, and Practice 1(4), 324–328 (1997)
13. Liu, J., Song, Y.-I., Lin, C.-Y.: Competition-based User Expertise Score Estimation. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information, pp. 425–434. ACM (2011)
14. Mease, D.: A Penalized Maximum Likelihood Approach for the Ranking of College Football Teams Independent of Victory Margins. The American Statistician 57(4), 241–248 (2003)
15. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Technical report, University of Stanford, California (1999), http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf
16. Pal, A., Harper, F.M., Konstan, J.A.: Exploring Question Selection Bias to Identify Experts and Potential Experts in Community Question Answering. ACM Transaction Information System 30(2), 10 (2012)
17. Rode, H., Serdyukov, P., Hiemstra, D., Zaragoza, H.: Entity Ranking on Graphs: Studies on Expert Finding. Technical report, University of Twente, The Netherlands (2007), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.2300&rep=rep1&type=pdf
18. Salton, G.M., Wong, A., Yang, C.: A Vector Space Model for Automatic Indexing. Communications of the ACM 18(11), 613–620 (1975)
19. Shah, C., Pomerantz, J.: Evaluating and Predicting Answer Quality in Community QA. In: Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 411–418. ACM (2010)
20. Zhang, J., Ackerman, M.S., Adamic, L.: Expertise Networks in Online Communities: Structure and Algorithms. In: Proceedings of the 16th International Conference on World Wide Web, pp. 221–230. ACM (2007)
21. Zhang, J., Tang, J., Li, J.: Expert finding in a social network. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 1066–1069. Springer, Heidelberg (2007)

Community Expansion in Social Network

Yuanjun Bi[1], Weili Wu[1,2], and Li Wang[2]

[1] Department of Computer Science, University of Texas at Dallas
Richardson, TX 75080, USA
[2] College of Computer Science and Technology, TaiYuan University of Technology
Taiyuan, Shanxi 030024, China
{yuanjun.bi,weiliwu}@utdallas.edu, wangli@tyut.edu.cn

Abstract. While most existing work about community focus on the community structure and the tendency of one individual joining a community; equally important is to understand social influence from community and to find strategies of attracting new members to join the community, which may benefit a range of applications. In this paper, we formally define the problem of community expansion in social network, which is under the marketing promotional activities scenario. We propose three models, Adopter Model, Benefit Model and Combine Model, to present different promotion strategies over time, taking into consideration the community structure characters. Specifically, Adopter Model is based on the factors that can make an individual come into a community. Benefit Model considers the factors that attract more new members. Combine Model aims to find a balance between Adopter Model and Benefit Model. Then a greedy algorithm *ETC* is developed for expanding a community over time. Our results from extensive simulation on several real-world networks demonstrate that our Combine Model performs effectively and outperforms other algorithms.

Keywords: community expansion, community, strategy, social network.

1 Introduction

More and more people use online social sites, such as Facebook, Twitter and LinkIn to share interests and contact with each other. Its pretty impressive to see that by 2012, Facebook has more than 800 million active users, with Twitter 100 million and LinkedIn over 64 million in North America alone[1]. Due to their great social influence, some researchers study the structure of social networking, e.g.community detection, to simplify the representation. Some research work focus on the viral marketing analysis based on social medias. However, social medias or social communities need to develop themselves such that they can obtain greater influence and provide better service. In this paper, we argue that it is equally important to study the strategy for community expansion, which may carry significant benefits for a range of applications, such as

* This work was supported in part by the U.S. National Science Foundation under Grant CNS-0831579, CNS-1016320, and CCF-0829993.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 41–55, 2013.

i)*Broaden Sales Horizons*. Every company is looking for more customers to increase new sales. How to find potential customers is important especially when the marketing budget is limited. Community expansion problem analyzes how existing customers organized. And who should be potential new customers. It can provide strategy to increase customer community.

ii)*Political Campaign*. With the advent of Internet technology, the concept of community has less geological constraint. In some cases, it refers to a group people who have common will. Enlarging alliance is a good approach for political candidates to spread their influence towards decision making process. Modeling such influence based on community structure offers valuable insight in choices during campaign activities.

iii)*Boost Exhibition Participation*. Trade shows and exhibitions support a significant opportunity to enhance brand and product visibility. As the organizer of exhibitions, lager size and higher level participators can improve the fame of exhibition. How to choose excellent exhibit display participators that meet marketing needs and budgetary requirements can be found in our problem.

To illustrate our problem clearly, consider the following example. Suppose a company employs some salesmen to do promotional activities towards new customers. All the salesmen know the structure of the whole social network, that is who has relationship with whom. Since the cost of the promotional activities is limited, each salesman can only do the promotion to one person during a certain period. Our goal is to find a strategy for each salesman so that after several times there are new members as much as possible.

The challenged part is how to select the next potential customers to do the promotion. If the total number of new customers is our objective, we should not only consider those who are easily persuaded to come into the given community. Notice that each time when new customers come into the community, these new customers have influence on the structure of the community. In common sense, popular people may bring more customers but it is more difficult to persuade these people join the community. However, if more people came into the community, it will have more probability to attract those popular people.

Our paper engage in solving this problem. The main contributions of this paper can be listed as follows:

i)Formulate the problem of *Community Expansion* in social networks.

ii)Build three models for expanding the community, Adopter Model, Benefit Model and Combine Model. The first model considers factors that make an individual come into a community. The second model considers factors that make an individual attract more new members. The third model aims to find balance between Adopter Model and Benefit Model.

iii)Propose a greedy algorithm based on the above three models to present the community expansion progress. Based on the experiment results, an analysis is given to show which model is proper for which community structure.

The rest of this paper is organized as follows. In the next section, a brief overview of related work is introduced. Section 3 present the formal definition of our problem and several relative terms. We also compare our problem with other

similar problems in this section. Section 4 give three models for our problem. A greedy algorithm based on these models are proposed in section 5. The simulation results and conclusions are showed in section 6 and 7 respectively.

2 Related Work

There are a lot of work focus on social network structure and their diffusion. Newman, *et al*[2, 3] gave the definition of community that the nodes inside the community have tighter connections than nodes outside the community. Newman's Q value is considered as an important measurement for detecting community. Nguyen, *et al*[4] analysis four basic events occurring in dynamic network and propose adaptive algorithms separately to update the network community structure. Lars, *et al*[5]use decision tree to study the factors which influence an individual to join communities. They also study which community has more propensity to grow and how to measure the movement of individuals between communities. Kumar, *et al*[6] partition the nodes in network into three segments: singletons, middle region and giant component. Instead of using snapshot of network, Kumar put their experiments on entire lifetime of two large social network Flickr and Yahoo! 360 to study the overall properties of network and how these communities grow and merge. Their work are both based on the self development of a community which are different from the strategy that aims to enlarge a community as we study here. Other researchers [7–9] focus on marketing on social network. Domingos, *et al*[10] employ Markov random field to model the marketing value for each individual from collaborative filtering databases. The model use the influence between customers to increase the benefit. In [11] the authors extend their previous work by considering each customer's fund and reducing the computational cost. They apply the idea on knowledge-share sites. Generally, their work focus on the benefit which one single individual bring to a network but ignore the global group profit.

Information propagation problem is to find a set of initial set of users in a social network such that from this set the spread of influence in the network can be maximized[12]. Linear Threshold (LT) Model and Independent Cascade(IC) Model are two main approaches to formalize the influence maximization problem. Chen, *et al*[13] propose a MIA model and its heuristic algorithm to address the scalability and efficiency issue in large scale networks. Shaojie, *et al*[14] consider the links relationship impact on the information propagation. Saito, *et al*[15] predict the final influence over the whole network from a given initial set without modeling the diffusion process. They apply the expectation maximization(EM) algorithm to learn the influence probabilities. Goyal, *et al*[16] use action log to learn influence probabilities on each user. Their method can predict whether a user will take an action and tell when the action will be performed. Our work is different from all of the above. The comparison will be given in section 3.4.

3 Problem Formulation

3.1 Preliminaries

We start with introducing a set of fundamental concepts used throughout the paper. We denote the social network as a graph $G = (V, E, W)$, where V is a set of vertices, E is the set of edges and W is the weight matrix for edges. In a social network a vertex v corresponds to a person. An edge $e = (v, u)$ represents a connection between vertices v and u. w_{vu} represents the connection weight between v and u.

Definition 1. ***Target Community(TC)****: TC is a subgraph of G whose size we aim to enlarge. TC satisfies the definition of community that nodes inside the community are more densely connected internally than with the rest of the network. We consider the nodes inside TC as original customers(OC) while the nodes outside TC as potential customers(PC). Let $TCs = \{TC_1, TC_2, ..., TC_T\}$ be a series of target community, where TC_k is a snapshot of a target community TC at time t_k,($k \in [1, ..., T]$).*

Definition 2. ***Sales List(SL)****: SL is a subset of nodes which are outside the target community (i.e PC). Suppose there are M sales lists such that $PC = SL(1) \cup SL(2) \cup ... \cup SL(M)$. Note that we allow that different SL can have different number of nodes and one SL can have different versions over time, denoting the version at time t_k as $SL(m)_k$,($k \in [1, ..., T]$,$m \in [1, ..., M]$).*

Definition 3. ***New Customers(NC)****: Some customers will decide to join in TC after promotion. Among these new customers someone join because of promotional activity from salesmen. We define them as Mark Customers **MC**. While others receive no promotion but are influenced by their friends. We define them as Automatical Customers **AC**. The number of NC changes with each promotion time t_k and $NC = MC \cup AC$.*

3.2 Problem Definition

The progress of community expansion can be considered as the result of community attraction to individuals outside the community. The attraction can be departed to two parts.

1. Due to promotional activities from the target community, some potential customers are attracted. We denote the direct influence from target community to an individual i as $f_{TC \to i}$.

2. Through "word of mouth", some customers should be influenced by their neighbors. We denote the influence from one individual i to another individual j as $f_{i \to j}$.

The final influence from target community to one individual i can be described as

$$F_{TC \to i} = f_{TC \to i} + \sum_{k=1}^{d(i)} w_{ij_k} f_{i \to j_k} \tag{1}$$

where $d(i)$ denotes the number of neighbors of i. w_{ij_k} denotes the influence weight between i and j.

Suppose at time slot t, each salesman chooses one customer from their sales list and form the promotion target customers set $L_t = \{i_1^t, i_2^t, ..., i_m^t\}$, recall that m is the number of sales lists. Then our problem can be defined as

Community Expansion Problem. Given social network $G(V, E, W)$, sales list SL, target community TC and time slot $t \in \{1, ..., T\}$, find nodes sets L_t, such that the influence from TC to L_t, $\sum_{t=1}^{T} F_{TC \to L_t}$ can be maximized.

$$\sum_{t=1}^{T} F_{TC \to L_t} = \sum_{t=1}^{T} (\sum_{a=1}^{m} f_{TC \to i_a^t} + \sum_{a=1}^{m} \sum_{b=1}^{d(i_a^t)} w_{i_a^t j_b} f_{i_a^t \to j_b}) \tag{2}$$

In our paper, the influence f refers to attracting new customers. Specifically, the result of $f_{TC \to i}$ can be seen as customer i who accepts promotion and chooses to be a new member of TC(i.e Mark Customer(MC)). The value of $f_{i \to j}$ can be considered as customer j who did not receive promotion but was influenced by friend i, decides to join TC as well(i.e Automatical Customer(AC)). To extend our problem, influence function f can be described in other ways, such as generating new connections or strengthening existing connections with TC.

3.3 Problem Assumptions

Our problem of expanding Target Community is based on the following assumptions:

i)**Specific Potential Client Each Time.** That means in each time slot t_k each salesman can do the promotional activity to one and only one potential customer in their corresponding Sales List. Once one potential customer was chosen to be promoted, he or she should be removed from the Sales List.

ii)**Closed Customers Information Open Community Information.** We assume that Salesmen don't share their own potential customers information with each other. That means any two Sales Lists are not overlapping. However, each person in the network G can get the latest information about the Target Community structure. After each time slot, each person will obtain the TC information and check whether the changes will affect its action.

iii)**No New Connection Details.** After each promotional activity, there might be some New Customers(NC) come into TC. However, in our paper we assume that the connections among the nodes in NC won't change in T time slots. That will be true since in real world, T time slots might be very small compared to the time which the Target Community uses to build up. The network may be very large and complex so that even there are some connections changed in T time slots, these changes won't affect the G's structure too much. On the other hand, the community organizers only care about enlarging the size of their community. They don't care about the new connections after new customers coming in TC.

3.4 Comparison with Influence Maximization Problem

Compare to Influence Maximization problem, our problem is different in that:

(1) In our problem, we focus on the influence from a community which has a specific structure, rather than the initial seed set in which nodes distributed randomly. The community structure constraints provide some factors which should be considered when building the models. While in Influence Maximization problem, the influence source has no constraints.

(2) In Influence Maximization problem there is only one interference in the diffusion progress which is choosing the initial seed set occurring at the beginning of diffusion. After choosing the seed set all the diffusion progress proceed automatically. While in our problem human interference occur during the whole progress. The salesmen do the promotion several times until the result is satisfied. Each time they will adjust the candidates list according to the current social graph structure.

(3) Our goal is to maximize the size of community not to spread the influence from the seed set as described in Influence Maximization problem.

4 Community Expansion Models

4.1 Intuitions

Before formally introducing the model, we first explain several key observations:

Observation 1. *In [5], the study shows that the tendency of an individual to join a community depends on the underlying network structure. The probability p of joining a community depends on the number of friends k who are already in the community. The relation between p and k is under the "law of diminishing returns". Besides k, how these friends connected in the community also affect an individual's decision. If an individual has no friends in the community, then "how far" from the people in the community will decide "how much" the community impacts on the person. [17] infers that everyone is approximately six steps away from others.*

Observation 2. *Approximately 25% of US advertisements employ celebrities in their media[18]. We cannot ignore that celebrities have positive impact on consumer attitudes towards the purchase intention. Considering of that, a company should consider the financial returns from celebrities. In the real world the celebrities are more likely be known by others. In the network graph, we can consider the nodes which have more connections with other nodes as the celebrities.*

Based on the intuitions and observations, we know that both the probability of an individual coming into a community and the benefit of an individual to a community depends on the current network graph structure. Now, we want to build the Adopter Model to present how easy a potential customer join TC and build the Benefit Model to denote how much benefit the customer can bring into TC. Then Combine Model considers these two factors both.

4.2 Adopter Model

In Adopter Model, an individual who wants to join in TC is affected by how many friends in TC and how close these friends are. We define the meaning of friends is the same as neighbors in the following context, who have direct connection with this individual. Let η denote the value of how easy an individual adopts the promotional activity from TC. We give the formulation of η by the value of k friends in TC

$$\eta = (a_1 \log k) + (a_2 * \frac{k}{n}) + (a_3 * d_{in})(k > 0) \tag{3}$$

$$\eta = \frac{a_4}{dis}(k = 0) \tag{4}$$

where, k is the number of friends in TC. n is the number of neighbors of the individual. d_{in} denotes the density of k friends connected in TC. dis is the distance between the individual and the first node which the individual meet in TC. a_1, a_2, a_3, a_4 are adjusted parameters to make the model to fit data set. Function 3 denotes the customers who already have friends in TC. Intuitionally, people are more likely join TC if they have more friends in TC. However, if there are enough people to affect the individual to decide to join TC, additional friends will have small effect. Such that η is not linearly changed with k. On the other hand, even if two persons have the same number of friends in TC, that does not mean they will both choose to accept TC. The high ratio of friends in TC will obtain greater influence from TC. So we consider $\frac{k}{n}$ here. The connection density of friends in TC is also important. The more mutual friends they have, the stronger power they have to affect another individual. Here, we compute d_{in} by

$$d_{in} = \frac{2 * \rho}{k * (k-1)} \tag{5}$$

where, ρ denotes the number of connections between k friends in TC. $\frac{k*(k-1)}{2}$ stands for the the number of connections if any two friends in TC have a relation. As showed in $Fig.1$, node V_1 has four friends V_2, V_3, V_4 and V_5 in TC. In $Fig.1(a)$, the connections between these four friends is 6 and their $d_{in} = 1$, while in $Fig.1(b)$, the four friends has no connection with each other so their $d_{in} = 0$.

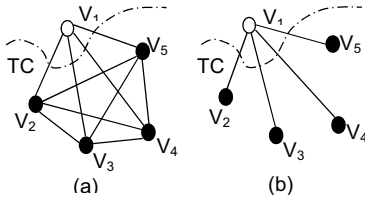


Fig. 1. Friends Connections in TC

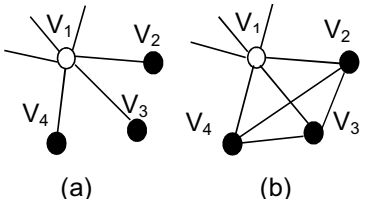


Fig. 2. Friends Connections not in TC

Function 4 give the η value if the individual has no friends in TC. The distance from TC determines how easy the individual to join in TC.

4.3 Benefit Model

Now we build Benefit Model to present the benefit that an individual can bring into TC. This model is important when the marketing strategy considers the cost constraints of promotional activities. Here to simplify the problem we assume that the promotional cost to each potential customer is the same. We will put the cost problem in the future extended work. Since our goal is to enlarge the size of Target Community as much as possible, the individual who knows more people will have more opportunities to introduce the Target Community to others, and attract more people coming into TC. On the other hand, if a popular node already has a lot of friends in TC, its benefit will be less than a node who have many friends that still are potential customers. Another factor that affects an individual's benefit is the structure of friends' connections. If the connections among friends is very density, it is difficult to persuade them to join TC since each individual is deeply influenced by the power of groups, not by a single person. As showed in $Fig.2$, node V_1 has three friends V_2, V_3 and V_4 who are potential customers. In $Fig.2(a)$, the connections between these three friends is 0 while in $Fig.2(b)$, these three friends have connections with each other so they are not that easily persuaded by V_1.

Based on the above analysis, we build the Benefit Model θ:

$$\theta = (b_1 * n) + (b_2 * \frac{n-k}{n}) - (b_3 * d_{out}) \tag{6}$$

where, n stands for the number of neighbors. $\frac{n-k}{n}$ presents the tendency of how many potential customers that the individual can attract. b_1, b_2, b_3 are adjusted parameters. d_{out} denotes the connections density among friends who are not in TC. It is computed by

$$d_{out} = \frac{2 * \sigma}{(n-k) * (n-k-1)} \tag{7}$$

where, σ denotes the number of connections among neighbors who are not in TC. $\frac{(n-k)*(n-k-1)}{2}$ stands for the the number of connections if any two neighbors who are not in TC have a relation with each other.

4.4 Combine Model

Combine Model wants to find a balance between Adopter Model and Benefit Model. Choosing customers who are not too easily joining TC but still have some benefit that will attract new customers in long term. We take some factors from Adopter Model and Benefit Model respectively to define Combine Model.

$$\gamma = c_1 \log k + c_2 * (n-k) + c_3 * d_{in} - c_4 * d_{out} \tag{8}$$

where c_1, c_2, c_3, c_4 are adjusted parameters. The definitions of k, n, d_{in} and d_{out} can be found in Adopter Model and Benefit Model.

5 The Algorithm

In this section, we propose an algorithm which includes three stages for Expanding Target Community (ETC). The first step is to find *Mark Customers(MC)* set after one promotional activity, in which we compute score for each potential customer and choose the one with highest score from each sales list. Then we compute its probability to see whether it will join TC or not. The second step is to update the graph information. The final step is to check whether there are *Automatical Customers(AC)* after graph was updated. After T times promotional activities we will get the total value MC and AC.

Algorithm 1: ETC Algorithm

Input: $G = (V, E)$,TC, m Sales Lists sl_1,$\cdots$,sl_m, T;
Output: NC,MC,AC;
$t = 0, MC = \emptyset, AC = \emptyset, NC = \emptyset$;
while $t < T$ **do**
  $t \leftarrow t + 1$;
  for *each* $sl_i, i < m$ **do**
    compute each n's score $S(n)$,$(n \in Sl_i)$;
    select the node v with the highest score in sl_i;
    compute v's joining probability $p(v)$;
    if $p(v) > \lambda$ **then**
      $MC \leftarrow MC \cup \{v\}$;
    end
  end
  for *each* $v \in MC$ **do**
    for *each v's neighbor* w **do**
      $w.k = w.k + p(v)$;
    end
  end
  for *each* $v \in MC$ **do**
    for *each v's neighbor* w **do**
      if $\frac{w.k}{w.n} > \lambda$ **then**
        $AC \leftarrow AC \cup \{w\}$;
      end
    end
  end
  $NC = MC \cup AC$;
end

5.1 Customer's Score and Joining Probability

To find Mark Customer set, we need to obtain the most reasonable potential customer in each Sale List. Based on three different models we discussed above *Adopter Model*, *Benefit Model* and *Combine Model*, we compute η, θ or γ for each node as its score separately. Sort each Sale List by the score's value in

descending order. The node with the highest score in each list will be severed the promotion. It comes into TC with some probability which has been studied by Lars, *et al*[5]. They find that the tendency of an individual to join a community is influenced by the number of friends within the community and by how those friends are connected to each other. In our algorithm, the probability of an customer v join in TC, $p(v)$, can be computed as Equation 9 for appropriate a, b, c.

$$p(v) = a \log k + b * d_{in} + c(k > 0) \tag{9}$$

For those customers who don't have friends in the community, we consider they still have probability to join in TC, with

$$p(v) = \frac{d}{dis}(k = 0) \tag{10}$$

The definitions of k, d_{in}, dis are the same as the description in Adopter Model. d is the parameter. Here we use algebraic function $f(x)$ to make sure the probability value is between 0 and 1.

$$f(x) = \frac{x}{\sqrt{1 + x^2}}$$

In our algorithm, whether a customer will join in TC is decided by threshold λ, $0 < \lambda < 1$. It is a factor reflects how easy an individual can join in a community. We will see how λ affects results in the experiments.

5.2 Graph Information Update and Automatical Customers

The coming of new Mark Customers will change their neighbor's information about friends number in TC k and connection density of friends. These updates will make some neighbors join TC automatically. We define that if there exists more than λ ratio of friends in TC, the customer will become *Automatic Customer*(AC). The nodes from AC will affect their neighbors as well, so the process of finding AC will not cease until no nodes from AC make their neighbors join TC. Figure 3 illustrates for V_1 how its k value is updated. After on promotional activity, node V_3 and V_4 join TC with the probability $p(3) = 0.65, p(4) = 0.6$. Node V_2 is an original customer in TC. Now node V_1 has $1+0.65+0.6 = 2.25$ friends in TC. Since $\frac{k}{n} = \frac{2.25}{4} > 0.5(\lambda = 0.5)$, node V_1 will join TC automatically.

6 Experiment

We conduct experiments on ETC algorithm as well as other two algorithms on four real-world networks. Our experiments aim at illustrating the performance of ETC algorithm from the following aspects: (1) Its capacity of attracting new members comparing to other algorithms; (2) Its efficiency of attracting new members comparing to other algorithms; (3) The tuning of its control parameter λ.

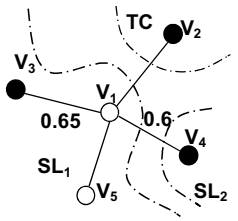


Fig. 3. Effect on neighbors

6.1 Experiment Setup

Datasets. We use three realistic data sets: American College Football, Arenas Email, NetHEPT and Facebook.

AmericanCollegeFootball(*ACF*) The network is a representation of the schedule of Division I games for the 2000 season, in which vertices represent teams (identified by their college names) and edges are regular-season games between the two teams they connect.

Arenas/email It comes from email interchange network, Univ. of Rovira i Virgili, Tarragona. The nodes are the members in this university and the edges represent email interchanges between members.

NetHEPT This data set is an academic collaboration network taken from the "High Energy Physics (Theory)"section (from 1991 to 2003) of arXiv. The nodes in NetHEPT denote the authors and the edges represent the co-authorship.

Facebook The nodes in Facebook denote the facebook users and the edges represent the friendship. We choose these networks since it covers a variety of networks with size ranging from $1K$ edges to $1M$ edges. Some statistics about these networks' properties are given in Table 1. Close customer refers to individual who has friends in TC.

Table 1. Data Sets Properties

DataSets	NetHEPT/Author	Arenas/Email	ACF/Team	Facebook
Number of Nodes	15233	1133	115	63732
Number of Edges	62774	10903	1226	1634180
Number of Sale List	1819	70	12	210
Average Sale List Size	8.4	15.9	8.9	227.5
Target Community(TC) Size	1251	179	12	15963
Ratio of close customer	0.057	0.36	0.26	0.01
Average Friends of close customer	2.06	2.26	1.26	1.84
Average Degree	3.76	9.17	10.67	0.33

Generating Target Community
To find TC and Sale Lists, we first partition the social network graph into several communities. We select the community with the maximum size as the Target Community TC. The rest communities are considered as Sale Lists.The partition

of NetHEPT and Facebook employ the methods in [19], and partition of Arenas and ACF use the methods in [20].

Algorithms
We use our ETC algorithm based on the three models discussed above. Compare the three models with a baseline model and another algorithm which solves the similar problem. The following is a list of algorithms we evaluate in our experiments.

(1) ETC: Our algorithm is a greedy algorithm. Base on our Adopter Model, Benefit Model and Combine Model, we have methods $ETCA, ETCB, ETCC$ respectively.
(2) Random: As a baseline comparison, simply select node from each Sales List each time.
(3) TABI: TABI is a heuristic algorithm proposed by Tao,*et al*[21] to solve the participation maximization problem. This algorithm calculates participants' influence and allocate thread according to influence ranking, which is similar with our ETC algorithm. However, TABI only considers people who have participated in the forum, which means the algorithm only chooses candidates from people who have connections with the community. It computes every participant v's influence by

$$(1 - \prod_{u \in v.K} (1 - w_{u,v}))(1 + \sum_{x \in (v.N - v.K)} w_{v,x} \prod_{y \in x.K} (1 - w_{y,x}))$$

Here, $v.K$ refers to v's friends set in the community. $v.N$ refers to v's neighbor set. Since our data sets are unweighed social networking, each edges has the same weight.

To obtain each algorithm's capacity of attracting new members. We run the simulation 1000 times and take the average of results, which matches the accuracy of the greedy algorithm.

6.2 Experimental Results

Capacity of Attracting New Customers. We measure the capacity of attracting new customers by two measurements, Automatical Customer size and New Customer size. The promotion time T ranges from 1 to 10. The first measurement is for evaluating the performance of attracting people who can bring more benefit to the community. The second measurement is for evaluating the performance of attracting more new customers totally.

For the moderate sized graph Arenas Email, as showed in Figure 4 and Figure 5, our ETCC performs best on both two measurements. When $T = 10$, for the New Customers measurement, ETCC is $4.1\%, 51\%, 81.8\%$ better, while for the Automatical Customers measurement, ETTC is $0.1\%, 32.8\%, 51.7\%$ better, comparing to ETCA, RANDOM and TABI respectively. ETTC performs even much better than ETCA, RANDOM and TABI when $T = 5$. The results of ETCB are very close to ETCC. TABI attracts more customers than RANDOM before

$T < 8$. After that, RANDOM obtains more customers. That phenomena proves that TABI is an expanding algorithm only considering people who have connection in the target community. TABI has weak capacity of attracting valuable customers who can bring automatical customers.

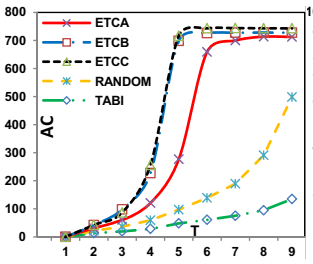


Fig. 4. Arenas_AC

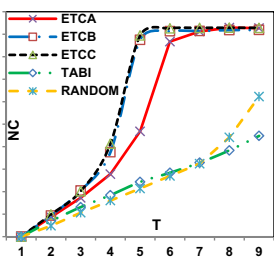


Fig. 5. Arenas_NC

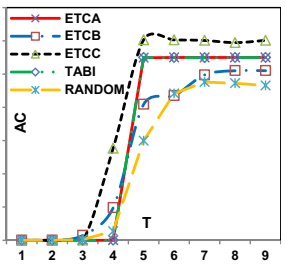


Fig. 6. ACF_AC

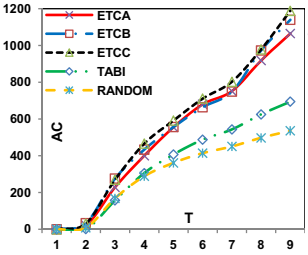


Fig. 7. NetHEPT_AC

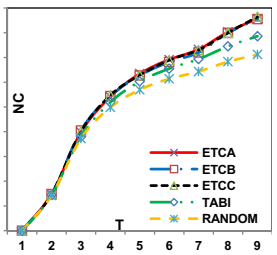


Fig. 8. NetHEPT_NC

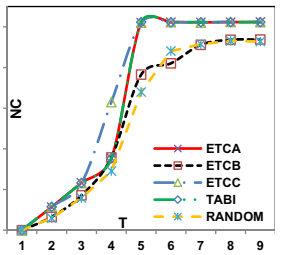


Fig. 9. ACF_NC

Figure 6 and Figure 9 show the results on ACF/Team dataset. ETCC still works the best on two measurements. For New Customers, ETCC is 8.4%, 9.3% better than ETCB, RANDOM respectively when $T = 10$. While for Automatical Customers, ETCC is 8.7%, 15.1%, 22.6% better than TABI, ETCB, RANDOM respectively. TABI which has the similar result as ETCA, performs better on this dataset. It is probably because ACF/Team is a small network with many people who has connection in TC(i.e a relatively large ratio of close friend). In this case, it seems choosing who can easily join TC is a better strategy for the community.

Next, for the 60 thousand edges NetHEPT dataset, Figure 7 and Figure 8 show ETCC performs slightly better than ETCA and ETCB, but consistently much better than TABI and RANDOM. For New Customers, ETCC is 9.1%, 17.6% better, while for Automatical Customers, ETCC is 41.6%, 54.9% better than TABI and RANDOM respectively when $T = 10$.

Finally, for the 1.6 million edge Facebook dataset, Figure 10 and Figure 11 show that this time ETCA performs much better than other algorithms. ETCB, ETCC and TABI have close results. For New Customers, ETCA is 55.4%, 75.4%

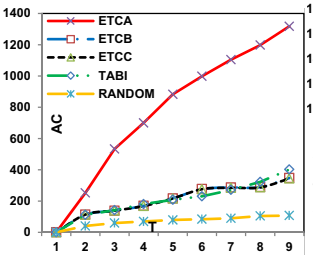


Fig. 10. Facebook_AC

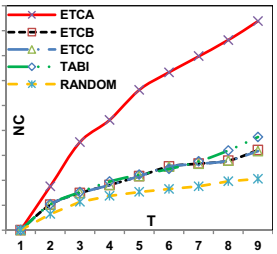


Fig. 11. Facebook_NC

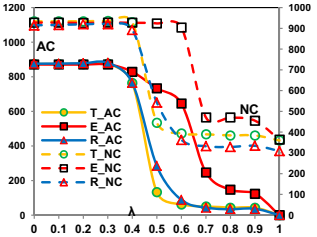


Fig. 12. Tuning of λ

better, while for Automatical Customers, ETCA is 69.5%, 91.9% better than TABI and RANDOM respectively when $T = 10$. This phenomena is quite different from phenomena of other datasets result as we have seen so far. Note that there are two unique features for this dataset: (a)the average degree of each node is small, which means the distribution of nodes in the network is scattered. As a result there are more nodes are easy to persuaded to join TC, which means ETCA is a good choice; (b)TABI seems to consider the nodes that can join in TC easily as well. However, TABI only considers nodes that have connections in TC while the ratio of close friend in this dataset is rather small. So TABI will ignore some nodes which in the view of ETCA is better choice.

Overall, we see that ETCC significantly outperforms the rest algorithms in most cases. ETCB and ETCA still have better results than TABI and RANDOM.

Efficiency of Attracting New Customers. The efficiency of attracting new customers is another important evaluation criterion, especially when the community considers the promotion time cost. In Figure 5 and Figure 9, we can see that when $T = 5$, ETCC curve has reach its peak value, which means it has attracted most new customers. While TABI and RANDOM need more time to reach their peak value.

Tuning of Parameter λ. We investigate the effect of the tuning parameter λ on the capacity of attracting new customers. λ ranges from 0 to 1. We compare the results of ETCC, TABI and RANDOM when $T = 10$. Since λ decides how easy an individual can join in TC, Figure 12 show that the capacity of attracting new customers increases when the λ value decreases, as expected. For both attracting NC and AC, ETCC(E_NC,E_AC) keep high value in lager range of λ than TABI(T_NC,T_AC) and RANDOM(R_NC,R_AC), indicating that ETCC performs more stable than TABI and RANDOM.

7 Conclusions

In this paper, we formally define the problem of expanding community. A greedy Expanding Target Community (ETC) algorithm is proposed, which employs three models Adopter Model, Benefit Model and Combine Model. These models

consider the factors that affect an individual to join community and the factors that attract new members. Experiment results based on four real world datasets shows that our models perform better than RANDOM and TABI algorithm.

References

1. Infographic: Social media statistics for 2012, http://www.digitalbuzzblog.com/social-media-statistics-stats-2012-infographic
2. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. In: PNAS (2002)
3. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Phys. Rev. (2004)
4. Nguyen, N.P., Dinh, T.N., Xuan, Y., Thai, M.T.: Adaptive algorithms for detecting community structure in dynamic social networks. In: INFOCOM (2011)
5. Backstrom, L., Huttenlocher, D., Kleinberg, J., Lan, X.: Group formation in large social networks: membership, growth, and evolution. In: KDD (2006)
6. Kumar, R., Novak, J., Tomkins, A.: Structure and evolution of online social networks. In: KDD (2006)
7. Burt, R.: Structural Holes: The Social Structure of Competition. Harvard University Press (1992)
8. McKenzie Mohr, D., Smith, W.: Fostering Sustainable Behavior: An Introduction to Community Based Social Marketing. New Society Publishers (1971)
9. Leskovec, J., Adamic, L., Huberman, B.: The dynamics of viral marketing. ACM Transactions on the Web (2007)
10. Domingos, P., Richardson, M.: Mining the network value of customers. In: KDD (2001)
11. Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: KDD (2002)
12. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: KDD (2003)
13. Wang, C., Chen, W., Wang, Y.: Scalable influence maximization for independent cascade model in large-scale soical networks. Data Mining and Knowledge Discovery (2012)
14. Tang, S., Yuan, J., Mao, X., Li, X., Chen, W., Dai, G.: Relationship classification in large scale online social networks and its impact on information propagation. In: INFOCOM (2011)
15. Saito, K., Nakano, R., Kimura, M.: Prediction of information diffusion probabilities for independent cascade model. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part III. LNCS (LNAI), vol. 5179, pp. 67–75. Springer, Heidelberg (2008)
16. Goyal, A., Bonchi, F., Lakshmanan, L.V.S.: Learning influence probabilities in social networks. In: WSDM (2010)
17. Milgram, S.: The small world problem. Psychology Today (1967)
18. Shimp, T.A.: Advertising promotion: Supplemental aspects of integrated marketing communications. South-Western College Pub. (2002)
19. Blondel, V.D., Guillaume, J., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. (2008)
20. Hu, Y., Chen, H., Zhang, P., Di, Z., Li, M., Fan, Y.: Comparative definition of community and corresponding identifying algorithm. Phys. Rev. (2008)
21. Sun, T., Chen, W., Liu, Z., Wang, Y., Sun, X., Zhang, M., Lin, C.: Participation maximization based on social influence in online discussion forums. In: ICWSM (2011)

Similarity Joins on Item Set Collections Using Zero-Suppressed Binary Decision Diagrams

Yasuyuki Shirai[1,3], Hiroyuki Takashima[1], Koji Tsuruma[2], and Satoshi Oyama[3]

[1] JST-ERATO MINATO Discrete Structure Manipulation System Project, Hokkaido University, Sapporo, Japan
{shirai,takashima}@erato.ist.hokudai.ac.jp
[2] NEC Corporation, Kanagawa, Japan
k-tsuruma@ak.jp.nec.com
[3] Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan
oyama@ist.hokudai.ac.jp

Abstract. Similarity joins between two collections of item sets have recently been investigated and have attracted significant attention, especially for linguistic applications such as those involving spelling error corrections and data cleaning. In this paper, we propose a new approach to similarity joins for general item set collections, such as purchase history data and research keyword data. The main objective of our research is to efficiently find similar records between two data collections under the constraints of the number of added and deleted items. Efficient matching algorithms are urgently needed in similarity joins because of the combinatorial explosion between two data collections. We developed a matching algorithm based on Zero-suppressed Binary Decision Diagrams (ZDDs) to overcome this difficulty and make matching process more efficient. ZDDs are special types of Binary Decision Diagrams (BDDs), and are suitable for implicitly handling large-scale combinatorial item set data. We present, in this paper, the algorithms for similarity joins between two data collections represented as ZDDs and pruning techniques. We also present the experimental results obtained by comparing their performance with other systems and the results obtained by using real huge data collections to demonstrate their efficiency in actual applications.

Keywords: similarity joins, error-tolerant matching, recommendation, zero-suppressed binary decision diagram.

1 Introduction

Similarity joins or error-tolerant matching algorithms between two collections of item sets have recently been investigated and have attracted significant attention [1, 2, 4, 6, 12–14, 17–19] for linguistic applications such as those involving spelling error corrections, data integration, data cleaning, and detection of similar words. The aims in these researches are to find similar records between two huge data sets based on similarity definitions such as cosine measure, Jaccard similarity, and edit distance.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 56–70, 2013.

Here, we propose a new efficient method of implementing similarity joins for general item set collections, such as those of purchase history and research keyword data. The main objective of our research is to efficiently find similar records between two data collections under the constraints of the number of added and deleted items. These constraints can be considered to be natural constraints for general unordered item set collections, and they have a broad range of possible applications. For example, consider the following item sets: $D_0 = \{a, b\}, D_1 = \{a, c\}, D_2 = \{a\}$ and $D_3 = \{a, b, c\}$, where all of the edit distances from D_0 to D_1, D_2 and D_3 are one. It is natural, however, to consider that the distance from D_0 to D_1 would be longer than that from D_0 to D_2 and D_0 to D_3 in real world data such as those in purchase histories rather than those in linguistic applications.

In this research, the similarity between records is not defined approximately but "exactly". Efficient matching algorithms are urgently needed in exact similarity joins because of the combinatorial explosion between two data collections. To overcome this difficulty and make matching process more efficient, we developed a matching algorithm based on Zero-suppressed Binary Decision Diagrams (ZDDs) [7, 9, 11]. ZDDs are special types of Binary Decision Diagrams (BDDs) [3] and suitable for implicitly handling large-scale combinatorial item set data. In our previous work [16], we developed an efficient method of set recommendation using ZDD structures. In our set recommendation, a set of items to be added or deleted is recommended to the initial item set so that the modified set is classified to the target class. In this paper, we extend this approach to establish set similarity joins between two collections of item sets.

The applications of our approach can cover a wide area of real applications where we need to find similar records between two huge collections of data sets such as those in :

- Patent searches
  To find similar previous patents by other research institutes, organizations, or countries.
- Data cleaning and duplicate entry detection
  To fix errors in databases compared with other databases, or detect duplicate entries between two databases.
- Similar research
  To find similar research papers or research activities in previous decades, those published by other organizations or countries, or those in different areas.
- Fraud detection
  To find fraudulent data in the database by comparing with previous fraud cases. In fraud detection, rule based method is one of the most realistic approaches. However, in fact, we can hardly define the rules explicitly to detect individual case of fraud. Our approach in this paper enables us to directly find similar cases to those in previous fraud data.

– Customer classification
  To find loyal or potentially-dependable customers, or to find unacceptable customers by comparing with those in previous customer databases.

In this paper, we present the experimental results obtained from comparing the performance of our approach with ordinary matching algorithms and other similar methods, and also the results from other experiments using real huge data collections such as those in DBLP research titles and NSF (National Science Foundation) research abstracts, to demonstrate its efficiency and availability in real applications.

The rest of the paper is organized as follows: Section 2 discusses some works related to our research, especially previous researches on similarity joins. Section 3 provides some definitions and describes the implementation of our framework using ZDD data structures. We present and discuss the results from evaluating the performance of our approach in Section 4, and applications using real data such as those from DBLP and NSF in Section 5. We conclude this paper in Section 6 with a brief summary and some additional comments, and mention future works.

2 Related Work

There have been many works about similarity joins [1, 2, 4, 6, 17–19] or error tolerant matching [12–14], which can be considered to be essential operations in many applications. The objectives in these researches have been to find similar records based on "exact" matching under various constraints such as cosine measure, Jaccard similarity and edit distance.

Chaudhuri [4] and Arasu [1] introduced a general operator called SSJoin, which could be extended to various measures such as edit distance, Jaccard similarity, Hamming distance. The algorithms PartEnum and WtEnum implemented with SSJoin are based on a filtering method, i.e., filtering effective similarity joins based on two ideas for signature generation called partitioning and enumeration.

Bayardo [2] proposed All-Pairs algorithms, on which irrelevant records could be filtered out using inverted lists that were dynamically created on the process according to constraints. They showed that All-Pairs algorithm was highly efficient and scalable to huge size of records.

Xiao [18] introduced a filtering approach focused on the number of "mismatching" and proposed Ed-Join algorithms. The mismatch-based filtering methods could reduce the numbers of candidates.

These approaches described above, were based on filtering-based methods to avoid redundant matching as much as possible. First, by generating the signatures for each data (e.g., string sequence), the candidates of similar pairs could be generated (filtering phase). Then, to find exactly similar pairs with defined similarities, these candidates were verified in a test phase. However, if the database consisted of sets of relatively short strings, signature based filtering methods could be an overhead since these approaches generally generated huge numbers of candidate pairs.

Feng [6] and Wang [17] proposed another approach rather than filtering methods, which was a trie based framework called Trie-Join, in which the data set could be represented as trie structures. Trie structures can share common prefixes for a set of strings. They proposed search procedures on trie structures and pruning methods for sub-structures of the trie.

In our work, we do not consider filtering methods, but focus on efficient representational structures and search algorithms to find similar records in two data collections. Although our approaches and motivation are close to those in Trie-Join [6, 17], Trie-Join assumed sequential patterns as input data and edit distance as constraints, because they considered language processing as a promising area of applications. However, our method takes into considerations general item sets (unordered sets) and the constraints are slightly different from the edit distance. Our system searches similar pairs under the constraints of the number of added and deleted items. We consider that our constraints on item addition and deletion are rather natural and generic for applications of general item set collections or various applications in bioinformatics [15].

In this research, we adopt Zero-suppressed Binary Decision Diagrams (ZDDs) [7, 9, 11] to implement the algorithms efficiently rather than trie structures. ZDDs are not tree structures but directed acyclic graph (DAG) structures that can share the same sub-structure.

3 Set Similarity Joins Using ZDDs

This section provides some definitions and examples of our set similarity joins based on the addition/deletion constraints. We then briefly introduce ZDDs to handle huge numbers of data sets, and present the algorithm for similarity joins on ZDD structures.

3.1 Preliminaries

We will provide various definitions and notations as follows.

Definition 1 (item). *An item is an atomic entity that represents a characteristic or feature and is denoted by a lower-case character* $(a, b, c, \ldots)$*. A set of all items to be considered is denoted by* Σ*.*

Definition 2 (item set and collection of item sets). *An item set is a set of items that represents the characteristics or features of an object (we use* D *to represent an item set). A collection of item sets is a set of item sets, and is represented by* $\mathcal{S}$ *or* $\mathcal{T}$*.*

We sometimes call an item set a "record", and an item set collection a "database" for simplicity.

Definition 3 (add/delete-constraints). *The upper bounds on the numbers of items that can be added or deleted for an item set are denoted by* N^+*,* N^-*, respectively.*

If we have item set $D_1 = \{a, b, c, d\}$ and $N^+ = 2$, $N^- = 2$, then item set $D_2 = \{a, e, f, d\}$ is a modification to D_1 under the constraints of N^+ and N^-.

Note that we do not consider a data set as an ordered sequence. Although the edit distance between D_1 and D_2 in the above example is two (if we consider the distance of replacement as one), the edit distance between D_1 and $D_3 = \{a, d, e, f\}$ is four, while D_1 and D_3 also satisfy the condition $N^+ = 2$ and $N^- = 2$ as well as D_1 and D_2.

As an example, suppose we have two collections of item sets, $\mathcal{S}$ and $\mathcal{T}$:

$$\mathcal{S} = \{\{a, b\}, \{a, c\}, \{c\}\} \tag{1}$$

$$\mathcal{T} = \{\{a, d\}, \{a, b, c\}, \{b\}\} \tag{2}$$

The similarity joins of $\mathcal{T}$ for $\mathcal{S}$ under the condition of $N^+ = N^- = 1$ would consist of the following six pairs : $\{\{a, b\}, \{a, d\}\}$, $\{\{a, c\}, \{a, d\}\}$, $\{\{a, b\}, \{a, b, c\}\}$, $\{\{a, c\}, \{a, b, c\}\}$, $\{\{a, b\}, \{b\}\}$, $\{\{c\}, \{b\}\}$.

If the conditions of addition and deletion are $N^+ = 1$ and $N^- = 0$, only the pair of $\{\{a, b\}, \{b\}\}$ satisfies the condition. If $N^+ = 0$ and $N^- = 1$, two pairs $\{\{a, b\}, \{a, b, c\}\}$ and $\{\{a, c\}, \{a, b, c\}\}$ satisfy the condition.

3.2 Zero-Suppressed Binary Decision Diagrams

This subsection reviews ZDDs, which are a variant of Binary Decision Diagrams [3, 7] (BDDs). BDDs are well-known and widely used for efficiently manipulating large-scale Boolean function data. A BDD is a directed graph representation of the Boolean function. The reduction rules in a BDD consist of a "node deletion rule" (delete all redundant nodes with two edges that point to the same node) and a "node sharing rule" (share all equivalent sub-graphs).

Zero-suppressed BDDs (ZDDs) [7, 9, 11] are special types of BDDs that are suitable for implicitly handling large-scale combinatorial item set data. The reduction rules for ZDDs are slightly different from those for BDDs and are outlined in Fig. 1.

- Share equivalent nodes as well as ordinary BDDs ((1) in Fig. 1).
- Delete all nodes whose 1-edge directly points to the 0-terminal node, and jump through to the 0-edge's destination ((2) in Fig. 1).

ZDDs are especially much more effective than BDDs for representing "sparse" combinations such as purchase history data. For instance, sets of combinations selecting 10 out of 1000 items can be represented by ZDDs up to 100 times more compactly than those by using ordinary BDDs.

Fig. 1 has an example of reduced ZDDs for $\mathcal{S}$ in 3.1. Note that the non-existence of a node on each path means that the item is negated in the ZDD representation. For example, $\{c\}$, which is a model of $\mathcal{S}$, can be represented by the heavy line path in ZDD representation. $\mathcal{S}$ can be rewritten in a sum of product representations as $ab + ac + c$ within the ZDD context.

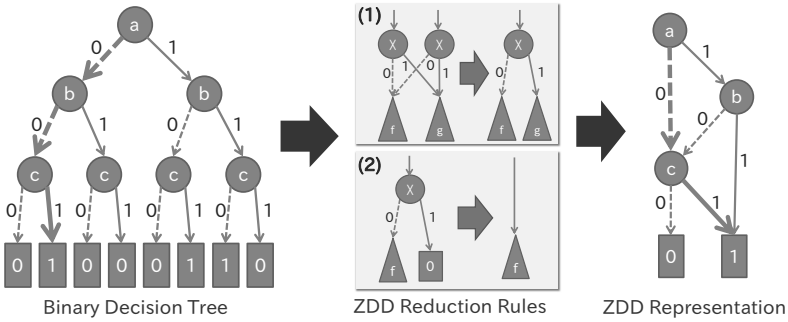


Fig. 1. Reduced Binary Decision Tree with ZDDs

3.3 ZDD Based Similarity Joins

In our approach, two collections of item sets in text format are transformed into ZDD structures via the ZDD package [10]. The input data of our system are two ZDD structures and the constraints description. This subsection presents the method for calculating similarity joins for two collections of item sets, i.e., for two ZDD structures.

Algorithm 1 outlines the search algorithm on ZDD structures. Fig. 2 has a simple example of set similarity joins using ZDD based on Algorithm 1, where the objective is to find similar item sets between $\mathcal{S}$ (diagram at left) and $\mathcal{T}$ (diagram at right) under the constraints of $N^+ = N^- = 1$. The search step begins to work from top node a on $\mathcal{S}$ ($search\_zdd(n)$ in the algorithm).

Each square box on the edge indicates the search results, which is a list of a pair: the current count of addition and deletion, and the corresponding path on $\mathcal{T}$. For example, box (1) on $\mathcal{S}$ is attached to the edge which indicates the negation of a. `+0-1:2` in box (1) indicates that the current count is "no additional item (`+0`) and that one item has been deleted (`-1`)" for edge 2 on $\mathcal{T}$. In the same manner, `+0-0:1` in box (1) indicates that the current count is "no additional or deleted items" for edge 1 on $\mathcal{T}$. The *update_candidate* in the algorithm adds new candidates to the current candidates, $n_1.cand$ and $n_0.cand$. The *reduce* function reduces the candidate set and checks the constraints.

Similarly, box (2) is created based on the results for box (1). We need the result for box (3) as well as the result for box (2) to create the box (4). The search process in our algorithm only proceeds if all parents of the node have already finished their processes. After box (3) is calculated, the calculation of box (4) starts based on the results for box (2) and (3).

For box (6) which means the addition of d in $\mathcal{S}$, all the counts in boxes (4) and (5) must add 1 to the addition of items, because no item sets in $\mathcal{T}$ have d as their element. As a result, none of the counts in boxes (4) and (5) satisfy constraints N^+. If no elements in the box satisfy the condition N^+ or N^-, searching along that path is terminated. Hence, box (6) becomes ϕ.

Algorithm 1. Search Algorithm on ZDD structures

```
n_0 is a top node of the ZDD S;
n_0.cand = {{+0 − 0 : 0}};
search_zdd(n_0);

function SEARCH_ZDD(n)
    if all of other ancestors of node n have not been processed then return
    end if
    if n is a terminal node then return cand; // output candidates
    else
        n_1 = n.edge_1.dest; // n_1 : destination of 1 edge of node n
        n_0 = n.edge_0.dest; // n_0 : destination of 0 edge of node n
        n_1.cand = update_candidate(n.edge_1, n.cand, n_1.cand);
        n_0.cand = update_candidate(n.edge_0, n.cand, n_0.cand);
        // update candidates for edge 1 and 0
        n_1.cand = reduce(n_1.cand);
        n_0.cand = reduce(n_0.cand);
        // reduction of the candidate set and check the constraints
        if n_1.cand is not NULL then return search_zdd(n_1);
        end if
        if n_0.cand is not NULL then return search_zdd(n_0);
        end if
    end if
end function
```

Although there are no solutions under the constraints for the example in Fig. 2, another example in Fig. 3 can obtain two sets of results, where the square box labeled `+0-1:2-4-5-6-7` indicates one item has been deleted from path `2-4-5-6-7` (i.e., $\{a, c, d\}$). We can hence, obtain the final result, $\{a, c, d\} - \{c\} = \{a, d\}$. We can similarly obtain $\{b, c, d, e\}$ (addition of e to $\{b, c, d\}$) from the label of `+1-0:1-3-5-6-7`.

4 Performance Evaluation

4.1 Comparison on Sorted Text Search

We first evaluate the efficiency of our approach based on ZDDs, using artificial data. The problem we provided to evaluate performance in this experiment consists of 170 items in total ($|\Sigma| = 170$), and each record randomly contains five items. We prepare two data sets as $\mathcal{S}$ which contain one million data and 10 million data respectively, and four data sets as $\mathcal{T}$ with sizes from $1000-1000000$. We compare three types of programs for set similarity joins for two given data sets, i.e., "(1) Text-Linear" : between two sorted text data sets, "(2) ZDD-Linear": between ZDD and a sorted text data set, "(3) ZDD-ZDD" : between two ZDDs presented in this paper.

All the systems were implemented in C++, and the experiments were run on a SUSE Linux Enterprise Server 11 with 32 Intel Xeon CPUs (2.66 GHz) and

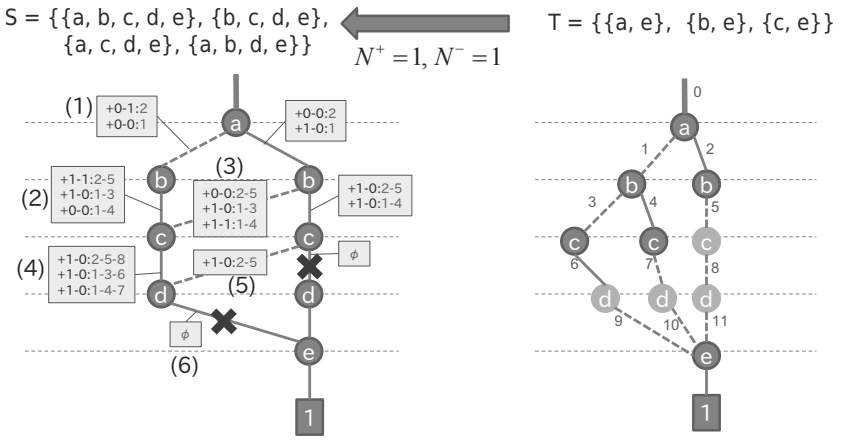


Fig. 2. Example of set-similarity joins using ZDD (1)

1.024 TB RAM. Table 4 compares the execution times. None of the execution times include the time for preparatory data processing, i.e., data sorting or ZDD construction.

As we can see from the table, the execution times for (1) Text-Linear and (2) ZDD-Linear increase linearly for the size of search data, while (3) ZDD-ZDD method can suppress the increase in execution time. We could conclude from these results that our algorithms based on two ZDD structures worked efficiently than linear searches (1) and (2).

4.2 Comparison with Trie-Join

As we previously described, Trie-Join [6, 17] is a similar approach to our system. It, however, only treats totally ordered sequences. Although it is difficult to make precise comparisons with our systems, the experimental results in this subsection provide some indications about their efficiency.

The data sets used in the experiments were as follows :

- Item sets are generated by the alphabet ("a" to "z" in Trie-Join, and "x01" to "x26" in the ZDD-based method) in alphabetical (or numerical) order.
- The length (number of items) of each record is 10. However, if duplicate items occur in the records, we suppress them since the ZDD approach does not distinguish the plural occurrence of items.

Table 1 lists the sample data we used in this experiment.

There are three variations of data sets, each of which consists of 100000, 500000, and 1000000 data records. The results from execution by Trie-Join[1] and

[1] We used the Trie-Join program on :
`http://dbgroup.cs.tsinghua.edu.cn/wangjn/codes/triejoin.tar.gz`

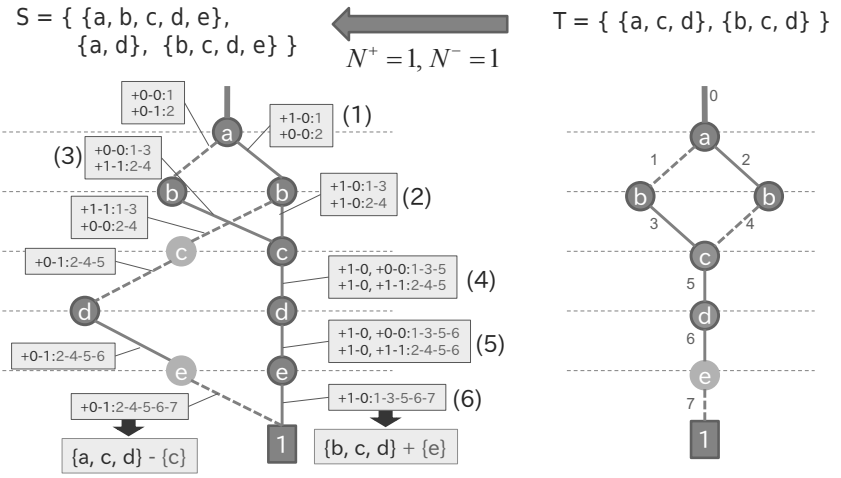


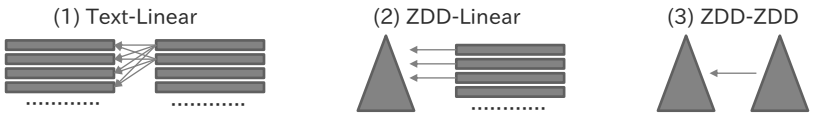
Fig. 3. Example of set-similarity joins using ZDD (2)

Table 1. Sample Data Used in Trie-Join and ZDD-based Method

Trie-Join input	ZDD-based method input
aegklorstw	x01 x05 x07 x11 x12 x15 x18 x19 x20 x23
bcegjmtvxy	x02 x03 x05 x07 x10 x13 x20 x22 x24 x25
filmrsux	x06 x09 x12 x13 x18 x19 x21 x24
dijkmqrt	x04 x09 x10 x11 x13 x17 x18 x20
aeinprst	x01 x05 x09 x14 x16 x18 x19 x20
kqrvwy	x11 x17 x18 x22 x23 x25
aefghlqvx	x01 x05 x06 x07 x08 x12 x17 x22 x24
acgknoruxy	x01 x03 x07 x11 x14 x15 x18 x21 x24 x25

our ZDD-based method are summarized in Table 2. Fig. 5 also compares performance of two systems for some selected results in Table 2 where the horizontal axis plots the number of results and the vertical axis plots the execution times (sec).

As can be seen in the table, we concluded that our system could achieve the same or better performance than Trie-Join for large scale problems. For example, in Table 2, the execution time with Trie-Join is 436.6 (sec) to generate approximately 149 million results for 1000000 records and edit distance = 2. On the other hand, the execution time with the ZDD-based method for 1,000,000 records and $N^+ = N^- = 2$ is 1039.0 (58.5 + 980.5) (sec) to generate approximately 809 million results which is over 5 times as much as the case of edit distance = 2 on Trie-Join (the set of 149 million results by Trie-Join is a proper subset of 809 million results by our ZDD-based method).



Data Set Size	Search Data Set Size	Condition		Execution Time (sec)			Num. of Visit Nodes	
				(1) Text-Linear	(2) ZDD-Linear	(3) ZDD-ZDD	ZDD-Linear	ZDD-ZDD
1,000,000	1000	add=1	delete=1	37.4	3.2	2.3	10,657,783	781,112
1,000,000	10,000	add=1	delete=1	406.6	30.8	6.5	106,268,058	1,363,943
1,000,000	100,000	add=1	delete=1	4055.9	310.2	20.1	1,055,439,664	1,987,588
1,000,000	1,000,000	add=1	delete=1	-	3124.0	98.9	10,537,412,252	2,616,617
10,000,000	1,000	add=1	delete=1	406.7	7.7	7.7	23,353,846	3,175,805
10,000,000	10,000	add=1	delete=1	4061.5	79.6	24.9	236,444,648	7,425,564
10,000,000	100,000	add=1	delete=1	39989.3	793.8	81.6	2,369,599,985	13,450,436
10,000,000	1,000,000	add=1	delete=1	-	7953.7	331.4	23,732,345,100	19,627,528

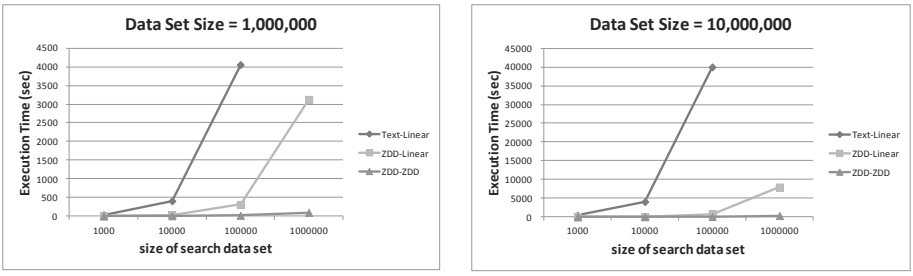


Fig. 4. Performance Evaluation

Since these two systems assumed different constraints, we could not conclude quantitative discussions preciously. Our system, however, could achieve at least potential ability comparable with Trie-Join for the problems of similarity joining with ordered sets.

5 Applications

This section presents some application results with real data sets that concern research topics. We used the data sets from DBLP and NSF data collections.

5.1 DBLP Research Titles

This experiment took into account the paper titles in a DBLP xml data set[2] whose tags include "article" and "inproceedings". The total size of records in this experiment is 863580.

We extracted the "publish year" and the "title" from the data set, and classified them into three collections according to the publish year, i.e., data set 1 (–1997 : 158706 records), data set 2 (1998 – 2007 : 348882 records), and data set 3 (2008– : 355992 records). In this experiment, we excluded words, each of whose frequency in all databases was less than 10. The size of Σ is 23224 (i.e., 23224 words in total).

[2] `http://dblp.uni-trier.de/xml`

Table 2. Comparison of ZDD-based method and Trie-Join

ZDD-based Method

Size of DB1	Size of DB2	Search Condition	Num. of Results	Exec.Time(sec)		
				ZDD Setting	Search	
100,000	100,000	add $\leq$ 1,delete $\leq$ 0	16,624	5.4	1.8	(Z1-1)
		add $\leq$ 0,delete $\leq$ 1	16,664		1.6	(Z1-2)
		add $\leq$ 1,delete $\leq$ 1	244,675		4.8	(Z1-3)
		add $\leq$ 2,delete $\leq$ 0	68,530		3.1	(Z1-4)
		add $\leq$ 0,delete $\leq$ 2	68,775		2.5	(Z1-5)
		add $\leq$ 2,delete $\leq$ 1	1,293,882		11.1	(Z1-6)
		add $\leq$ 1,delete $\leq$ 2	1,295,137		10.7	(Z1-7)
		add $\leq$ 2,delete $\leq$ 2	8,713,274		37.2	(Z1-8)
500,000	500,000	add $\leq$ 1,delete $\leq$ 0	401,823	27.9	12.6	(Z2-1)
		add $\leq$ 0,delete $\leq$ 1	405,813		10.2	(Z2-2)
		add $\leq$ 1,delete $\leq$ 1	5,922,847		31.5	(Z2-3)
		add $\leq$ 2,delete $\leq$ 0	1,649,671		19.2	(Z2-4)
		add $\leq$ 0,delete $\leq$ 2	1,681,117		16.4	(Z2-5)
		add $\leq$ 2,delete $\leq$ 1	31,163,852		84.4	(Z2-6)
		add $\leq$ 1,delete $\leq$ 2	31,554,584		86.1	(Z2-7)
		add $\leq$ 2,delete $\leq$ 2	210,967,890		368.7	(Z2-8)
1,000,000	1,000,000	add $\leq$ 1,delete $\leq$ 0	1,532,292	58.5	27.6	(Z3-1)
		add $\leq$ 0,delete $\leq$ 1	1,563,436		23.1	(Z3-2)
		add $\leq$ 1,delete $\leq$ 1	22,695,485		67.7	(Z3-3)
		add $\leq$ 2,delete $\leq$ 0	6,312,234		40.0	(Z3-4)
		add $\leq$ 0,delete $\leq$ 2	6,524,292		36.2	(Z3-5)
		add $\leq$ 2,delete $\leq$ 1	119,123,804		210.6	(Z3-6)
		add $\leq$ 1,delete $\leq$ 2	121,636,513		208.0	(Z3-7)
		add $\leq$ 2,delete $\leq$ 2	809,214,292		980.5	(Z3-8)

Trie-Join

Size of DB1	Size of DB2	Search Condition	Num. of Results	Exec.Time(sec)	
100,000	100,000	edit distance=1	78,315	1.1	(T1-1)
		edit distance=2	1,746,849	19.8	(T1-2)
500,000	500,000	edit distance=1	1,827,303	8.2	(T2-1)
		edit distance=2	40,789,678	159.1	(T2-2)
1,000,000	1,000,000	edit distance=1	6,701,562	23.1	(T3-1)
		edit distance=2	149,904,112	436.6	(T3-2)

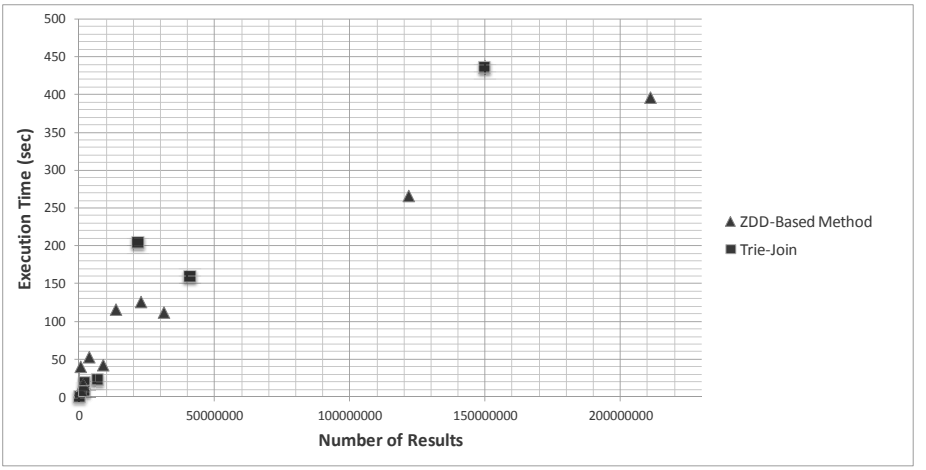


Fig. 5. Performance Comparison with ZDD-based Method and Trie-Join

Table 3. Experimental Results for DBLP database

Experiments for articles on DBLP (1) (– 1997 : 1998 – 2007)

Num.	Year	Title
1-a	1987	On The Complexity of Computable Real Sequences.
1-b	2001	Descriptive complexity of computable sequences
2-a	1982	Approximations for the waiting time distribution of the M/G/c queue.
2-b	2004	Mean Waiting Time Approximations in the G/G/1 Queue.
3-a	1979	On the connectivity of cayley graphs.
3-b	2005	Parameters of connectivity in (a, b)-linear graphs.
4-a	1993	An Affinity-Based Dynamic Load Balancing Protocol for Distributed Transaction Processing Systems.
4-b	2006	Dynamic Load Balancing Protocol for Locally Distributed Systems.
5-a	1989	Efficient monotone circuits for threshold functions.
5-b	2006	Monotone circuits for monotone weighted threshold functions.
6-a	1981	The reconstruction of maximal planar graphs. I. Recognition.
6-b	2004	A simple recognition of maximal planar graphs.

Experiments for articles on DBLP (2) (1998 – 2007 : 2008 –)

Num.	Year	Title
1-a	2005	k-Center problems with minimum coverage.
1-b	2008	Asymmetric k-center with minimum coverage.
2-a	2006	On complexity of multistage stochastic programs.
2-b	2008	On Stability of Multistage Stochastic Programs.
3-a	2006	A remote laboratory for electrical engineering education.
3-b	2011	Developing a remote laboratory for engineering education.
4-a	2006	Arboricity and tree-packing in locally finite graphs.
4-b	2008	Locally finite graphs and embeddings.
5-a	2005	Transforming semantics by abstract interpretation.
5-b	2009	Abstract interpretation of resolution-based semantics.
6-a	2006	PolicyUpdater: a system for dynamic access control.
6-b	2008	A privacy-aware access control system.

The experimental results are summarized in Table 3, where we can see the results for similarity joins between data sets 1 and 2, and between data sets 2 and 3. For example, 6-a (published in 1981) and 6-b (published in 2004) are extracted as similar research papers by different authors. Since both 6-a and 6-b involve the recognition of maximal planar graphs, these two research are intimately related with each other. On the other hand, the main theme in the paper of 1-a is the computational complexity of real sequences, while the theme in 1-b is the descriptive complexity for binary sequences. In this case, there seems to be no deep relationship among these two researches.

In fact, although we can recognize that other characteristics (e.g., research abstract) for research papers should be included if the results are practically in use, similar research activities between decades can be detected by using set similarity joins for item collections.

5.2 NSF Research Abstracts

We applied our system to the NSF data collection as another experiment, which consists of NSF Research Award Abstracts from 1990 to 2003. There are 129000

Table 4. Experimental Results for NSF database

Num.	Year	Title and Abstract
1-a	1992	Design of Parallel Algorithms – support for postdoctoral associate, – experimental computer science – developed for idealized parallel computers on real parallel computers, – load balancing techniques
1-b	1999	WORKSHOP: Parallel CFD'99 International Conference – computational fluid dynamics research on parallel computers, – parallel software system, – case studies from fluid dynamics, – early experiences on teraflops-class computers
2-a	1993	Constructing, Maintaining, and Searching Geometric Structures – construction and maintenance of geometric structures, – efficiently searching in such structures, – computational geometry unsolved problems, – dynamic maintenance of geometric structures, – computer graphics and computer vision, – sequential and parallel computation models
2-b	1999	Towards Simpler Algorithms in Computational Geometry – design and analysis of algorithms for large amounts of geometric data, – efficient algorithms for fundamental problems in computational geometry, – computer graphics and computer vision, – geometric optimization, – construction of basic geometric structures, – randomization, approximation, and techniques for correcting pessimistic worst-case analyses
3-a	1991	Undergraduate Computer Integrated Design Laboratory – establishment of an undergraduate computer integrated design laboratory, – development of a unified education program, – computer graphic simulation that gives insight into complex phenomena
3-b	1998	The Development of a Communication Networks Laboratory at Queens College – computer communication networks laboratory, – design and implementation of the Token Ring and Ethernet Local Area Networks, – undergraduate laboratory and an associated laboratory manual

entries (title, period, budget, abstract, area key, ...) in the database, and the bag-of-word data for the total records. The data set is disclosed in the UCI Machine Learning Repository[3].

We divided the data into "before 1996" and "after 1996" in this experiment, and we focused on entries whose abstracts include the word "computer". We created data sets, each of which consists of words in the title and the abstract. In this experiment, the size of Σ is 3444 (i.e., 3444 words in total).

Part of the experimental results are listed in Table 4, where $N^+ = 3$ and $N^- = 3$, which shows that relevant research activities are detected as well as those from the DBLP experiments.

For example, 1-a (1992) is a project for studying basic techniques for parallel algorithms, while 1-b (1999) is a workshop where more practical applications of parallel computing such as fluid dynamics are discussed. In this experiment, we

[3] NSF Research Award Abstracts in UCI Machine Learning Repository `http://archive.ics.uci.edu/ml/datasets/NSF+Research+Award+Abstracts+1990-2003`

can conclude that similar research projects can be detected as well as the case of DBLP application.

6 Conclusion

In this paper, we described a new approach to similarity joins for general item set collections under the constraints of the number of added and deleted items.

We introduced a matching algorithm based on Zero-suppressed Binary Decision Diagrams (ZDDs), which are special types of Binary Decision Diagrams (BDDs). ZDDs can represent huge databases efficiently, especially for sparse data collections. We developed efficient algorithms and pruning techniques for two ZDD structures.

We presented some experimental results from evaluating performance with other methods including Trie-Join, which is well known as an efficient implementation for similarity joins, although it is based on slightly different problem setting. As a result, our approach could achieve comparable results with Trie-Join for the problems we presented in this paper.

We also showed experimental results with actual huge data collections such as those from DBLP research titles and NSF research abstracts to demonstrate the availability in real applications.

Future works include extending our results in several directions such as :

- We intend to investigate various pruning techniques or filtering techniques exploited in other systems [1, 2, 6, 17] to adopt them in our system. Some pruning techniques [6, 17], such as length pruning and single branch pruning would also be especially helpful in our system.
- We need to classify items to various classes such as editable, essential, and requisite items, and we need to define transformation costs to replace these items. These extensions would make the results quite effective for real use.
- We intend to investigate more enhanced algorithms using ZDDs such as valuable ordering in ZDD construction or pruning techniques. In fact, it is well known that the valuable ordering in ZDDs is sensitive to performance in some cases.
- Sequence BDD [5, 8] offers considerable promise as a basic computation framework instead of ZDD in dealing with string similarity problems. Sequence BDD shares the same common sub-sequence as DAG structures, and can provide compact representations for manipulating sets of strings. We are now investigating the algorithms for sequential similarity joins using the sequence BDD framework.

References

1. Arasu, A., Ganti, V., Kaushik, R.: Efficient Exact Set-Similarity Joins. In: Proc. of 32nd International Conference on Very Large Data Bases, VLDB 2006 (2006)
2. Bayardo, R.J., Ma, Y., Srikant, R.: Scaling Up All Pairs Similarity Search. In: Proc. of 16th International Conference on World Wide Web (2007)

3. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Transactions on Computers 35(8) (1986)
4. Chaudhuri, S., Ganti, V., Kaushik, R.: A Primitive Operator for Similarity Joins in Data Cleaning. In: Proc. of 22nd International Conference on Data Engineering, ICDE 2006 (2006)
5. Denzumi, S., Yoshinaka, R., Minato, S., Arimura, H.: Efficient Algorithms on Sequence Binary Decision Diagrams for Manipulating Sets of Strings, Hokkaido University, TCS Technical Reports, TCS-TR-A-11-53 (2011)
6. Feng, J., Wang, J., Li, G.: Trie-join: a trie-based method for efficient string similarity joins. The VLDB Journal 21, 437–461 (2012)
7. Knuth, D.E.: The Art of Computer Programming. Bitwise Tricks & Techniques, vol. 4(1), pp. 117–126. Addison-Wesley (2009)
8. Loekito, E., Bailey, J., Pei, J.: A Binary decision diagram based approach for mining frequent subsequences. Knowledge and Information Systems 24(2) (2010)
9. Minato, S.: Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems. In: Proc. of 30th ACM/IEEE Design Automation Conference, DAC 1993 (1993)
10. Minato, S.-I.: VSOP (Valued-Sum-of-Products) Calculator for Knowledge Processing Based on Zero-Suppressed BDDs. In: Jantke, K.P., Lunzer, A., Spyratos, N., Tanaka, Y. (eds.) Federation over the Web. LNCS (LNAI), vol. 3847, pp. 40–58. Springer, Heidelberg (2006)
11. Minato, S.: Implicit Manipulation of Polynomials Using Zero-Suppressed BDDs. In: Proc. of IEEE The European Design and Test Conference (1995)
12. Neuhaus, M., Bunke, H.: An Error-tolerant Approximate Matching Algorithm for Attributed Planar Graphs and its Application to Fingerprint Classification. In: Fred, A., Caelli, T.M., Duin, R.P.W., Campilho, A.C., de Ridder, D. (eds.) SSPR&SPR 2004. LNCS, vol. 3138, pp. 180–189. Springer, Heidelberg (2004)
13. Oflazer, K.: Error-tolerant Finite-state Recognition with Applications to Morphological Analysis and Spelling Correction. Computational Linguistics 22(1) (1996)
14. Oflazer, K.: Error-tolerant Tree Matching. In: Proc. of 16th Conference on Computational Linguistics, COLING 1996 (1996)
15. Shimizu, K., Tsuda, K.: SlideSort: All Pairs Similarity Search for Short Reads. Bioinformatics 27(4) (2011)
16. Shirai, Y., Tsuruma, K., Sakurai, Y., Oyama, S., Minato, S.-I.: Incremental Set Recommendation Based on Class Differences. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part I. LNCS (LNAI), vol. 7301, pp. 183–194. Springer, Heidelberg (2012)
17. Wang, J., Feng, J., Li, G.: Trie-Join: Efficient Trie-based String Similarity Joins with EditDistance Constraints. In: Proc. of the VLDB Endowment, vol. 3(1-2) (2010)
18. Xiao, C., Wang, W., Lin, X.: Ed-join: an efficient algorithm for similarity joins with edit distance constraints. In: Proc. of VLDB Endowment, vol. 1(1) (2008)
19. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient Similarity Joins for Near Duplicate Detection. In: Proc. of 17th International Conference on World Wide Web (2008)

Keyword-Matched Data Skyline in Peer-to-Peer Systems

Khaled M. Banafaa, Ruixuan Li*, Kunmei Wen, Xiwu Gu, and Yuhua Li

School of Computer Science and Technology,
Huazhong University of Science and Technology,
Wuhan, Hubei 430074, P.R. China
kbanafaa@smail.hust.edu.cn,
{rxli,kmwen,guxiwu,idcliyuhua}@hust.edu.cn

Abstract. Data and storage management is turning to distributed due to the huge increase in data volumes. To satisfy users' requirements and preferences, advanced query operators, such as skyline, have been introduced and implemented. Skyline offers users with interesting objects, which has been explored in centralized, distributed and peer-to-peer (P2P) systems. However, keyword-matched skyline has not been considered in distributed and P2P systems. This paper introduces keyword-matched data skyline algorithms in P2P systems. Differing from other operators, skyline algorithms are devised to exploit its properties to reduce traversed peers for a query. By partitioning data space and using distributed hash tables (DHTs) and Bloom filters, we design new algorithms, Nk-sky and Ck-sky, to reduce the required traversed peers to answer keyword-matched data skyline queries. We apply the algorithms on Chord as an example of DHT overlay P2P systems. Experimental results show a significant reduction of traversed peers with the *Cover-set tuples* algorithm Ck-sky.

Keywords: Peer-to-peer system, skyline query, keyword-matched skyline.

1 Introduction

With the fast growing and huge volumes of data in the current Internet environment, advanced queries in distributed systems have been introduced, studied and designed. Skyline operator introduced by Börzsönyi in [1] is an example of such advanced queries. It recommends some interesting data objects to the users. The interesting objects are dominating objects that a user would be more interested in than the other objects. On the other hand, the processing limitations of centralized systems lead to distributed system designs. Different distributed systems have been suggested and used to solve the problems efficiently. In peer-to-peer

* Corresponding author.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 71–85, 2013.

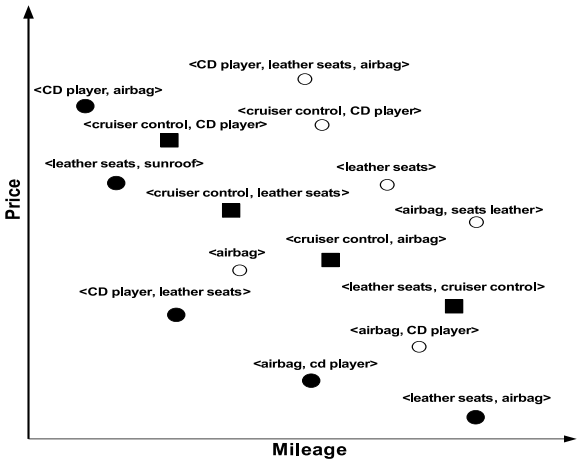


Fig. 1. An example of keyword-matched skyline

(P2P) systems, distributed hash tables (DHTs) have been discovered; and different network overlays have been designed (e.g. Content Addressable Network (CAN) [2], Chord [3] and Harmonic Ring (HRing) [4]). A query nature may result in a preference of network overlays to others. Skyline queries, like others, have been studied in different distributed systems and P2P system overlays (e.g. CAN as in [5], and BAlanced Tree Overlay Network (Baton) as in [6][7]).

A skyline point is a point that is not dominated by any other point in all dimensions. In general, the domination in one dimension is the user preference in that dimension (e.g. cheaper, lower mileage, and shorter distance). As an example, a user may be interested in buying a cheap used car with low mileage as shown in Fig. 1. The skyline query will return the black-filled rounded points in Fig. 1. However, a user may be interested in skyline for only points with some features. For example, a dealer may have many used cars with different features. Some cars come with cruiser controls, cd players, and/or airbags etc. A user, who is interested only in cars with cruiser control, may not be interested in the results that do not consider their preferences. The black-filled rectangular points shown in Fig. 1 are what that user expects. Another example may come from a user who is only interested in a restaurant with some dishes (e.g. sushi, seafood). Some restaurants may serve sushi but some restaurants may not. Thus, the traditional algorithms may result in skyline of no interest to the user. A skyline for only restaurants that serve sushi is what a user needs to see. These types of queries are called keyword-matched skyline queries.

Another example, consider some online scientific data analysis system where different participants publish their findings and use others' findings. Each participant may focus on different parts of the experiments. Keywords-matched skyline can help such scientists identify outstanding data and results of their interests. There could be large numbers of keywords-matched skyline queries triggered by

different scientists in a short time; thus, it is crucial to respond to such queries quickly using as few as possible of peers.

In distributed systems, traditional skyline algorithms do not consider keywords in their design. The work in [5] [6] [7], for example, considers values in dimensions to distribute tuples. These studies exploit prune-ability and incomparable partitions of skyline queries. Nevertheless, they do not consider keywords. Modifying them to satisfy users' requirements by ignoring undesired discovered points results in traversing unneeded peers. On the other hand, traditional keyword search query [8] and Napster [9] ignore skyline incomparability and prune-ability features. To exploit skyline pruning ability and incomparability as well as keyword search algorithms, this paper devise the keyword-matched skyline algorithms to combine keyword search and skyline algorithms to efficiently answer keyword-matched skyline. The algorithms keep some order of the peers to exploit prune-ability and incomparability. They also use DHT functions, Bloom filters and Cover-set features to keep track of points' and peers' keywords as explained in Section 3.

The contributions of this paper are as follows:

- Bloom filters are used to figure out the candidate peers for query keywords with cover-set tuples and nodes.
- Keyword-matched skyline algorithms are designed and implemented in P2P systems.
- Experiments have been carried out and show that the proposed approaches resulted in reduction of traversed peers while preserving progressiveness.

The rest of the paper is organized as follows. We first discuss related work in Section 2. Problem definition and algorithms are discussed in Section 3. Section 4 discusses experiments and our findings. We conclude our paper in Section 5.

2 Related Work

Börzsönyi's paper [1] was the first work to introduce skyline into databases. Block Nested Loop (BNL) uses a window to compare all points and discover the skyline points. Nearest neighbor (NN) [10] used R-trees and a *to-do* list to get skyline; branch and bound skyline (BBS) [11], however, uses R-trees and a heap to get rid of duplicates introduced in NN.

Because BNL, BBS, NN and the other centralized algorithms are not efficient for distributed and P2P systems, the distributed algorithms have been suggested. In [5], for example, data are distributed vertically. The traditional skyline is retrieved using a round-robin on the presorted attributes. In the feedback-based distributed skyline algorithm (FDS) [12], the coordinator iteratively contacts the other nodes providing a feed-back. Some algorithms [6][13] have converted multi-dimensional data into a single-data index and adapted it into P2P.

Other types of skyline queries (e.g. Subspace skyline, constrained skyline queries) in P2P systems have also been considered in literatures. For constrained skyline, data space is partitioned horizontally in the Distributed SkyLine query

(DSL) [14]. SkyFrame [7] uses greedy and relaxed skyline search on Baton (a balanced tree structure for peer-to-peer networks). In the Parallel Distributed Skyline (PaDSkyline) [15] and SkyPlan [16], the querying peer collects the minimum bounding rectangles (MBRs) from other peers, and with different measures (e.g. weighted edges and spanning trees), a plan is mapped for incomparable peers to work in parallel.

Skypeer [17] and Distributed Caching Mechanism (DCM) [18] are meant for subspaces skyline queries in distributed systems. A super-peer architecture is used for Skypeer. They defined the *extended skylines* which are collected by super-peers from the other peers. Queries are submitted to a super-peer which contacts the other super-peers for their subspace skyline. DCM explores caching. The results of subspaces queries are cached in peers using a distributed cache index (DCI) on Baton or Chord. The subspace queries use the DCI to forward next subspace skyline queries. Hose and Vlachou [19] have studied skyline processing in distributed systems in more details.

Even though the above algorithms answer skyline query efficiently, they are not designed to consider keyword-matched skyline. They all only use quantity values in the attributes to take advantage of skyline feature of prune-ability and incomparability. Attributes that may be boolean is not supported. A modification of those algorithms to satisfy user's requirements can reflect inefficiency.

Keyword-matched skyline has been introduced in [20]. It uses an R-tree. While R-trees are efficient for centralized systems, they are inefficient for P2P systems due to the heavy data volumes required for maintenance. In this paper, we investigate the keyword-matched data skyline in P2P systems using Chord [3] as our overlay. Other overlay structures may also be applied.

For keyword query, the traditional techniques either used centralized search as Napster [9], query broadcasting as Gnutella [21], or well-known naming such as Freenet [22]. They are not efficient for skyline queries because they do not exploit skyline properties, such as pruning ability and incomparability.

To the best of our knowledge, keyword-matched skyline has not been considered in distributed and P2P systems. Our aim is to minimize the visited peers in the network while preserving progressiveness.

3 Keyword-Matched Skyline Queries in P2P Systems

Some formalizations to keyword-matched skyline are presented in Section 3.1. Algorithms for keyword-matched skyline in P2P systems will be discussed in the next three Sections.

3.1 Problem Definition

In this subsection, some definitions are stated for keyword-matched skylines. Without loss of generality, we assume minimum values of attributes are preferred (e.g. cheaper is preferred to expensive, less mileage is preferred to high mileage, etc). For maximum value preferences, the inverse of the values can be used.

A tuple t in a d-dimensional space D^d is defined as <V,W> where V = $(v_1, v_2, ..., v_d)$ is a value vector of d-numerical values; and W = $(w_1, w_2, ..., w_k)$ is a set of k keywords for the tuple t. In addition, the value vector of a tuple t_i is denoted by $t_i.V$, while its set of keywords is denoted by $t_i.W$.

Definition 1. *A keyword-matched tuple to a query keyword ($Q_k(D^d, W)$). For a query q with a set of query keywords $q.W$, a tuple t is a keyword-match tuple to the query q if and only if $\forall w \in q.W$, $w \in t.W$.*

Definition 2. *Domination. Let t and t' be two tuples in D^d, where $t.V = (v_1, v_2, ..., v_d)$ and $t'.V = (u_1, u_2, ..., u_d)$. Then, t dominates t' ($t \prec t'$) if and only if $\forall i$, $v_i \leq u_i$ and $\exists i$, $v_i < u_i$. Conversely, t does not dominate t', denoted $t \not\prec t'$ if and if only $\exists i, v_i > u_i$.*

Definition 3. *Skyline Tuple. In skyline operator ($Q_s(D^d)$), a tuple t in D^d is a skyline tuple if and only if $\nexists t' \in D^d$; $t' \prec t$*

Definition 4. *A Keyword-matched Skyline Tuple. Let A be all keyword-matched tuples to a query q with keywords W. A tuple $t \in A$ is a keyword-matched skyline tuple to the query q if and only if $\forall t' \in A$; $\nexists t' \prec t$.*

Definition 5. *A keyword-matched skyline query ($Q_{ks}(D^d, W)$). Given a set of query keywords W and a dataset D^d, a keyword-matched skyline query denoted as $Q_{ks}(D^d, W)$, retrieves the set of skyline tuples whose each textual attribute contains all words of W. Thus, the following equivalent rule is true:*

$$Q_{ks}(D^d, W) \equiv Q_s(Q_k(D^d, W)) \tag{1}$$

Definition 6. *Let m be $min_x \in D(x_{min})$, Initial Skyline Peers P and Candidate Skyline Points M as*
$P = \{P_i | \exists x \in D_{P_i} \text{ such that } x_{min} = m\}$
$M = \{x | x \in D_P \wedge x_{min} = m\}$,
where $D_P = \bigcup_{P_i \in P} D_{P_i}$.

Theorem 1. *If S_D and S_M are the skylines of D and M respectively, then $S_M \subseteq S_D$ and $S_M \neq \phi$.*

Proof. From Definition 6 for M, for any $x \in S_M$, x is not dominated by any other point in M. Suppose $m' = min(x'_{min})\ \forall\ x' \in D - M$. As m < m', x can not be dominated by any $x' \in D - M$. Therefore, x is not dominated by any other point in D, i.e., $x \in S_D$. Thus, we have $S_M \subseteq S_D$. Since M is not empty, hence $S_M \neq \phi$.

Theorem 2. *A tuple t_1 with a minimum value $t_1.v_i$ in a dimension i can not be dominated by any point t_2 with a minimum value $t_2.v_j$ in any dimension j where $t_1.v_i < t_2.v_j$.*

Proof. Let's assume t_2 dominates t_1. Thus, $t_2.v_i \leq t_1.v_i$. Since $t_2.v_i \leq t_1.v_i \Rightarrow t_2.v_i < t_2.v_j$. This contradicts that $t_2.v_j$ is the minimum value of t_2. If $t_2.v_i$ is the minimum value of y, the condition of our theorem is not satisfied. On the other hand, if $t_2.v_i > t_1.v_i$, t_2 does not dominate t_1.

In the next three sections, we present three algorithms: a baseline (Ch-isky) which is based on a state of the art, and two new algorithms: Node-based keyword skyline (NK-sky) and Cover-based keyword skyline (Ck-sky). Definition 6 and Theorem 1 of isky[6] [13] are modified here to fit to our problem. Note that nodes and peers are used interchangeably to mean peers in this paper.

3.2 Chord-Based isky (Ch-isky)

The baseline approach is based on isky [6] [13]. It is applied on Chord overlay which we call Ch-isky. The algorithm is shown in Algorithm 1. As in isky, data values in each dimension are assumed to be in the period [0, 1]. The period [1, d+1] is distributed among peers with an equal continuous periods. Each peer is responsible of the next period in clockwise fashion starting with peer 0. A tuple is looked for its minimum value in all of its dimensions. The sum of the minimum value and the dimension it is found in is used to determine its destination peer when it is distributed. If the minimum value is found in more than one dimension, the lowest dimension is taken.

Once a query is triggered, it blindly travels through nodes exploiting pruneability. Lines [3-7] in Algorithm 1 require the querying peer to first broadcast the query to all Initial Skyline peers P (i.e. all peers that include the dimension values {1,2, ... ,d} in their period). In line 8, the skyline tuples are calculated using the volume filter and the min-max pruning ability to reduce calculations. In line 9, a new volume filter and a new min-max values are found using Equations 3 and 4. Line 10, the keyword-matched skyline tuples are sent to the querying peer and then returned to the user if these tuples exactly form a skyline. A volume filter and the min-max pruning are used. In lines [11-13], the query travels from a peer to the next peer in a clockwise fashion.

Our next two algorithms are based on the DHTs for the keywords and Bloom filters to minimize the number of candidate peers for keyword-matched skyline query. Even though Bloom filters can introduce few false positives, they do not affect the correctness of our algorithms as shown later.

Bloom Filters. To summarize membership in a set, a hash-based data structure called a Bloom filter [8] is used. A peer A can send its Bloom filter for its elements set E_A to another peer B with an element set E_B instead of sending its elements set. Thus, it reduces the amount of communication required for a peer to determine $A \cap B$. The membership test will never return false negatives, but it may return false positives with a tunable, predictable probability as shown in Equation 2 [8]. The results of the intersection in a peer E_B with E_A will contain all of the true intersection and may have also a few (false positive) hits that are only in E_B and not E_A. As the size of the Bloom filter increases, the number of false positives falls exponentially. Equation 2 predicts the probability of a false positive p_{fp} when an optimal choice of hash functions is given, and the Bloom filter bits m, and the number of elements in the set n are also given.

Algorithm 1. Ch-isky: Chord-based isky algorithm

```
1: Input: MinMax-filter, VDR, Querying-Peer, keywords
2: BEGIN
3: if QueryingPeer then
4:    for each Peer P includes an integer (1 to D) in its period do
5:       send Keyword-matched-Skyline-Query
6:    end for
7: end if
8: Calc-key-matched-sky-using-VDR-MinMax()
9: Calc-VDR-and-MinMax() /* using Equations 3, 4 */
10: Send-results-to-query-peer
11: if min(nextPeer) ≯ MinMax then
12:    send-query-to-next-peer(MinMax,VDR)
13: end if
14: END
```

Thus, to maintain a low false-positives probability, the Bloom filter size needs to be proportional to the number of represented elements.

$$p_{fp} = 0.6185^{m/n} \tag{2}$$

3.3 Node-Based Keyword-Matched Skyline Algorithm (Nk-sky)

In this section, we propose Nk-sky, a node-based keyword-matched skyline algorithm. For construction of Nk-sky, tuples are distributed to the peers according to the sum of their minimum values and dimension of the minimum value as explained above in Ch-isky. Each peer builds its keyword set. The peer's keyword set is the union of all points' keywords in a peer. It hashes each of its keywords using the DHT functions and sends them along with the peer's id to the keyword responsible peers. The keyword responsible peer hashes the node ID into the Bloom filter of that keyword.

The Nk-sky skyline query runs through two stages: 1) discovering candidate peers, and 2) skyline calculation.

1) Discovering Candidate Peers. Once the query is triggered, the querying peer hashes one keyword using the DHT function and sends the query to the responsible peer. The responsible peer gets the query peer Bloom filter for that query keyword and sends it to the next keyword's responsible peer. Each responsible peer receiving a Bloom filter would do the intersection with its keyword's Bloom filter and sends the results to the next responsible peer until all query keywords are processed. The last peer sends the results of the intersections to the querying peer. The results are the nodes with all query keywords. There might be few false positives but they will not affect correctness of the query results. For each keyword, it may require at most O(log n) lookup messages. Thus, for k query keywords, at most O($k$log n) lookup messages may be expected.

2) Skyline Calculations. At this stage, candidate peers, in addition to a few false positives, are known. The false positives will only affect traversing those peers unnecessarily but they will not affect the correctness of the results. The skyline query chooses peers from the candidates to broadcast to. A peer is chosen if it is the closest above or equal candidate peer to any peer responsible of the periods that include {1,2, . . .,d}. A peer c is *the closest above or equal candidate to another peer responsible of a value* i if and only if there is no other peer in the set that has a value v closer to i than any value in c and both v and c greater or equal to i. For example, suppose a peer p is responsible of period [1.8, 2.2) if p is in candidate peers, the query is sent to it because it includes 2. If p is not in the candidate peers, the query is sent to the closest candidate peer within the period [2.2, 3). Because there are at most d peers that include i ∈ {1,2, ... ,d}, the query broadcasts to a maximum of d candidate peers . All query processing is done in parallel.

In Nk-sky, a peer processes the keyword-matched skyline query and sends its results to the querying peer. The querying peer returns the results to the user if no future point can dominate them progressively. The processing peer also sends the query, a max-min value filter [6][13] and a Volume of Dominating Region (VDR) [23] to the closest above candidate peer in a clockwise fashion.

$$SF_{global} = min_{x \in S}(x_{max}) \tag{3}$$

In Equation 3, the min-max-value filter (SF_{global}) is used because we use minimum value as opposed to maximum value used in [6][13]. It is obtained from the already found skyline points (S). A peer is pruned if its minimum value is greater than SF_{global}. VDR, shown in Equation 4, is used to prune points within a peer. It is expected to prune more points than others due to its volume.

$$VDR_p = \prod_{i=1}^{d}(b_i - p_i) \tag{4}$$

where b_i is the maximum value in dimension i, and p_i is the value of p in dimension i. A peer and the following peers can be pruned if its minimum value is greater than SF_{global}.

Lemma 1. *Keyword-matched skyline results of Nk-sky are correct and complete.*

Proof. Correctness. Let t_1 and t_2 be two tuples with v_1^i and v_2^i be values in dimension i, respectively. Let t_1 dominates ($\prec$) t_2. If t_1 and t_2 have their minimum values in dimension i (i.e. v_1^i and v_2^i, respectively), t_1 is visited before t_2. Thus, t_2 will be pruned. On the other hand, lets t_1 and t_2 have their minimum values in different dimensions (i.e. v_1^i and v_2^j, respectively). From Theorem 2, in the algorithm, t_2 will not be declared as a keyword-matched skyline tuple until t_1 is seen. t_1 will prune t_2. Thus, no false skyline tuple will be produced.

Completeness. All tuples are checked. A peer is pruned if its minimum values are greater than the maximum value of a skyline tuple found so far. No tuple is pruned if not dominated.

Algorithm 2. Ck-sky: cover-based keyword-matched skyline algorithm

```
 1: Input: MinMax-filter, VDR, QueryingPeer, keywords, Peers-to-be-traversed
 2: BEGIN
 3: if QueryingPeer then
 4:    /* first stage */
 5:    Peers-to-be-traversed = get-candidate-peers-using-BloomFilter()
 6:    for all P ∈ Peers-to-be-traversed do
 7:       if p closest above or equal peer to an integer (1 to D)  then
 8:          send keyword-matched-skyline-query
 9:       end if
10:    end for
11: end if
12: /* Second stage */
13: Calc-keyword-matched-skyline-using-VDR-MinMax()
14: Calc-VDR-and-MinMax() /* using Equations 3, 4 */
15: Send-results-to-query-peer
16: nextPeer = closest-candidate-peer-in-an-increase-order
17: if  min(nextPeer) ⊁ MinMax then
18:    send-query-to-next-peer(MinMax,VDR)
19: end if
20: END
```

3.4 Cover-Based Keyword-Matched Skyline Algorithm (Ck-sky)

Using nodes instead of tuples in Nk-sky in Section 3.3 seems to be natural and more attractable to reduce false positives according to Equation 2. However, in reality, it is not the case for two reasons as shown in the experiments:

1) Node's keywords produced by OR-ing do not mean a node have a tuple with all query keywords.

2) Skyline query algorithms use pruning ability, which reduces the peers with query keywords as well as false positives from the candidates.

We propose Ck-sky, a cover-based keyword-matched skyline algorithm, as shown in Algorithm 2. In Ck-sky, a keyword-matched skyline query also runs in the same two stages of Nk-sky. However, we use tuple keywords instead of node keywords in Ck-sky. In lines [3-11], the first stage is presented. Thus, instead of sending only a node id to the keyword responsible peer, the tuple id is also sent. Bloom filter is used for the tuple ids. In the first stage, the querying peer gets the candidate peers using bloom filter as shown in line 3. In lines [6-10], instead of going to the dimension values peers discussed in Ch-isky, only the candidate peers closest to the dimension values peers are used to start the query. Lines [12-19]are responsible of the second stage. Line 13 calculates the skyline using the VDR and MinMax filters. The new VDR and MinMax are calculated in line 14 using Equations 3 and 4. The results are sent to the querying peer in line 15. The next peer to visit is calculated in line 16. In line 18, the query is sent to next peer if it is not pruned.

To reduce the number of tuples sent and maintained by the Bloom filter, we suggest Cover set.

Cover Tuple: A tuple t is said to be a cover for a tuple t' if and only if $\forall w \in t'.W$, w$\in t.W$.

Mutual Cover Tuple Set: All tuples that cover each other in set (i.e. they have the same keywords).

Cover Set: A cover set is the set of all tuples that have no cover in a node in addition to a tuple from each mutual cover tuple set.

In Ck-sky, a peer can send its Cover set instead of sending all tuples at the distribution phase. In the first stage, tuples are considered in the Bloom filter as opposed to nodes in Nk-sky. The second stage are done as in Nk-sky.

In the first stage, a query may need O($k$log n) lookup messages to reach to all k query keyword peers for their Bloom filters. It is, however, optimized by visiting keywords peers in order (in a clockwise order).

In the second stage, for an m candidate peers, a maximum of m jumps may be needed. Each jump may require O(log n) lookup messages. Due to the traversal order of our algorithms, the larger distance between candidate peers, the larger number of peers pruned. The portion of the m candidate peers that may be pruned, however, depends on the tuples in each peer.

Lemma 1 is also applicable for both Nk-sky and Ck-sky. The following Lemma is complement to Lemma 1.

Lemma 2. *Covered set does not affect the correctness of the keyword-matched skyline query.*

Proof. Let's assume a peer p with a tuple t. t is either in the Cover set of p or not. If t is in Cover set, then p will be included in the candidate peers. Let's assume t is not in the Cover set of p. This means there is t' that covers t if t is keyword-matched tuple to a query keyword q (Definition 1). Since ($\forall w \in q.W \rightarrow w \in t.W$) and ($\forall w' \in t.W \rightarrow w' \in t'.W$) $\Longrightarrow$ ($\forall w'' \in q.W \rightarrow w'' \in t'.W$). Thus, the peer p will be included in the candidate peers.

4 Performance Evaluation

In this section we evaluate our algorithms by checking the reduction of the visited peers for keyword-matched skyline queries. For a thorough investigation, we use synthetic datasets in our experiments to show the reduction in traversed peers for different variances (parameters). Table 1 summarizes the used parameter settings. The parameters include value distribution, dimensionality, cardinality, query keywords size, network size, and skew factor of the word distribution.

As in [1], for the experiments, we generated three types of synthetic datasets: (i) independent datasets, (ii) correlated datasets and (iii) anti-correlated datasets with 1000 distinct keywords.

Table 1. Parameter settings in the experiments

Parameters	Values
Cardinality(N) of tuples	100k, 200k, 400k, 600k, 800k, **1M**
Dimensionality	2, **3**, 4, 5
The number of query words (k)	1, **2**, 3, 4, 5
Zipf skew factor (θ)	0.0, 0.2, 0.4, 0.6, **0.8**, 1.0
Distribution(for values)	independent, correlated, anti-correlated
Tuple's keywords	6
Network size (no. of peers)	100, 1000, **2000**, 3000, 4000

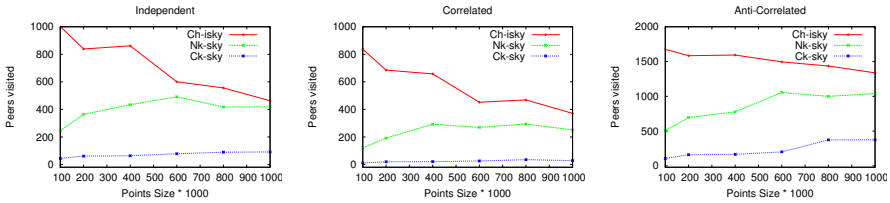


Fig. 2. Visited Peers vs. Number of Tuples

The three types of datasets are usually used for the evaluation of skyline. In the independent datasets, the values in each dimension is independent of each other; while in correlated datasets, the values are correlated as in a good grades of one student may come with more papers published. On the other hand, in anti-correlated datasets, a better value in one dimension will probably mean a worse value in the other dimensions as in the hotel example where a closer distance hotel to the beach will be a more expensive hotel.

The experiments were carried out on Intel(R) Core(TM) i3 CPU M350 (2.27 GHz), 3 GB RAM using Peersim-1.0.5 [24].

Our Ck-sky has shown to perform better than the other two algorithms in reducing the visited peers for a query.

4.1 The Effects of Cardinality

In this section, we show our findings with experiments to evaluate scalability with respect to dataset cardinality. Our experiments were done for various cardinalities with the range [100k,1M] and the default parameters shown in Table 1. Fig. 2 depicts our findings.

It shows that in all distribution of values (independent, correlated, and anti-correlated), Ck-sky preforms better. These results come from the fact that, in Ck-sky, visited peers only will probably contribute to the answer to the query. Using the minimum values to distribute tuples also contribute to minimizing the traversed peers by pruning peers that definitely can not contribute to the answer.

Nk-sky performs better than Ch-isky because in Nk-sky, only suspected peers are visited. Ch-isky, however, visits all peers in order of minimum values until it finds a pruning answer. In the independent and anti-correlated distributions,

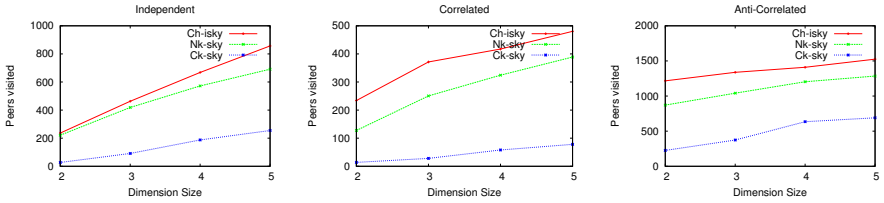


Fig. 3. Visited Peers vs. Dimensions

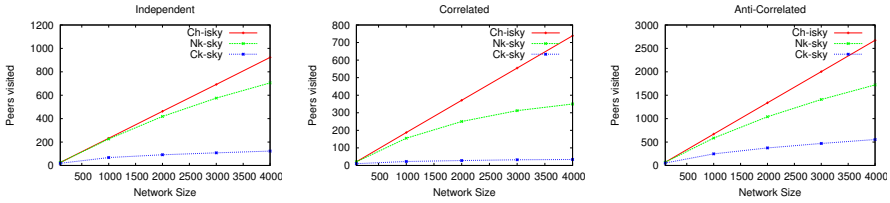


Fig. 4. Traversed Peers vs. Network Size

the curves for the Ch-isky and Nk-sky algorithms are higher than the curves in correlated. In correlated datasets, fewer skyline points are expected due to their correlations; and they can be found in early peers.

4.2 The Effects of Dimensionality

As the number of dimensions (attributes) increases, an increase in skyline points is expected. Larger variation among tuples is expected when higher dimensions are used. In this section, we show the effect of dimensions on the number of traversed peers in our algorithms. Our experiments are done with different dimensions [1D,2D ... 5D] with default values shown in Table 1. Fig. 3 goes along with our expectations that as dimensions increase, the number of traversed peers increases. Ck-sky is still better than the other algorithms. Ck-sky's increase as the dimensions increase is also expected as the pruning ability decreases with higher dimensions. In anti-correlated distribution, visited peers for all algorithms show a slight increase in traversed peers because the expected skyline points are also higher. All algorithms do better in correlated database distribution than the others because the correlation between values results in a better pruning. As stated earlier, Ck-sky visits only peers that will probably have skyline points.

4.3 The Effects of Network Size

As network size increases, we expect traversed peers to increase. How does the increase vary with our algorithms? In this section, we answer this question. Fig. 4 shows that the increase in Ch-isky is with a slope of one. The OR-ing used in the Nk-sky algorithm has more effects as the number of tuples in a peer decreases due to the increase of network size. Nk-sky becomes better than Ch-isky depending on the network size and type of data set. However, the affect is not as good as the

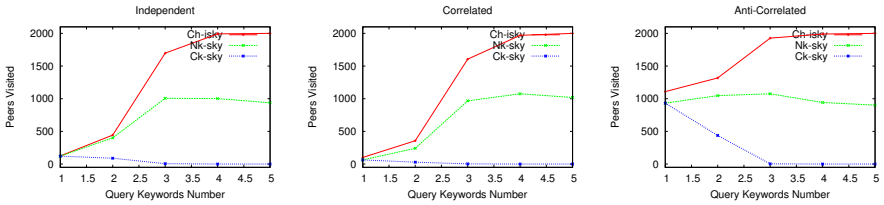


Fig. 5. Traversed Peers vs. Query Keyword Size

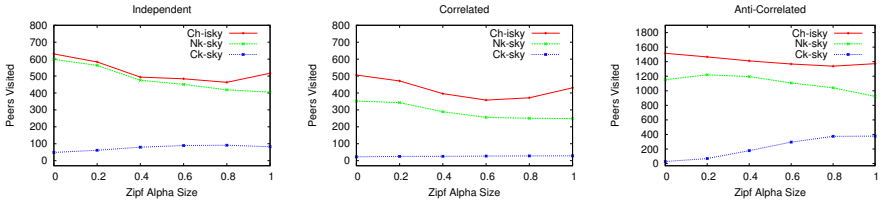


Fig. 6. Traversed Peers vs. Skewness

Ck-sky algorithm. Ck-sky traversed peers increase is expected as skyline tuples are distributed to more peers when network size increases. Thus, the increase in traversed peers in Ck-sky for the three types of data sets complies with [1]. Skyline tuples are more for the anti-correlated than the other models. Traversed peers increase in this model with our experiments for the same reason. Its effect is less for the other models due to the wrong peers traversed.

4.4 The Effects on the Size of Query Keywords

Query keywords can also affect traversed peers. As query keywords increase, fewer tuples would probably satisfy the query. This is also shown in Fig. 5 for Ck-sky. Due to the false peers that result from the other algorithms, more peers are visited. For Ck-sky and Nk-sky they will have the same number of visited peers when the query keyword is one because Nk-sky will not have false peers. False peers increase as we go farther. Ch-isky can probably have the same traversed peers at the beginning due to the probability that a peer will have tuples with fewer keywords than many keywords. The difference becomes big as the system does not have a tuple with the query keywords. However, Ck-sky can discover this at the first stage.

4.5 The Effects of Word Distribution

The results in Fig. 6 might be surprising because more traversed peers are expected in Ch-isky and Nk-sky because more tuples will satisfy the query. The reduction should not be surprising for two reasons: 1) Even though more peers are supposed to have satisfied tuples, they are contained in the unnecessary peers that are traversed with low zipf factor. 2) The skyline algorithms exploit pruning. Thus, because the probability of visiting a real candidate peer increases and

the prune-ability increases for the wrong peers. However, this reduction is not big due to the big probability of a wrong visited peer. The increase in traversed peers in Ck-sky is due to more tuples are expected to be in the solution. This is shown more obviously in anti-correlated distribution where more tuples are expected. For independent distribution, the number of traversed peers is slightly higher than correlated distribution due to the prune-ability and the results size.

5 Conclusion

This paper addresses keyword-matched skyline in peer-to-peer systems. Tuples may have keywords in addition to value (quantity) attributes. Keywords are boolean attributes that a tuple may have or may not. The designs of the traditional skyline algorithms only consider value attributes in all tuples. By specifying some keywords in the query, a user needs a skyline for only tuples with these keywords. It is called keyword-matched skyline. Modifying traditional skyline algorithms is inefficient. The traditional keyword algorithms are not good for keyword-matched skyline because they do not exploit prune-ability found in skyline queries.

In this paper, node and tuple-based algorithms are designed to solve keyword-matched skyline in peer-to-peer systems efficiently. The algorithms use DHTs functions and Bloom filters to minimize the number of traversed peers. Cover sets are also defined for peer's tuples to reduce false positives peers resulted from Bloom filters. Results show that tuple-based cover set (Ck-sky) algorithm performs better than the other algorithms. It considers only necessary tuples (cover set) in a node when keywords are hashed. Even though we studied the keyword-matched skyline in P2P systems in this paper, other issues could also be investigated, such as keyword-matched skyline in streams, subspaces and probability of keyword-matched skyline. Those issues are to be considered for the future work.

Acknowledgments. This work is supported by National Natural Science Foundation of China under Grants 61173170 and 70771043, National High Technology Research and Development Program of China under Grant 2007AA01Z403, and Innovation Fund of Huazhong University of Science and Technology under Grants 2012TS052 and 2012TS053.

References

1. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: The 17th International Conference on Data Engineering (ICDE 2001), pp. 421–432 (2001)
2. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: ACM SIGCOMM, pp. 161–172 (2001)
3. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM, pp. 149–160 (2001)
4. Zhuge, H., Chen, X., Sun, X., Yao, E.: Hring: a structured p2p overlay based on harmonic series. IEEE Transactions on Parallel and Distributed Systems 19(2), 145–158 (2008)

5. Balke, W.-T., Güntzer, U., Zheng, J.X.: Efficient distributed skylining for web information systems. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 256–273. Springer, Heidelberg (2004)
6. Cui, B., Chen, L., Xu, L., Lu, H., Song, G., Xu, Q.: Efficient skyline computation in structured peer-to-peer systems. IEEE Transactions on Knowledge and Data Enginering 21(7), 1059–1072 (2009)
7. Wang, S., Vu, Q.H., Ooi, B.C., Tung, A.K.H., Xu, L.: Skyframe: a framework for skyline query processing in peer-to-peer systems. VLDB Journal 18(1), 345–362 (2009)
8. Mullin, J.: Optimal semijoins for distributed database systems. IEEE Transactions on Software Engineering 16(5), 558–560 (1990)
9. http://www.napster.com/
10. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: VLDB, pp. 275–286 (2002)
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. Zhu, L., Tao, Y., Zhou, S.: Distributed skyline retrieval with low bandwidth consumption. IEEE Transactions on Knowledge and Data Enginering 21(3), 384–400 (2009)
13. Chen, L., Cui, B., Lu, H., Xu, L., Xu, Q.: isky: Efficient and progressive skyline computing in a structured p2p network. In: The 28th International Conference on Distributed Computing Systems (ICDCS 2008), pp. 160–167 (2008)
14. Wu, P., Zhang, C., Feng, Y., Zhao, B.Y., Agrawal, D., El Abbadi, A.: Parallelizing skyline queries for scalable distribution. 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. 112–130. Springer, Heidelberg (2006)
15. Chen, L., Cui, B., Lu, H.: Constrained skyline query processing against distributed data sites. IEEE Transactions on Knowledge and Data Enginering 23(2), 204–217 (2011)
16. Rocha-Junior, J., Vlachou, A., Doulkeridis, C., Nørvåg, K.: Efficient execution plans for distributed skyline query processing. In: EDBT, pp. 271–282 (2011)
17. Vlachou, A., Doulkeridis, C., Kotidis, Y., Vazirgiannis, M.: SKYPEER: Efficient subspace skyline computation over distributed data. In: ICDE, pp. 416–425 (2007)
18. Chen, L., Cui, B., Xu, L., Shen, H.T.: Distributed cache indexing for efficient subspace skyline computation in P2P networks. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010, Part I. LNCS, vol. 5981, pp. 3–18. Springer, Heidelberg (2010)
19. Hose, K., Vlachou, A.: A survey of skyline processing in highly distributed environments. The VLDB Journal–The International Journal on Very Large Data Bases 21(3), 359–384 (2012)
20. Choi, H., Jung, H., Lee, K., Chung, Y.: Skyline queries on keyword-matched data. Information Sciences (2012), doi:10.1016/j.ins.2012.01.045
21. http://rfc-gnutella.sourceforge.net/developer/stable/index.html
22. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. In: Federrath, H. (ed.) Anonymity 2000. LNCS, vol. 2009, pp. 46–66. Springer, Heidelberg (2001)
23. Huang, Z., Jensen, C., Lu, H., Ooi, B.: Skyline queries against mobile lightweight devices in manets. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006), pp. 66–66 (2006)
24. http://peersim.sourceforge.net/

Adaptive Query Scheduling in Key-Value Data Stores

Chen Xu[1], Mohamed Sharaf[2], Minqi Zhou[1],
Aoying Zhou[1], and Xiaofang Zhou[2]

[1] East China Normal University, Shanghai, China
[2] The Universtiy of Queensland, Brisbane, Australia
chenxu@ecnu.cn, {m.sharaf,zxf}@uq.edu.au,
{mqzhou,ayzhou}@sei.ecnu.edu.cn

Abstract. Large-scale distributed systems such as Dynamo at Amazon, PNUTS at Yahoo!, and Cassandra at Facebook, are rapidly becoming the data management platform of choice for most web applications. Those key-value data stores rely on data partitioning and replication to achieve higher levels of availability and scalability. Such design choices typically exhibit a trade-off in which data freshness is sacrificed in favor of reduced access latencies. Hence, it is indispensable to optimize resource allocation in order to minimize: 1) query tardiness, i.e., maximize Quality of Service (QoS), and 2) data staleness, i.e., maximize Quality of Data (QoD). That trade-off between QoS and QoD is further manifested at the local-level (i.e., replica-level) and is primarily shaped by the resource allocation strategies deployed for managing the processing of foreground user queries and background system updates. To this end, we propose the *AFIT* scheduling strategy, which allows for selective data refreshing and integrates the benefits of SJF-based scheduling with an EDF-like policy. Our experiments demonstrate the effectiveness of our method, which does not only strike a fine trade-off between QoS and QoD but also automatically adapts to workload settings.

1 Introduction

A fundamental requirement in modern web applications is to consistently meet the user's expectations for response time as expressed by a Service Level Agreement (SLA). An example of a simple SLA is a web application guaranteeing that it will provide a response within 300ms for 99.9% of its requests for a peak client load of 500 requests per second [6].

Recently, distributed key-value data stores have emerged as the data management platform supporting such Web applications (e.g., Dynamo [6], PNUTS [5], Cassandra [9], etc.). Key-value data stores present a highly-replicated data model based on simplified primary-key data access. However, achieving serializability over such a globally-replicated distributed system is very expensive and often unnecessary [5]. In particular, web applications expect and tolerate the weaker levels of consistency achieved via optimistic replication [11].

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 86–100, 2013.

While providing weaker levels of consistency allows for high availability (i.e., low latency), this often comes at the expense of reduced data freshness where user queries might access stale data. Recent work (e.g., [1,14]) has studied that consistency/latency trade-off at a global-level (i.e., system-level), often in the light of CAP Theorem and quorum protocols. The same tradeoff is further manifested at the local-level (i.e., each replica node) and is primarily influenced by the resource scheduling strategy, which is responsible for allocating the limited computational resources at each replica for the processing of incoming: 1) foreground user-queries, and 2) background system-updates [13]. Hence, it is indispensable to optimize resource allocation in order to minimize: 1) query tardiness, i.e., maximize Quality of Service (QoS), and 2) data staleness, i.e., maximize Quality of Data (QoD).

In a key-value data stores, each data object is accessed by its key leading to a clear relationship between the arriving queries and their corresponding pending updates, which offers a valuable opportunity to achieve the aforementioned optimization goals. For instance, the *On Demand (OD)* scheduling mechanism is well suited to achieve that goal since it couples the execution of the pending updates together with the arriving user requests where all the data items read by a certain query are refreshed on demand before the execution of that query [3]. Clearly, however, OD might sacrifice QoS in order to maximize QoD. Unlike OD, the *Freshness/Tardiness (FIT)* scheduling policy might selectively skip applying some of the pending updates in order to minimize query tardiness (i.e., maximize QoS) [15].

FIT is based on the classical Shortest Job First (SJF) scheduling policy, which is well-known to perform reasonably well under most workload settings. SJF, however, is oblivious to deadlines (i.e., SLAs), which leaves room for significant improvements when it is integrated with other deadline-aware policies such as Earliest Deadline First (EDF) [7]. Motivated by that, in this work we propose the *Adaptive Freshness/Tardiness (AFIT)* method for scheduling queries and updates in key-value data stores. AFIT exhibits the following desirable properties: 1) it exploits the coupling between the pending queries and corresponding updates in order to optimize resource usage, 2) it employs a selective policy in applying pending updates in order to balance the trade-off between QoS and QoD, and in turn maximize the overall system utility, and 3) it complements and integrates the SJF-based scheduling with an EDF-like policy, which allows it to adapt to a wide spectrum of workload settings.

Our experimental evaluation shows that AFIT outperforms the existing strategies in terms of striking a fine balance between QoS and QoD, which leads to maximizing the overall system utility. In addition, those gains are maintained at any feasible workload setting due the adaptivity feature of AFIT, which allows it to dynamically and automatically adjust to those settings.

The remainder of this paper is organized as follows. Firstly, Section 2 describes the system model. Then, Section 3 describes our proposed solution, i.e., AFIT, and Section 4 shows the experimental results. Finally, Section 5 concludes our paper.

2 System Model

Our system model is based on modern distributed data management platforms, in which each data item, as well as its replicas, are in the form of key-value pairs. Under the weak consistency model, issued queries can access any replica while updates are propagated in the background [6]. Hence, a user query might access stale data items if any updates are pending for this item. In this section, we discuss our system model including the data access operations (i.e., queries and updates) as well as the performance metrics (i.e., query tardiness and data staleness).

2.1 Queries

Without loss of generality, a query in our model is a read-only data access operation. For instance, most web applications such as browsing the recent messages on a social network web site, involve executing a large number of such read-only queries. In response to an API-level request, such as `get()` in Dynamo, the object replicas associated with the query key are located in the storage system and the corresponding value is returned.

In this paper, Q_i notates the i-th query to arrive into the system, which is associated with an arrival time A_i and a cost C_i. That cost is basically the amount of time needed to access the requested data item and it incorporates both CPU and I/O costs, which are statistically estimated by monitoring the execution of previous queries over a reasonable time window.

2.2 Updates

Under the weak consistency model, updates are typically propagated in the background to different replicas. Those updates are not applied immediately but they are queued to refresh the corresponding data item. For instance, a user might update some of her profile information on Facebook, but her friends overseas may still see the old information. Generally, each update refreshes one data item. Requests such as `set()` in PNUTS and `put()` in Dynamo are included in the high-level API to update the value of a data item given its key.

In this work, we also assume that updates are *blind* [10] such that a newly arriving update automatically makes any pending updates on the same data item invalid. That is, to get the newest data, it is only needed to apply the latest update rather than the intermediate updates. Hence, in our system model, U_i is the cost of applying the latest update on the data item accessed by Q_i. Similar to query processing costs, the cost of applying an update includes both the CPU and I/O costs and is typically evaluated by monitoring the processing of previous updates over time.

2.3 Metrics

In order to quantify the user-perceived Quality of Service (QoS) and Quality of Data (QoD), we discuss two metrics, namely tardiness and staleness.

Tardiness. To specify the user's requirement on response time (i.e., QoS), each query is associated with a tardiness deadline, which is formally defined as follows:

Definition 1. *Tardiness deadline TD_i for query Q_i is defined as $TD_i = A_i + \gamma_i$, where A_i is the arrival time of Q_i and γ_i is the tardiness tolerance of Q_i.*

At time t, if the tardiness deadline TD_i of a query Q_i has been missed, the tardiness is calculated as $t - TD_i$. Otherwise, that tardiness is zero. Therefore, the tardiness of Q_i can be represented as:

$$max(0, t - TD_i) \tag{1}$$

Staleness. To specify the user's requirement on data freshness (i.e., QoD), each query is associated with a staleness deadline, which is formally defined as follows:

Definition 2. *Staleness deadline SD_i for query Q_i is defined as $SD_i = R_i + \delta_i$, where R_i is the arrival time of the first unapplied update on the data item accessed by Q_i and δ_i is the staleness tolerance of Q_i.*

Like tardiness, we use time-based staleness measure which is especially useful in a distributed environment [8]. At time t, if the staleness deadline SD_i of query Q_i has passed, the staleness is calculated as $t - SD_i$. Otherwise, the staleness is zero. Therefore, the staleness of Q_i can be represented as:

$$max(0, t - SD_i) \tag{2}$$

2.4 Problem Statement

Given the definitions of tardiness in Eq. 1 and staleness in Eq. 2, the total penalty of Q_i at time t in terms of both QoS and QoD penalty is:

$$P_i(t) = W_i[\lambda_i max(0, t - TD_i) + (1 - \lambda_i) max(0, t - SD_i)] \tag{3}$$

where W_i is the query weight, λ_i is the QoS fraction, and $1 - \lambda_i$ is the QoD fraction. Here, W_i represents the inter-query importance, whereas λ_i and $1 - \lambda_i$ represent the intra-query QoS and QoD preference, respectively. The QoS and QoD components in Eq. 3 are depicted in Figure 1 and Figure 2, respectively.

The total penalty incurred by a query Q_i is evaluated at its finish time F_i, i.e., $P_i(F_i)$. For a number of queries, the *average total penalty* is defined as:

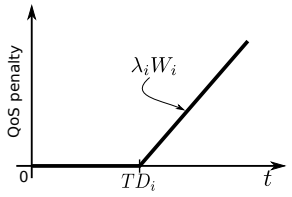


Fig. 1. QoS Penalty

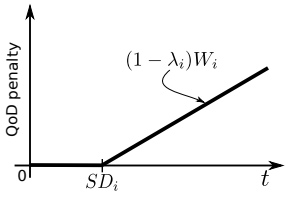


Fig. 2. QoD Penalty

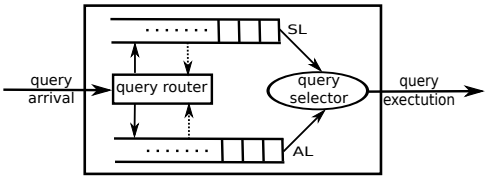


Fig. 3. Framework of AFIT Scheduler

Definition 3. *The average total penalty for N queries is:* $\frac{1}{N}\sum_{i=1}^{N} P_i(F_i)$*, where* F_i *is the finish time of* Q_i*.*

In this work, our goal is to find an optimal strategy for query scheduling to minimize the average total penalty. The following sections describe our proposed solution towards addressing this problem.

3 The AFIT Query Scheduling

Scheduling strategies such as *On Demand (OD)* [3] and *Freshness/Tardiness (FIT)* [15] are well suited to optimize resource allocation in key-value data stores so that to maximize both QoS and QoD. However, OD might sacrifice QoS in order to maximize QoD since it always refreshes a data items before it is accessed by a query. Although FIT allows to selectively skip some of the pending updates to balance QoS and QoD, it employs a scheduling policy that is purely based on the classical Shortest Job First (SJF) which is oblivious to deadlines.

In general queueing systems, it is well-known that Earliest Deadline First (EDF) works reasonably well in the presence of SLAs except for systems that exhibit high-utilization or fairly tight SLAs, in which case SJF typically outperforms EDF [7,12]. This exact trade-off between EDF and SJF motivates us to further study the performance of query scheduling policies in key-value data stores. In particular, in this work we propose *Adaptive Freshness/Tardiness (AFIT)*, an adaptive policy for the scheduling of queries and updates in a key-value data store. Like FIT, our AFIT scheduler employs a selective policy in applying pending updates so that to balance the trade-off between QoS and QoD, and in turn minimize the overall system penalty. Differently, however, AFIT employs a hybrid scheduler that integrates and extends both SJF and EDF into the scheduling decision so that to dynamically adapt to variabilities in the workload settings.

As shown in Figure 3, the AFIT scheduler maintains two query lists: 1) *Apply List (AL)*; and 2) *Skip List (SL)*. In both lists, each query is assigned a priority value according to a weighted variant of SJF which considers the trade-off between benefit and cost. In the context of our work and according to our problem statement presented in Section 2.4, that benefit is expressed in terms of minimizing the penalty paid by the system for violating the QoS and QoD requirements. Hence, the higher the penalty, the higher the weight (i.e., priority). Accordingly, those priorities are computed as follows:

- *Apply List (AL)*: A query Q_i is added to this list when AFIT decides to apply the latest pending update to the data item accessed by Q_i before it is executed. Accordingly, each query in this list would access a fresh data item. Hence, AFIT needs to only consider the fraction of weight pertaining to QoS (i.e., λ_i) and the priority of each query Q_i in AL is computed as: $p_i = \frac{\lambda_i W_i}{C_i + U_i}$.
- *Skip List (SL)*: A query Q_i is added to this list when AFIT decides to skip the latest pending update to the data item accessed by Q_i before it is executed. Accordingly, a query in this list might access a stale data item. Hence, AFIT needs to consider the total weight of QoS and QoD (i.e., W_i) and the priority of each query Q_i in SL is computed as: $p_i = \frac{W_i}{C_i}$.

In order to make the decision in which list to place a new query, AFIT employs a query router component (as shown in Figure 3). The query router decides the placement of a new query according to the critical condition described in Section 3.1. In addition, the query router also checks the existing queries in the two lists and reallocates the queries for which that critical condition has been violated. Figure 3 also shows the query selector component which, at each scheduling point, decides the query to be scheduled for execution. The query selector chooses between the two queries with the highest priority in the AL and SL lists and its decision is based on an integration of the SJF and EDF policies as described in Section 3.2.

3.1 Query Routing

The query routing component of AFIT is responsible for assigning each newly arriving query to either the AL or the SL list. Central to that decision is estimating the *slack* available for each query. That estimation is dependent on the candidate target list (i.e., AL or SL) as well as the performance metrics under consideration (i.e., tardiness and staleness). Before further discussing the details of query routing, the different definitions of slack are given below:

Definition 4. *The **AL tardiness slack** time ts_i^a of query Q_i is the maximum amount of time that Q_i can wait before it misses its tardiness deadline TD_i. This excludes the cost for applying the last pending update to the accessed data item (i.e., U_i) and the cost for query execution (i.e., C_i). Specifically, $ts_i^a = TD_i - (t + C_i + U_i)$ where t is the current system time.*

Definition 5. *The **SL tardiness slack** time ts_i^s of query Q_i is the maximum amount of time that Q_i can wait before it misses its tardiness deadline TD_i. This excludes the cost for query execution (i.e., C_i). Specifically, $ts_i^a = TD_i - (t + C_i)$ where t is the current system time.*

Clearly, for any query Q_i, $ts_i^s \geq ts_i^a$ at any time t. That is, the available slack time to meet the tardiness deadline by skipping an update (i.e., ts_i^s) is always greater than or equal the available slack time when an update is to be applied

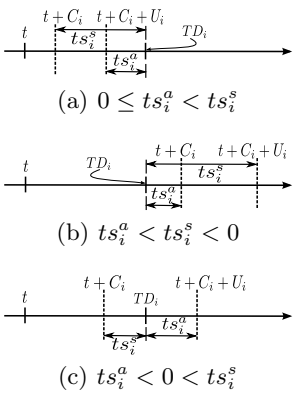


Fig. 4. Potential Tardiness Increased

(i.e., ts_i^a). However, if an update is to be skipped, it is essential to measure the available time to meet the staleness deadline. That is, the accessed data item meets the pre-specified staleness tolerance, which is defined as:

Definition 6. *The* ***staleness slack*** *time ss_i of query Q_i is the maximum amount of time that Q_i can wait before it misses its staleness deadline SD_i. This excludes the cost for query execution (i.e., C_i). Specifically, $ss_i = SD_i - (t + C_i)$ where t is the current system time.*

Intuitively, a query Q_i is inserted into AL if the penalty incurred by applying the latest pending update is greater than the one incurred by skipping it, and vice versa. Based on the definitions above, the critical condition employed by the query router is represented as: *query Q_i is inserted into AL, if either Eq. 4a or Eq. 4b meets.*

$$\begin{cases} ts_i^a \geq 0 & \text{(4a)} \\ \lambda_i[U_i - max(0, ts_i^s)] \leq (1 - \lambda_i) max(0, -ss_i) & \text{(4b)} \end{cases}$$

Otherwise, Q_i would be inserted into SL.

The equations above basically exploit the estimated potential penalty so that to choose the appropriate list assignment for each incoming query. Only a QoS penalty is incurred if the pending update is applied, whereas both QoS and QoD penalties are expected if that update is skipped.

To further explain the intuition underlying the AFIT query router, lets consider the different valid relationships between ts_i^a and ts_i^s as shown in Figures 4(a), 4(b) and 4(c).

1. $0 \leq ts_i^a < ts_i^s$: Figure 4(a) shows that even if Q_i waits for data refreshing, i.e., no QoD penalty, it still meets its tardiness deadline, i.e. no QoS penalty. Hence, Q_i should be inserted into AL since Eq. 4a is satisfied in that case.
2. $ts_i^a < ts_i^s < 0$: Figure 4(b) shows the case in which Q_i will miss its tardiness deadline TD_i, even if the pending update is skipped. Specifically, if the

pending update is skipped, Q_i misses TD_i by $t + C_i - TD_i$ and misses SD_i by $max(0, t+C_i-SD_i)$, i.e., $max(0, -ss_i)$. Hence, the total incurred penalty P_s is $\lambda_i W_i(t + C_i - TD_i) + (1 - \lambda_i)W_i max(0, -ss_i)$. To the contrary, If Q_i waits for the accessed data item to be updated, the total penalty P_a will only include the QoS component and is estimated as: $\lambda_i W_i(t + C_i + U_i - TD_i)$. Therefore, Q_i should be inserted into AL if $P_a \leq P_s$, i.e.,

$$\lambda_i U_i \leq (1 - \lambda_i) max(0, -ss_i) \tag{5}$$

3. $ts_i^a < 0 < ts_i^s$: Figure 4(c) shows the final case in which Q_i misses its tardiness TD_i if the data item is refreshed, whereas it meets TD_i if the data item is left stale. Similar to previous case, we compare the penalty incurred by applying or skipping the pending update. If Q_i skips the update, only QoD penalty is incurred and then the total penalty P_s equals to $(1 - \lambda_i)W_i max(0, -ss_i)$. To the contrary, if Q_i applies the update, only QoS penalty exists and the total penalty P_a is $\lambda_i W_i(t+C_i+U_i-TD_i) = \lambda_i W_i(U_i - ts_i^s)$. Hence, Q_i is inserted into AL if $P_a \leq P_s$, i.e.,

$$\lambda_i(U_i - ts_i^s) \leq (1 - \lambda_i) max(0, -ss_i) \tag{6}$$

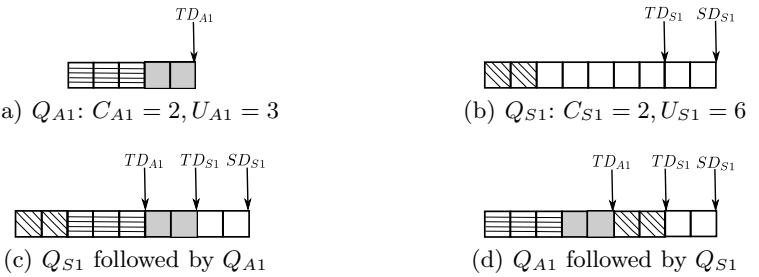
In the AFIT query router, the first case above is handled by Eq. 4a, whereas the second and third cases are uniformly handled by Eq. 4b. In particular, a newly arriving query is dispatched to either AL or SL according to those two equations. Moreover, at each scheduling point, the query router maintains the two lists as follows: (1) queries in SL that satisfy Eq. 4a or 4b are moved into AL; and (2) queries in AL that satisfy Eq. 4a and 4b are moved into SL.

3.2 Query Selection

Clearly, the query with the highest priority in AL should be executed prior to the other queries in that list. Similarly, the query with the highest priority in SL should be the first one to be executed in that list. Hence, at each scheduling point, the task of the query selector is to compare the head query in AL (say Q_{A1}) to the head query in SL (say Q_{S1}) and the winner is selected for execution (as illustrated in Figure 3).

One option is to compare those two queries according to the weighted SJF priority p_i assigned to them when first inserted in AL and SL, respectively. This is similar to the approach followed by FIT [15], which is purely based on cost-benefit analysis and has the drawback of ignoring the respective deadlines of those two queries. To illustrate the impact of that drawback, consider the following example:

Example 1. Figure 5 shows two queries Q_{A1} and Q_{S1}, for which we assume equal QoS fractions for the two queries (i.e., $\lambda_{A1} = \lambda_{S1}$) as well as equal weights (i.e., $W_{A1} = W_{S1}$). Computing the priority of each query according to weighted SJF (as described earlier in the beginning of Section 3) results in $p_{A1} = \frac{\lambda_{A1} W_{A1}}{C_{A1}+U_{A1}}$ and $p_{S1} = \frac{W_{S1}}{C_{S1}}$. Given the query and update costs illustrated in Figure 5, in



(a) Q_{A1}: $C_{A1} = 2, U_{A1} = 3$ (b) Q_{S1}: $C_{S1} = 2, U_{S1} = 6$

(c) Q_{S1} followed by Q_{A1} (d) Q_{A1} followed by Q_{S1}

Fig. 5. Motivated Example for Query Selection

this example $p_{S1} > p_{A1}$ and a simple cost-benefit comparison would result in Q_{S1} being executed before Q_{A1}. However, that exact execution order results in a system penalty (as shown in Figure 5(c)), whereas the reverse order, in which Q_{A1} is executed first, incurs no penalty (as shown Figure 5(d)).

Example 1 calls for an alternative scheduling approach which takes deadlines into consideration. In this work, AFIT leverages the different available slacks (as defined in Section 3.1) to extend that basic cost-benefit comparison and inject EDF-like deadline-aware scheduling into the query selector decision. In particular, the AFIT query selector employs the following criterion for query execution: *Q_{S1} is executed first if it meets the following condition:*

$$W_{A1}\lambda_{A1}[max(0, C_{S1} - max(0, ts^a_{A1}))] < W_{S1}[\lambda_{S1} max(0, C_{A1} + U_{A1} - max(0, ts^s_{S1})) + (1 - \lambda_{S1}) max(0, C_{A1} + U_{A1} - max(0, ss_{S1}))] \quad (7)$$

Otherwise, Q_{A1} is the one scheduled for execution.

Eq. 7 compares the estimated penalty achieved by two different sequences: 1) $Q_{S1} \rightarrow Q_{A1}$ (i.e., Q_{S1} followed by Q_{A1}), and 2) $Q_{A1} \rightarrow Q_{S1}$ (i.e., Q_{A1} followed by Q_{S1}). Accordingly, it selects for execution the query that yields the lower penalty to the system (if any). Those two alternative execution orders are illustrated in Figures 6 and 7 and are further discussed next.

- $Q_{S1} \rightarrow Q_{A1}$: We estimate the tardiness experienced by Q_{A1} due to waiting for Q_{S1}'s execution according to the cases shown in Figure 6 and listed below:
  - $ts^a_{A1} < 0$: Figure 6(a) shows the case in which Q_{A1} has already missed its tardiness deadline TD_{A1} regardless of the execution order. Hence, waiting for Q_{S1} to finish execution increases Q_{A1}'s tardiness by C_{S1}.
  - $0 < ts^a_{A1} < C_{S1}$: Figure 6(b) shows that Q_{A1} has enough slack that it will incur no tardiness if it was to be executed before Q_{S1}. However, waiting for Q_{S1} to finish execution increases Q_{A1}'s tardiness by $C_{S1} - ts^a_{A1}$.
  - $0 < C_{S1} < ts^a_{A1}$: Figure 6(c) shows that Q_{A1} has large enough slack that it will incur no tardiness regardless of its execution order. Hence, waiting for Q_{S1} to finish execution leads to no increase in Q_{A1}'s tardiness.

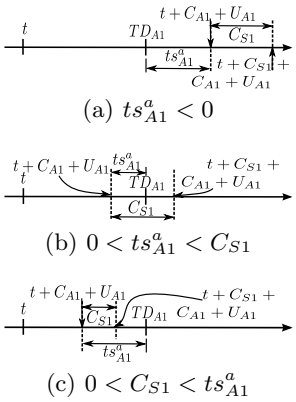


Fig. 6. Q_{A1}'s Tardiness Incurred by Q_{S1}

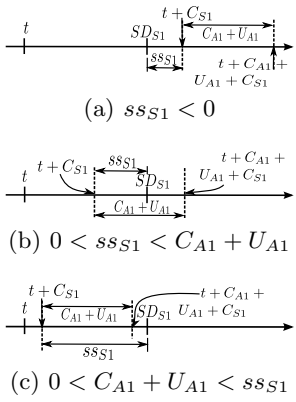


Fig. 7. Q_{S1}'s Staleness Incurred by Q_{A1}

Hence, the increase in total penalty $P_{S \to A}$ due to Q_{S1}'s execution is computed as: $W_{A1}\lambda_{A1} max(0, C_{S1} - max(0, ts^a_{A1}))$, which is the left-hand component of Eq. 7.

– $Q_{A1} \to Q_{S1}$: Similar to the first sequence, the increase in QoS penalty experienced by Q_{S1} due to Q_{A1}'s execution is: $W_{S1}\lambda_{S1} max(0, C_{A1} + U_{A1} - max(0, ts^s_{S1}))$. In addition, the increase in Q_{S1}'s staleness is discussed in following three cases:

- $ss_{S1} < 0$: Figure 7(a) shows the case in which Q_{S1} has already missed its staleness deadline SD_{S1} regardless of the execution order. Hence, waiting for Q_{A1} to finish execution increases Q_{S1}'s staleness by $C_{A1} + U_{A1}$.
- $0 < ss_{S1} < C_{A1} + U_{A1}$: Figure 7(b) shows that Q_{S1} will incur no stalness if it was to be executed before Q_{A1}. However, waiting for Q_{A1} to finish execution increases Q_{S1}'s staleness by $C_{A1} + U_{A1} - ss_{S1}$.
- $0 < C_{A1} + U_{A1} < ss_{S1}$: Figure 7(c) shows that Q_{S1} will incur no staleness regardless of its execution order. Hence, waiting for Q_{A1} to finish execution leads to no increase in Q_{S1}'s staleness.

Hence, the increase in total penalty $P_{A \to S}$ due to Q_{A1}'s execution is the right-hand component of Eq. 7.

By the discussion, if $P_{S \to A} < P_{A \to S}$, i.e., satisfies Eq. 7, Q_{S1} is executed first.

3.3 Implementation

Putting it together, Algorithm 1 outlines the general framework of AFIT. In particular, the AFIT query router places each newly arriving query into either AL or SL (line 2). Further, at each scheduling point, AFIT would maintain both AL and SL where some queries might be shuffled between the two lists by calling the *Adjust*() procedure (line 4). Finally, the AFIT query selector will compare the two queries with the highest priority in AL and SL and the winner is scheduled

Algorithm 1. Framework of AFIT

```
1 while true do
2     foreach new arrival query Q_i do
3         QueryRouter (Q_i) ;            /* the condition in Section 3.1 */
4     Adjust() ;
5     Q_A1 ← head of AL, Q_S1 ← head of SL;
6     Q_e ← QuerySelector (Q_A1, Q_S1) ;  /* the condition in Section 3.2 */
7     execute Q_e;
```

Procedure Adjust

```
1 foreach query in SL do
2     if Eq. 4a ∨ Eq. 4b then
3         move into AL;
4 foreach query in AL do
5     if ¬ Eq. 4a ∧ ¬ Eq. 4b then
6         move into SL;
```

for execution (line 6). In our implementation, AL and SL are maintained as two heaps. The cost of element insertion and deletion in a heap is $\log n$. Hence, the complexity of Algorithm 1 is $O(n \log n)$ which is similar to other classical scheduling policies including SJF, EDF, etc.

To implement AFIT in key-value data stores, the query API is extended by adding two parameters to specify the QoS and QoD requirement. AFIT works at each node to schedule the execution of queries in the local query queues. That is, it replaces existing scheduling methods that are oblivious to QoS and QoD requirements (e.g., FCFS in Cassandra).

4 Experiments

We have created a detailed simulator to evaluate our proposed method. For each simulated point, we generate 5000 queries and report the average results of 5 runs. The details of our simulator are as follows:

Queries: The processing cost C_i of each query Q_i depends on the accessed data item and is generated according to a uniform distribution over the range [10, 20] mSec. Each query Q_i is assigned a tardiness deadline $TD_i = A_i + k_i * C_i$, where k_i is generated uniformly over the range [1, k_{max}] and $k_{max} = 10$ in our experiments. To assign staleness deadlines, each query Q_i is associated with a staleness tolerance δ_i that is uniformly distributed over [0, 100] mSec. Hence, Q_i's staleness deadline $SD_i = R_i + \delta_i$ where R_i is the arrival time of the earliest unapplied update to the data item accessed by Q_i. Further, each query is assigned a weight W_i uniformly distributed over the range [1, 10] which represents the

importance of that query. The QoS fraction of the weight (i.e., λ_i) follows *Zipf* distribution in the range [0, 1] where in the default setting the skewness of *Zipf* distribution is 0 (i.e., uniform). The arrival of queries is modeled as a poisson process, where the arrival rate is a simulation parameter.

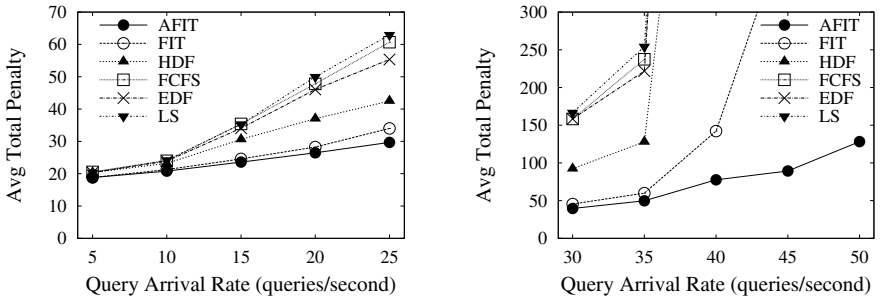
Updates: The processing cost of each update is generated according to a *Zipf* distribution over the range [10, 50] mSec. The skewness of *Zipf* distribution allows us to control the impact of updates on the system load. In the default setting, it skews towards the high-end of the cost range and the skewness is 0.3. The arrival of updates is also modelled as a poisson process, where we set the arrival rate to 20 updates/sec.

Baselines: In this work, we follow the popular design principle on the separation of *mechanism* and *policy*. In particular, we make the distinction between the high-level general mechanism used for specifying the dependency between queries and updates and the low-level scheduling policy used for ordering the execution of those queries and updates. As such, we compare AFIT against the *OD* mechanism, under which we have implemented several low-level scheduling policies, namely: First Come First Served (FCFS), Earliest Deadline First (EDF), Least Slack (LS) [2] and Highest Density First (HDF) [4]. For setting the priority under LS, we use the *AL tardiness slack* and the priority is simply computed as $-ts_i^a$. For HDF, the priority is $W_i/(C_i + U_i)$ which considers both the cost of applying the last pending update to the accessed data item (i.e., U_i) and the cost of query execution (i.e., C_i). We also compare against FIT as it has been described in [15].

4.1 Impact of Query Arrival Rate

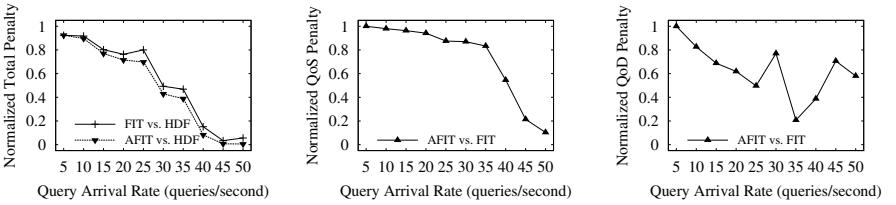
In this experiment, we set all the parameters to the default values aforementioned. The query arrival rate is varied from 5 to 50 queries/second. Figure 8 shows the average total penalty incurred by the different policies. For all policies, the total penalty increases along with increasing the query arrival rate. At low query arrival rate (Figure 8(a)), all the *OD*-based policies (i.e., FCFS, EDF, LS and HDF) perform poorly since they always need to refresh any accessed data item. That poor performance is further emphasized in the case of high query arrival rate (Figure 8(b)). Figure 8(b) also shows that at high query rate, AFIT significantly outperforms the original FIT policy.

To further illustrate the performance of AFIT vs. FIT, in Figure 9(a) we plot their performance normalized to that of HDF (the best OD-based policy as shown in Figure 8). From the figure, it is clear that both FIT and AFIT are superior to HDF. However, AFIT consistently outperforms FIT, especially at medium query loads where a mix of EDF and SJF scheduling is much needed. For instance, at a load of 25 queries/second, AFIT reduces the average total penalty by 30% compared to HDF whereas FIT reduces it by only 20%. Finally, the normalized performance of AFIT vs. FIT in terms of the QoS and the QoD components is shown in Figures 9(b) and 9(c) respectively.



(a) Low Query Rate (b) High Query Rate

Fig. 8. Impact of Query Arrival Rate: Average Penalty



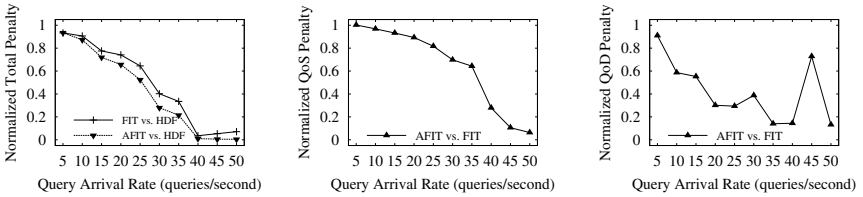
(a) Total Penalty (b) QoS Penalty (c) QoD Penalty

Fig. 9. Impact of Query Arrival Rate: Normalized Penalty

4.2 Impact of Update Cost

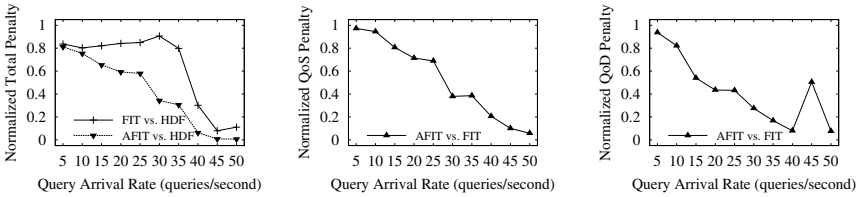
In this experiment, we keep all the default settings except for the update skewness factor, which is set to be 0.8 (i.e., skewed towards high update costs). Figure 10(a) shows the average penalty achieved by AFIT and FIT normalized to that of HDF. The figure shows that the reductions in penalty provided by AFIT and FIT at hight update costs are much higher than those under the default setting (as previously shown in Figure 9(a)). For example, at 35 queries/second, AFIT provides a 53% reduction in penalty compared to HDF under the default setting (Figure 9(a)) and this reduction increases to 66% at high update costs (Figure 10(a)). The same pattern of relative performance applies for FIT. AFIT, however, adapts better than FIT as the update cost increases. For instance, at 35 queries/second, FIT provides only a 36% reduction in penalty compared to HDF (vs. the 66% reduction provided by AFIT).

The results on the QoS and QoD penalties are depicted in Figures 10(b) and 10(c), respectively. Overall, the figures above show that, compared to FIT, AFIT does better in recognizing updates with high cost and it either skips them to reduce the tardiness of their corresponding queries or assign those queries a low priority to enable the early execution of other pending queries.



(a) Total Penalty (b) QoS Penalty (c) QoD Penalty

Fig. 10. Impact of Update Cost: Normalized Penalty



(a) Total Penalty (b) QoS Penalty (c) QoD Penalty

Fig. 11. Impact of the QoS and QoD Preferences: Normalized Penalty

4.3 Impact of the QoS and QoD Preferences

In this experiment, to evaluate the impact of the QoS and QoD preferences, we set the *Zipf* skewness of QoS fraction to 0.7 which leads to a higher preference for QoS. Figure 11(a) shows that the performance gap between AFIT and FIT becomes significantly larger with higher QoS preference, especially at moderate query loads. For instance, at 30 queries/second, FIT only reduces the total penalty by 10% compared to HDF, whereas AFIT reduces it by 61%. In Figures 11(b) and 11(c), we break down the total penalty into its QoS and QoD components to further illustrate the relative performance of AFIT vs. FIT. The figures show AFIT's ability to handle queries with high QoS preferences and selectively skipping their pending updates whenever it is beneficial.

5 Conclusions

In this paper, we propose a novel scheduling method called AFIT for scheduling queries and updates in key-value data stores in order to minimize the combined QoS and QoD penalty. Towards this, AFIT employs a selective policy in applying pending updates to balance the trade-off between QoS and QoD. Additionally, AFIT complements the SJF-based scheduling with an EDF-like policy. Due to the effective query routing and query selection, AFIT outperforms the other existing approaches for a wide spectrum of workload settings, which highlights its adaptivity as demonstrated by our experimental evaluation.

Acknowledgment. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is partially supported by National Science Foundation of China under grant numbers 61003069 and 60925008, National Basic Research (973 Program) under grant number 2010CB731402, and Australian Research Council grant DP110102777.

References

1. Abadi, D.: Consistency tradeoffs in modern distributed database system design: Cap is only part of the story. IEEE Computer 45(2), 37–42 (2012)
2. Abbott, R.K., Garcia-Molina, H.: Scheduling real-time transactions: A performance evaluation. ACM Trans. Database Syst. 17(3), 513–560 (1992)
3. Adelberg, B., Garcia-Molina, H., Kao, B.: Applying update streams in a soft real-time database system. In: SIGMOD Conference, pp. 245–256 (1995)
4. Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Pruhs, K.R.: Online weighted flow time and deadline scheduling. In: Goemans, M.X., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX-RANDOM 2001. LNCS, vol. 2129, pp. 36–47. Springer, Heidelberg (2001)
5. Cooper, B.F., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H.-A., Puz, N., Weaver, D., Yerneni, R.: Pnuts: Yahoo!'s hosted data serving platform. PVLDB 1(2), 1277–1288 (2008)
6. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: amazon's highly available key-value store. In: SOSP, pp. 205–220 (2007)
7. Guirguis, S., Sharaf, M.A., Chrysanthis, P.K., Labrinidis, A., Pruhs, K.: Adaptive scheduling of web transactions. In: ICDE, pp. 357–368 (2009)
8. Labrinidis, A., Roussopoulos, N.: Exploring the tradeoff between performance and data freshness in database-driven web servers. VLDB J. 13(3), 240–255 (2004)
9. Lakshman, A., Malik, P.: Cassandra: structured storage system on a p2p network. In: PODC, p. 5 (2009)
10. Qu, H., Labrinidis, A.: Preference-aware query and update scheduling in web-databases. In: ICDE, pp. 356–365 (2007)
11. Saito, Y., Shapiro, M.: Optimistic replication. ACM Comput. Surv. 37(1), 42–81 (2005)
12. Sharaf, M.A., Chrysanthis, P.K., Labrinidis, A., Amza, C.: Optimizing I/O-intensive transactions in highly interactive applications. In: SIGMOD Conference, pp. 785–798 (2009)
13. Sharaf, M.A., Xu, C., Zhou, X.: Finding the silver lining for data freshness on the cloud: [extended abstract]. In: CloudDB, pp. 49–50 (2012)
14. Wada, H., Fekete, A., Zhao, L., Lee, K., Liu, A.: Data consistency properties and the trade-offs in commercial cloud storage: the consumers' perspective. In: CIDR, pp. 134–143 (2011)
15. Zhu, Y., Sharaf, M.A., Zhou, X.: Scheduling with freshness and performance guarantees for web applications in the cloud. In: ADC (2011)

Near-Optimal Partial Linear Scan for Nearest Neighbor Search in High-Dimensional Space

Jiangtao Cui[1], Zi Huang[2], Bo Wang[1], and Yingfan Liu[1]

[1] School of Computer Science and Technology, Xidian University
Xi'an 710071, China
{cuijt,wangbo_st,yfliu_st}@xidian.edu.cn
[2] School of Information Technology and Electrical Engineering,
The University of Queensland, QLD 4072, Australia
huang@itee.uq.edu.au

Abstract. One-dimensional mapping has been playing an important role for nearest neighbor search in high-dimensional space. Two typical kinds of one-dimensional mapping method, direct projection and distance computation regarding to reference points, are discussed in this paper. An optimal combination of one-dimensional mappings is achieved for the best search performance. Furthermore, we propose a near-optimal partial linear scan algorithm by considering several one-dimensional mapping values. During the linear scan, the partial distance to the query point computed in the 1D space is used as the lower bound to filter the unqualified data points. A new indexing structure based on clustering with Gaussian Mixture is also designed to facilitate the partial linear scan, which can reduce both the I/O cost and distance computations dramatically. Comprehensive experiments are conducted on several real-life datasets with different dimensions. The experimental results show that the proposed indexing structure outperforms the existing well-known high-dimensional indexing methods.

Keywords: indexing methods, nearest neighbor search, one-dimensional mapping.

1 Introduction

k-Nearest Neighbor (k-NN) search in high-dimensional space has many applications such as multimedia retrieval, time-series matching, data mining, and the like. One serious problem in achieving efficient k-NN search is the notorious "curse of dimensionality". The traditional hierarchical indexing structures always degenerate to visiting the entire dataset when the dimensionality is high, and are eventually outperformed by linear scan[5]. Therefore, for k-NN queries in high-dimensional space, linear scan method remains an efficient search strategy for similarity search [4] [5].

The design of indexing algorithm is also governed by hardware constraints. For the disk-based indexing structure, I/O operations often dominate the cost

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 101–115, 2013.

of query processing because of the relatively low transfer speed between memory and disk. The linear scan avoids seeking the specified page, and it is much faster than random access. Furthermore, linear scan methods are a lot easier to integrate within a database engine or query engine. The typical linear scan methods are the VA-file approachs [20] [8] and the omni-sequential. However, these methods usually need to linearly scan the whole dataset, and every point in the dataset incurs expensive distance computations.

Recently, one-dimensional mapping methods have attracted the attention, which are widely used for the exact or approximate k-NN search. Two typical one-dimensional mapping methods in metric space are projection-based techniques and distance-based techniques. For the approximate NN search, both the Locality Sensitive Hashing (LSH) [10] and NV-tree [14] are based on the concept of projecting data points onto a line and classifying locations along this line with different symbols. The random projection is applied in LSH methods, and Principal Component Analysis (PCA) is used in NV-tree. For the exact NN search, one-dimensional mapping is a good choice for the filter-and-refine strategy, because the 1D mapping values have the lower-bounding property. When mapping the high-dimensional data to 1D space, only partial data points need to be accessed during the query. The omni-sequential [9] and iDistance [12] are two typical methods using distance-based mapping.

When sorting the 1D mapping values, a simple linear scan algorithm using filter-and-refine model can be presented. During the search, we locate the first accessed point with its 1D value nearest to the mapping value of the query, and then perform a two-stage linear scan. Since the 1D mapping value has the lower-bounding property, only partial data file need to be linearly scanned. We call this search strategy as "Partial Linear Scan (PLS)". The PLS has the advantage of linear scan, and it also avoids accessing the whole data file.

In this paper, we aim at finding a best mapping scheme to support PLS. Two kinds of mapping methods were analyzed and compared in this paper. To the best of our knowledge, the performance of different one-dimensional mappings is firstly studied in this paper, and our observations will be helpful for the researches and applications in the high-dimensional space. To summarize, we make the following main contributions:

- We formalize the PLS algorithm with respect to one-dimensional mapping and discuss its advantages over existing linear scan approaches.
- We identify the parameter that affect the performance of PLS and present a near-optimal PLS to take both processor and I/O time into account. By using the variance of 1D mapping values as the parameter, we find the optimal reference point for distance-based mapping and the optimal projection line for projection-based mapping. Furthermore, an optimal combination of different 1D mapping values is observed.
- We present PLS-VA, an efficient indexing structure for high-dimensional datasets. Clustering with Gaussian Mixture is applied in PLS-VA and experiment results on multimedia datasets show it can facilitate the PLS on the approximate vectors.

The rest of this paper is organized as follows. In Section 2, we survey the main current indexing methods. Some observations of our work and the PLS algorithm are presented in Section 3. Section 4 introduces our PLS-VA indexing structure. An extensive performance study is reported in Section 5, and finally, we conclude our paper in Section 6.

2 Related Work

Research on high-dimensional indexing can be divided into exact and approximate retrieval. In the approximate category, Hashing has been demonstrated to be effective for similarity search. LSH is one typical hashing method [10]. In Euclidean space, the basic idea is to project data points onto a line and classify locations along this line with different symbols. Recently, the distance-based hashing has also been introduced [1]. Several heuristic variants of LSH have also been suggested. For example, Multi-probe LSH can obtain the same search quality with much less tables [15], while LSB-tree addresses both the quality and efficiency of multimedia retrieval [19]. NV-tree is another representative index for approximate NN search [14]. Using partitioning and projections based on PCA, NV-tree can give approximate answers with a single random disk read.

For exact k-NN queries in high-dimensional space, there exist mainly three categories of high-dimensional indexing methods, such as dimensionality reduction, data approximation and one-dimensional mapping. A well known approach to improving the indexing performance is Dimensionality Reduction (DR) before indexing the data in the reduced-dimensionality space [16]. The linear DR approach first condenses most of information in a dataset to a few dimensions by applying PCA or other techniques. Two strategies for dimensionality reduction include Global DR and Local DR [7].

VA-file is the representative high-dimensional index for data approximation, which suggests accelerating the linear scan by the use of data compression and filtering of the feature vectors [20]. Some extensions of VA-file have been proposed, such as IQ-tree [3], which achieves better query performance by combining a tree structure with VA-file. VA$^+$-file improves the approximate ability of VA-file by transforming the data points in PCA space [8]. The Vector Approximation can be seen as the scalar quantization. The Hyperplane Bound (HB) method uses vector quantization to compress data points, and a new filtering algorithm based on bounding hyperplane has been presented [17].

One-dimensional mapping approaches provide another direction for high-dimensional indexing. The 1D mapping value has the lower-bounding property. Therefore, data points can be cut off based on the 1D values, and the real nearest neighbors are verified in the set of candidates. The typical example is iDistance [12]. The dataset is partitioned and a reference point of each partition is defined. Then data points are mapped to 1D values based on their distance to the reference point. However, its performance is sensitive to the selection of reference points and too much random access of data pages is required. Omni-sequential method chooses some reference points as global 'foci' and gauges all

other data points based on their distances to each focus [9]. During the search, the distances of the query point to each focus are computed, and the triangular inequality can be used to reject the impossible point.

3 Optimal One-Dimensional Mapping

3.1 One-Dimensional Data Transformation

The basic idea of one-dimensional mapping is transforming high-dimensional points into 1D values, which can be used to compute the lower bound of the distance between points in the original high-dimensional space. Two typical one-dimensional mapping methods in metric spaces are projection-based techniques and distance-based techniques with respect to a chosen reference point.

Definition 1 (Projection-based 1D Transformation). *Given a point p and a vector X in the high dimensional space $\boldsymbol{R}^d$, p can be projected onto X with its one-dimensional mapping value $p \cdot X$.*

Definition 2 (Distance-based 1D Transformation). *Given a point p and a reference point o in the high dimensional space $\boldsymbol{R}^d$, the one-dimensional mapping value of p is defined as the distance between p and o, which is denoted as $dist(p, o)$. For Euclidean distance, $dist(p, o) = ||p - o||$, where $|| \cdot ||$ is L_2 norm.*

Lemma 1. *Given two points p and $q \in \boldsymbol{R}^d$, whose 1D mapping values are denoted as p^1 and q^1 respectively, we have $dist(p, q) \geq |p^1 - q^1|$*

Proof. There are two cases need to be considered here.
(1) For projection-based transformation, $p^1 = p \cdot X$ and $q^1 = q \cdot X$. We have

$$dist(p, q) = ||p - q|| \geq ||pX - qX|| = |p^1 - q^1|$$

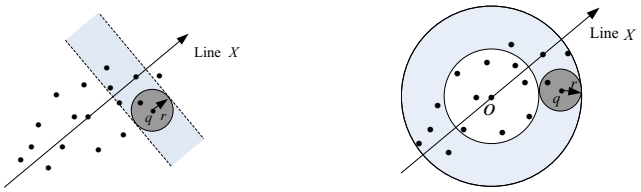
(2) For distance-based transformation, $p^1 = dist(p, o)$ and $q^1 = dist(q, o)$. By considering the triangular inequality, we have

$$dist(p, q) \geq |dist(p, o) - dist(q, o)| = |p^1 - q^1|$$

3.2 Partial Linear Scan

As proved in the Lemma 1, given two points, their full distance in the original high dimensional space cannot be smaller than their 1D transformation distance. Thus, their 1D transformation values (or 1D values, in short) can be used to compute the lower bound of the full distance. According to this property, a synchronous bi-directional linear scan algorithm can be designed to perform an efficient *PLS*.

Given a collection of data points, they are firstly sorted in ascending order according to their 1D values. Given a query point q, a distance array $kDist[\,]$ of size k is employed to store the k-NN distances found so far. During the search,



(a) Projection (b) Distance

Fig. 1. Search Space after 1D transformation

we locate the first point to be accessed as p_b whose 1D value p_b^1 is nearest to q^1. A bidirectional scan will be conducted by accessing data points from p_b (i.e., forward and backward from p_b). In practice, the linear scan in the disk always accesses data pages along one direction. The bidirectional search can be modified as two search processes with the same direction. The detailed algorithm is shown in Algorithm 1 and explained below.

Require: Query point q, dataset D
Ensure: $kDist\,[]$
1: Initialize $kDist\,[]$ with $MAXREAL$
2: Locate the first accessed point p_b w.r.t q^1 in the first stage
3: **for** $i = b$ to N **do**
4: Calculate $\left|p_i^1 - q^1\right|$
5: **if** $\left|p_i^1 - q^1\right| > kDist[k]$ **then**
6: break //End of the first stage scan
7: **else**
8: Calculate $dist\,(q, p_i)$ and update $kDist\,[]$
9: **end if**
10: **end for**
11: Locate the first accessed point p_s w.r.t q^1 and $kDist[k]$ in the second stage
12: **for** $i = s$ to $b - 1$ **do**
13: Calculate $\left|p_i^1 - q^1\right|$
14: **if** $\left|p_i^1 - q^1\right| > kDist[k]$ **then**
15: break //End of the second stage scan
16: **else**
17: Calculate $dist\,(q, p_i)$ and update $kDist\,[]$
18: **end if**
19: **end for**

Algorithm 1. Partial linear scan for exact kNN search

We perform the first stage scan first (Line 3-10). Before computing the full distance between the query q and the current data point p_i, the distance between q^1 and p_i^1 is calculated firstly (Line 4). If $\left|p_i^1 - q^1\right| > kDist\,[k]$, p_i can be safely pruned without computing its full distance $dist\,(q, p_i)$ of all dimensions. Furthermore, the linear scan in this direction can be terminated (Line 5-6). If $\left|p_i^1 - q^1\right| < kDist\,[k]$, the full distance between p_i and q is required to be calculated while $kDist\,[\,]$ will be updated (Line 8). When the scan in the first stage

is terminated, we perform the second stage scan similarly (Line 11-18). The first accessed point p_s at the second stage satisfies $\left|p^1_{s-1} - q^1\right| > kDist[k]$ and $\left|p^1_s - q^1\right| \leq kDist[k]$. Fig. 1 shows data points accessed during the PLS when performing projection-based and distance-based 1D transformation respectively, where the k-th NN distance is denoted as r.

3.3 Performance Analysis on One-Dimensional Mappings

In this subsection, we will discuss how to evaluate the performance of different 1D transformation methods and introduce several important observations. The filtering efficiency is a critical indicator of the PLS performance based on a certain 1D data transformation. It is formally defined as below.

Definition 3 (Filtering Efficiency, FE). *Given a number of N data points and their 1D transformation, N_f is the number of data points pruned in PLS. The filtering efficiency (fe) is defined as: $\boldsymbol{fe} = N_f/N$, where $N = |D|$.*

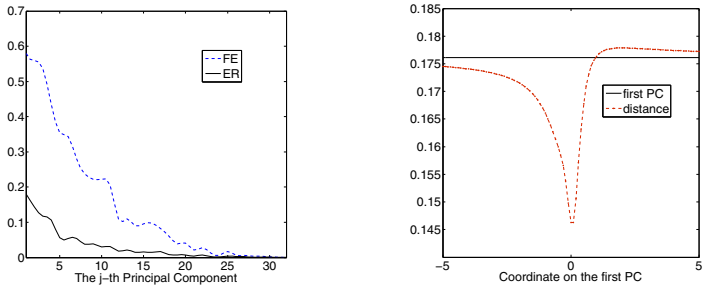
The best and most widely used approach for data projection is based on PCA. For any two points p and q in a dataset $D \in \mathbf{R}^d$, they can be projected onto the j-th PC of D, denoted as p_j and q_j respectively. We give another difinition according to the PCA.

Definition 4 (Energy Ratio, ER). *Let σ_j^2 denotes the variance along the j-th PC. The energy ratio of the j-th PC, denoted as $er\,(j)$, is defined as follows: $er\,(j) = \sigma_j^2 / \sum_{k=1}^{d} \sigma_k^2$*

Apparently, a larger fe implies a smaller number of data points to be accessed during the PLS. The $er(j)$ measures the percentage of the variance introduced by j-th PC over the whole distance variance. It's difficult to analyze the **FE** theoretically, since datasets have different distributions and different query points have different query performance. However, several important observations have been found according to the experiment results. The first group of experiments are conducted on a widely used real-life dataset, COLHIST[1], which contains a number of 68,040 32-dimensional color histograms. The conclusions and observations found on the COLHIST dataset are also validated by the experiments on other datasets such as, the LANDSAT dataset and the SIFT dataset. To get fairly results, 500 10-NN searches are performed to get the average performance. We first test the **FE** and **ER** along different PCs, which are shown in Fig. 2(a). We can observe that two curves have similar tendency and a larger **ER** leads to a higher **FE**.

Observation 1. *For projection-based 1D data transformation, the ER can reflect the FE of the PCs. In result, the first PC is the optimal projection line for 1D data transformation.*

[1] `http://kdd.ics.uci.edu/database/CorelFeatures`



(a) FE and ER (b) Variances comparison

Fig. 2. Performance Analysis

The Observation 1 can be easily justified. Data variance measures how data points are spread out along the corresponding direction. As shown in Fig. 1(a), the shadow area is the search space of the query q. The larger the data variance is along X, the fewer the data points fall into the shadow area, leading to a higher **FE**.

For distance-based 1D transformation, the **FE** mainly depends on the choice of reference point. We have shown the strong correlation between the data variance and the filtering ability. An optimal reference point should be the point maximizing the variance of the distances from the data points to the reference point. It is impossible to test the entire space to find the optimal reference point. Fortunately, PCA can be used to find the direction with largest data variance for a dataset. The optimal reference point most likely lie on the line identified by the first PC [18]. We select points along the first PC as the reference points, and test the corresponding data variances. Fig. 2(b) shows the variances of the distances w.r.t a reference point when selecting the reference point along the first PC. The tests on the the other PCs also show the similar tendency.

Observation 2. *The largest variance of distances w.r.t the reference point lies out of one side of the line determined by the first PC. When the reference point lies very faraway the origin along the first PC, the limit of variance of distances is equal to the variance of the first PC.*

3.4 Combination of Several One-Dimensional Mappings

In this subsection, we will focus on 1D data transformation by utilizing both project-based and distance-based mapping to achieve a better filtering efficiency. By taking more information into account, the data points can be better distinguished. Two models are studied in our work. In the first model, we consider both 1D values computed by data projection and the distance to a reference point. In the second one, we select two reference points to get distances. The examples of these two advanced 1D transformation models are shown in Fig. 3, in which only data points labeled with asterisk need to be accessed. We firstly

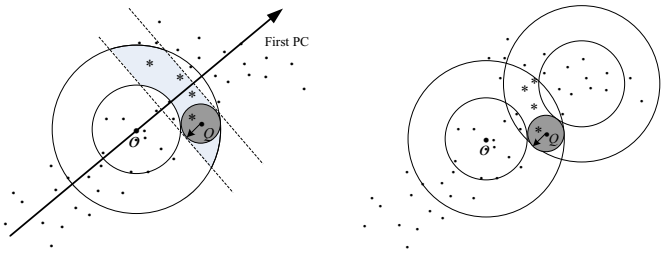


Fig. 3. Combination of 1D transformation models

test the combined **FE** of projection and distance by selecting reference point along the first PC. Fig. 4(a) shows the results.

Observation 3. *When combining 1D values achieved by projection-based and distance-based 1D transformations, the largest **FE** can be reached when the reference point lies on the origin and the projection line is along the first PC.*

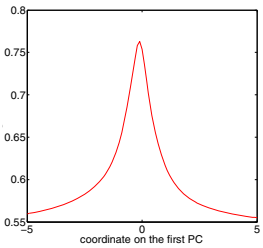
The distance of data points to the origin is the norm. When the reference point lies far outside of origin, the accessed region using projection and distances will overlap, and result in almost the same **FE** as that just using projection.

Now, we will discuss how to get the largest **FE** when selecting two reference points. By fixing one reference point on the first PC, we test the combined **FE** when selecting the other reference point along the first PC. Three typical points are chosen as the first reference point to illustrate different curve tendency, whose 1D value on the first PC are 0, 0.5 and 3 respectively. Fig. 4(b) shows the **FE** when one reference point is fixed.

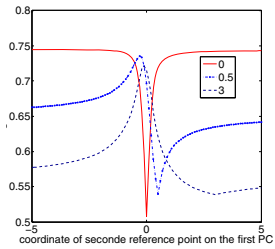
When the first reference point lies in the origin (i.e., 0), the largest **FE** can be achieved if the second reference point locates very far outside of the origin. As mentioned above, when the reference point lies very far outside of origin, the accessed region using distances and projection will fully overlap. Therefore, we can get the last observation.

Observation 4. *When 1D transformation is based on the distance w.r.t two reference points, its highest **FE** cannot be greater than the highest **FE** achieved by utilizing both projection-based and distance-based 1D transformations.*

So far, we can design a near-optimal PLS algorithm to take both processor and I/O time into accout. Since combination of projection on the first PC and norm can get better **FE**, and the variance of first PC is larger than the norm as shown in Fig. 2(b), we sort data points according to the projections on the first PC, and the norms of data points are also embedded. This algorithm is not optimal if just considering I/O cost, since the largest variance of distances is not applied. When performing k-NN search, the projection values on the first PC are used for terminating the searching process. And the norms are used to reject more impossible candidate points. During the high-dimensional distance computation, the Partial Distortion Search (*PDS*) algorithm can be adopted which use the



(a) Combining projection and distance



(b) Combining two distances

Fig. 4. Filtering efficiency

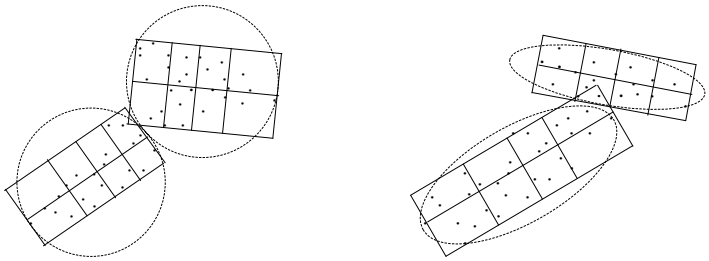
projections on the other PCs to reject points. The *PDS* algorithm was widely used in the Vector Quantization encoding process [2]. Since most of energy is condensed to the first several PCs, we can perform *PDS* algorithm only on the first several PCs. If data points can not be rejected on these dimensions, we directly calculate $dist\,(Q, P_i)$ on all dimensions.

4 PLS-VA

We have discussed how to get the near-optimal query performance when performing *PLS*. In this section, we aim to build a new indexing structure applying the PLS. One of the state of the art indices for linear scan is the VA-file approach. Our new indexing structure is proposed by utilizing VA-file and PLS, which is so called PLS-VA.

4.1 The Framework of PLS-VA

The proposed indexing structure is specially designed for real-life datasets, in which PCA is employed to get the projection values on the first PC. To facilitate the PLS, we aim at getting the better **FE** by using partition technique. The similar idea has been presented in iDistance, in which all data points in different partitions are represented in a single dimensional space. It is also well known that LDR can outperform GDR since it can find the correlations in the intrinsic clusters. Several clustering techniques have been investigated, such as K-means clustering and clustering with Gaussian Mixture [6]. We find that the clustering with Gaussian Mixture is more suitable for PLS than K-means clustering. Fig. 5 shows an example. Gaussian Mixture is a model-based clustering approach, which consists in using certain models for clusters and attempting to optimize the fit between the data and the model. The Gaussian mixture architecture estimates probalility density fuctions for each class, and then performs classification based on Bayesian rule.



(a) K-means partition (b) GMM partition

Fig. 5. Different partitions for building approximate vectors

Obviously, building approximate vectors in several partitions separately can also improve the approximate ability, and the additional cost is that several codebooks need to be maintained. After performing the clustering, PCA is applied to each cluster. The approximate vectors belonging to the same cluster are firstly sorted according to the 1D projection values and then loaded in contiguous data pages. The bound of projection values of each page can be described using a two-dimensional array $[\alpha, \beta]$. A B^+-tree is applied to manage all the arrays in one cluster, where only two values in one page are indexed as the keys in the B^+-tree.

4.2 Searching in PLS-VA

When performing k-NN search in the PLS-VA, the accessing order of each cluster is determined by computing the lower bounds between the query point and each cluster. Each cluster can be denoted as a hyper rectangle or a hyper sphere. The hyper rectangle can be seen as an MBR (minimum bounding rectangle) used in the R-tree [11], where the lower bound of distance between q to the MBR can be easily calculated. The lower bound between q and the sphere can also be computed, and the maximum value between two lower bounds is chosen as the real lower bound between q and the cluster.

For the k-NN query in the whole dataset, we use a priority queue to navigate the clusters accessed in increasing order of their lower bounds to the query point. If the lower bound of a cluster to the query point is zero, the distance between the centroid and q can be compared. During the search, if the lower bound of one cluster to q is larger than the k-th smallest distance found so far, all the data points in this cluster do not need to be accessed and the search can be terminated. The k-NN search in PLS-VA includes two phases. The first phase is to linearly scan the partial approximation file, and the second phase is the access of exact vectors. The first search phase can guarantee no false dismissals for the query. However, the search results may contain false positives which have to be further refined in the second phase. In the first stage, it is noted that we need to compute lower and upper bounds of distance instead of exact distances. Therefore, the **FE** of PLS-VA is lower than the PLS in the exact vectors.

5 Experiments

In this section, we performed extensive experiments on the high-dimensional real-life datasets to demonstrate the practical effectiveness of PLS-VA. All experiments were executed on HP workstation xw9300 with AMD Opteron 2.2 GHz CPU, 2GB RAM and 73GB 10Krpm SCSI disk. The data page size used in the experiments is 4096 Bytes. We use real-life datasets to evaluate the effectiveness of our method. Two datasets which are widely used in high-dimensional indexing were chosen. The first dataset is **COLHIST**[2], which contains 68,040 32-dimensional color vectors. The other dataset is called Satellite Image Texture (**LANDSAT**)[3], which contains 275,245 60-dimensional feature vectors extracted from satellite images.

We use 500 queries to obtain the average results on 10-NN, 20-NN and 50-NN search. A comprehensive performance study has been conducted on VA, iDistance, HB, and PLS-VA. The results show the superiority of PLS-VA.

5.1 I/O Cost Model

The search in PLS-VA is comprised two phases. The first phase is to linearly scan the approximate file, and the second phase is the random access of exact vectors. We use the random page access as the I/O metric. The total I/O cost of PLS can be calculated as: $IO_{total} = IO_l + IO_r$. Suppose there are M clusters, we need compute the sum of linear I/O cost of M clusters. Accessing of a new cluster also need additional 2 random page access. The total I/O cost of PLS-VA can be calculated as: $IO_{total} = IO_l + IO_r + 2 \times M$. Assuming linear I/O is 10 times faster than random I/O [7] [13], and the post processing of candidate can be one I/O per exact vectors [20]. However, as observed in [7], this assumption is overly pessimistic. We, therefore, can assume that the total I/O cost of PLS-VA is: $IO_{total} = num\_l/10 + num\_r/2 + 2 \times M$. Where num_l is the total number of page accessed in the first phase, and num_r is the number of exact vector accessed during the second searching process.

5.2 Performance Study on the Filtering Efficiency

In the first experiment, we study the effect of different clustering methods. The results of 10-NN queries of PLS algorithm are shown in Fig. 6. We can see that the clutering with GMM can improve the **FE** compared with the K-means clustering. As expected, as the number of cluster increases, PLS incurs larger **FE**. While we can choose a large number of clusters to improve the **FE**, this will increase the random I/O cost. So a moderate number of clusters is fine. In our other experiments, we have used 4 as the default number of clusters.

The second experiment studies the **FE** of different methods. We compare PLS with two other well known indexing methods using clustering technique,

[2] `http://kdd.ics.uci.edu/database/CorelFeatures`
[3] `http://vision.ece.ucsb.edu/datasets/`

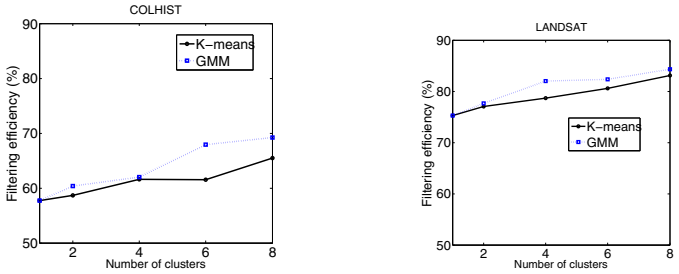


Fig. 6. Comparison of filtering efficiency, the result of 10-NN queries

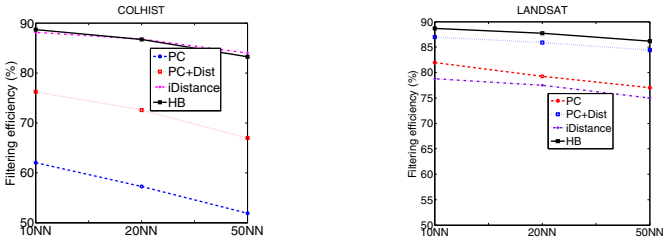


Fig. 7. Comparison of filtering efficiency

iDistance and HB, which will be discussed later in detail. Both iDistance and HB use the filter-and-refine strategy, and need to select a good number of clusters. We use 64 partitions for iDistance in two datasets. We also tried the HB for several numbers of clusters, then use 80 clusters for the COLHIST and 200 clusters for the LANDSAT. We also studied the effect of combination of two one-dimensional mappings. The results are shown in Fig. 7. The "PC" denotes the **FE** of the first PC, and "PC+Dist" deontes the **FE** when combining the first PC and the norm. The combination of several 1D mapping values can improve the **FE**. From the figures, we can see very small difference between "PC+Dist" and HB, and HB performs the best for the two datasets.

5.3 Comparative Study of PLS-VA with Other Methods

In this section, we compare PLS-VA with VA-file, iDistance and HB. For VA and PLS-VA, the average approximate bit length is 6 bits per dimension for the two datasets. When using PLS algorithm in PLS-VA, the distortion distance on the first 5 PCs are used to prune data points. iDistance need to select a good number of reference points to work efficiently. In our comparative studies, we choose 64 reference points, which was reported to have the best average performance. In COLHIST, the first search radius and the increasing step are 0.01, and in LANDSAT the same parameters are set 0.1. HB uses the separating hyperplane boundaries of Voronoi clusters to complement the clustering-based index. However, the HB suffers a huge number of lower bounds computations, suppose

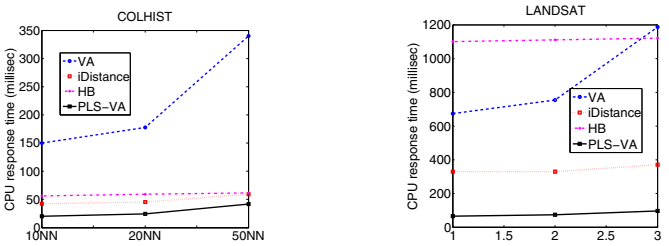


Fig. 8. Comparison of CPU cost

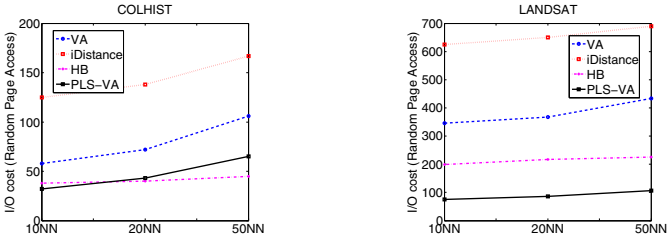


Fig. 9. Comparison of I/O cost

there are K clusters, and as many as $K(K-1)/2$ lower bounds calculations are needed during the query. As mentioned above, 80 clusters are applied in COLHIST and 200 clusters in LANDSAT.

First we present the comparision of CPU response time of different methods. The results are shown in Fig. 8. The **FE** of HB and iDistance is larger than PLS-VA, but PLS-VA has the shortest response time. It has a speedup factor of more than 2 over iDistance and 6 over VA. The CPU response time of HB is relative to the number of clusters. HB does not yet provide effective pruning mechanisms of data points in the candidate cluster. To achieve better **FE**, more clusters are needed. It can be seen its CPU time is more than 1 second in the LANDSAT. PLS-VA is better than iDistance because the combination of several 1D mapping values enables more effective pruning of data points.

We also compared the I/O cost between different methods. The results are shown in Fig. 9. Both PLS-VA and HB have less random page accesses than VA and iDistance. The iDistance has the most random page accesses, because iDistance incrementally enlarges the search space to find NNs in different partitions, and more partitions result in more fragmented pages and lead to more random page accesses. HB linearly scan the candidate clusters to find NNs, therefore it has less I/O cost. PLS-VA and HB have similar I/O cost in COLHIST, and PLS-VA has the least I/O cost in LANDSAT. To summarize, PLS-VA outperforms VA and iDistance in both CPU and I/O cost, and PLS-VA has less total response time than HB.

6 Conclusions

In this paper, we have studied the performance of different 1D transformation methods and derived a novel yet more effective model to transform high dimensional data points into 1D subspace, in which an efficient PLS can be performed. A novel indexing structure PLS-VA is proposed which partitions the original data space with GMM and builds approximate vectors on each cluster separately. The proposed indexing structure can be easily built, and it highly improves the pruning power of the linear scan on the vector approximate file. Extensive experiments are conducted on two real-life multimedia datasets. The results confirm that PLS-VA outperforms existing indexing structures.

References

1. Athitsos, V., Potamias, M., Papapetrou, P., Kollios, G.: Nearest neighbor retrieval using distance-based hashing. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, pp. 327–336. IEEE Computer Society, Washington, DC (2008)
2. Bei, C.-D., Gray, R.M.: An improvement of the minimum distortion encoding algorithm for vector quantization. IEEE Trans. Communication 33(10), 1132–1133 (1985)
3. Berchtold, S., Bohm, C., Jagadish, H.V., Kriegel, H.-P., Sander, J.: Independent quantization: An index compression technique for high-dimensional data spaces. In: ICDE 2000: Proceedings of the 16th International Conference on Data Engineering, p. 577. IEEE Computer Society, Washington, DC (2000)
4. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1998)
5. Böhm, C., Berchtold, S., Keim, D.A.: Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Comput. Surv. 33(3), 322–373 (2001)
6. Celeux, G., Govaert, G.: A classification em algorithm for clustering and two stochastic versions. Comput. Stat. Data Anal. 14(3), 315–332 (1992)
7. Chakrabarti, K., Mehrotra, S.: Local dimensionality reduction: A new approach to indexing high dimensional spaces. In: VLDB 2000: Proceedings of the 26th International Conference on Very Large Data Bases, pp. 89–100. Morgan Kaufmann Publishers Inc., San Francisco (2000)
8. Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., Abbadi, A.E.: High-dimensional nearest neighbor searching. Information Systems 31(6), 512–540 (2006)
9. Filho, R.F.S., Traina, A.J.M., Traina Jr., C., Faloutsos, C.: Similarity search without tears: The omni family of all-purpose access methods. In: Proceedings of the 17th International Conference on Data Engineering, pp. 623–630. IEEE Computer Society, Washington, DC (2001)
10. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proceedings of the 25th International Conference on Very Large Data Bases, VLDB 1999, pp. 518–529. Morgan Kaufmann Publishers Inc., San Francisco (1999)
11. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: SIGMOD 1984: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, pp. 47–57. ACM, New York (1984)

12. Jagadish, H.V., Ooi, B.C., Tan, K.-L., Yu, C., Zhang, R.: iDistance: An adaptive b+-tree based indexing method for nearest neighbor search. ACM Trans. Database Syst. 30(2), 364–397 (2005)
13. Koudas, N., Ooi, B.C., Shen, H.T., Tung, A.K.H.: Ldc: Enabling search by partial distance in a hyper-dimensional space. In: ICDE 2004: Proceedings of the 20th International Conference on Data Engineering, p. 6. IEEE Computer Society, Washington, DC (2004)
14. Lejsek, H., Ásmundsson, F.H., Jónsson, B.P., Amsaleg, L.: Nv-tree: An efficient disk-based index for approximate search in very large high-dimensional collections. IEEE Trans. Pattern Anal. Mach. Intell. 31, 869–883 (2009)
15. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Multi-probe lsh: efficient indexing for high-dimensional similarity search. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007, pp. 950–961. VLDB Endowment (2007)
16. Postma, E.: Dimensionality reduction: A comparative review 10(February), 35 (October 2009)
17. Ramaswamy, S., Rose, K.: Adaptive cluster distance bounding for high-dimensional indexing. IEEE Trans. on Knowl. and Data Eng. 23(6), 815–830 (2011)
18. Shen, H.T., Ooi, B.C., Huang, Z., Zhou, X.: Towards effective indexing for very large video sequence database. In: SIGMOD 2005: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 730–741. ACM, New York (2005)
19. Tao, Y., Yi, K., Sheng, C., Kalnis, P.: Quality and efficiency in high dimensional nearest neighbor search. In: Proceedings of the 35th SIGMOD International Conference on Management of Data, SIGMOD 2009, pp. 563–576. ACM, New York (2009)
20. Weber, R., Schek, H.-J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB 1998: Proceedings of the 24rd International Conference on Very Large Data Bases, pp. 194–205. Morgan Kaufmann Publishers Inc., San Francisco (1998)

AVR-Tree: Speeding Up the NN and ANN Queries on Location Data

Qianlu Lin, Ying Zhang, Wenjie Zhang, and Xuemin Lin

The University of New South Wales Australia
{qlin,yingz,zhangw,lxue}@cse.unsw.edu.au

Abstract. In the paper, we study the problems of nearest neighbor queries (NN) and all nearest neighbor queries (ANN) on location data, which have a wide range of applications such as Geographic Information System (GIS) and Location based Service (LBS). We propose a new structure, termed AVR-Tree, based on the R-tree and Voronoi diagram techniques. Compared with the existing indexing techniques used for NN and ANN queries on location data, AVR-Tree can achieve a better trade-off between the pruning effectiveness and the index size for NN and ANN queries. We also conduct a comprehensive performance evaluation for the proposed techniques based on both real and synthetic data, which shows that AVR-Tree based NN and ANN algorithms achieve better performance compared with their best competitors in terms of both CPU and I/O costs.

1 Introduction

With the emergence of location-aware mobile device technologies, communication technologies and GPS systems, the location-aware queries have attracted great attentions in many important applications such as location based service (LBS) and geographic information system (GIS). Particularly, the nearest neighbor search and its variants play an important role among these queries since it is very natural to find various facilities with close spatial proximity. In the paper, we focus on the nearest neighbor (NN) and all nearest neighbor (ANN) queries on location data. Given an object (e.g., a user location) and a set of facilities (e.g., warehouses, petrol stations and restaurants), the nearest neighbor search identifies the closest facility regarding an object. All nearest neighbor (ANN) query aims to retrieve nearest neighbors for a set of objects regarding a set of facilities. Both queries are fundamental in spatial queries and have a wide spectrum of applications. Below is a motivating example.

Fig. 1 shows a map consisting of seven supermarkets and three warehouses. To deliver stocks from a warehouse to a supermarket with the minimized delivery cost and time, it is desirable to put the stocks in the nearest warehouse of each supermarket. For simplicity, the distance between two spatial locations is calculated as Euclidean distance. By issuing a nearest neighbor query on supermarket s_1, it is reported that w_3 is its nearest warehouse. On the other hand,

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 116–130, 2013.

we may issue an all nearest neighbor query against all supermarkets and warehouses. Then we can find w_1 is the closest warehouse to supermarket s_2, w_2 is the closest to s_4, s_6 and w_3 is the closest to s_1, s_3, s_5 and s_7.

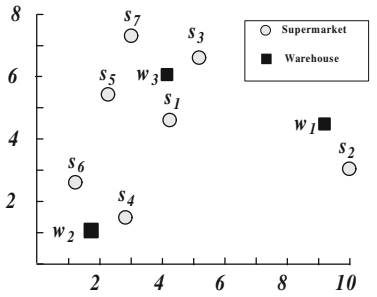


Fig. 1. Motivation Example

Since the number of objects and facilities might be massive in many applications, it is desirable to develop an indexing technique to facilitate the nearest neighbor query and its variants. R-tree technique has been widely employed to support various nearest neighbor queries which relies on the simple rectangular grouping principle, where objects or facilities are grouped by minimal bounding rectangles (MBRs) in a hierarchical way. Efficient NN and ANN algorithms [12,6] have been proposed based on R-tree structure, where the algorithms utilize various distance metrics (e.g., minimal distance and minimal maximal distance) against MBRs. Observe that the R-tree technique may lead to some unnecessary traversals in nearest neighbor query because MBR only provides a coarse granule for various NN queries, Sharifzadeh *et al.* [13] propose the VOR-Tree which incorporates voronoi diagrams into R-tree such that users can quickly identify the search regions with R-tree technique and then provide effective pruning for various NN queries based on the voronoi diagram techniques.

Although VOR-Tree can take advantage of voronoi diagram technique, the benefit is offset by the following facts. Firstly, the intermediate entries of VOR-Tree cannot take advantage of the voronoi diagram technique since they only keep the MBRs of the facilities. This implies that we cannot derive more effective pruning metric compared with the MBR based pruning techniques in [6]. Secondly, since all the vertexes of the voronoi cell and delaunay neighbors are kept for each facility, the capacity (i.e., fan-out) of the VOR-Tree is small. As shown in the empirical study, this results in high I/O cost in NN and ANN queries.

To address the above issues, in the paper we propose a new index approach, termed AVR-Tree, to achieve a better trade-off between the pruning effectiveness and the space usage of the index structure when voronoi diagram technique is applied. More specifically, our index structure is also built on R-tree. But instead of keeping the voronoi cell vertexes and delaunay neighbor information, we maintain the minimal bounding box of a voronoi cell (VBR) and the maximal

nearest neighbor distance[1] (MND) for each facility. Moreover, we also maintain the aggregate information of VBRs and MNDs in the intermediate entries. Then effective pruning techniques are proposed for NN and ANN queries based on AVR-Tree structure.

Contributions. Our contributions can be summarized as follows.

- We propose a new data structure, termed AVR-Tree, that effectively utilizes the advantage of the voronoi diagram and the R-tree techniques.
- Efficient nearest neighbor (NN) and all nearest neighbor (ANN) queries are proposed based on AVR-Tree.
- Comprehensive experiments demonstrate the effectiveness and efficiency of our techniques.

Organization of the Paper. The remainder of the paper is organized as follows. Section 2 formally defines the NN and ANN query. In Section 3, we propose our new data structure, AVR-Tree. Efficient NN and ANN algorithms are proposed based on AVR-Tree in Section 4 and Section 5 respectively. Results of comprehensive performance studies are presented in Section 6. Section 7 introduces the related work. Finally, Section 8 concludes the paper.

2 Preliminary

In this section, we will formally define the problems we study in this paper in Section 2.1. In Section 2.2 we briefly describe the voronoi diagram. Table 1 below summarizes the notations frequently used throughout the paper.

Table 1. The summary of notations

Notation	Definition
o ($\mathcal{O}$)	object (a set of objects)
f ($\mathcal{F}$)	facility (a set of facilities)
$NN(o)$	the nearest neighbor of an object o in facilities $\mathcal{F}$
VBR	the voronoi minimal bounding rectangle for facility entries (data entries and intermediate entries)
MND	the maximal nearest neighbor distance for data entries and intermediate entries

2.1 Problem Definition

A point (location) p referred throughout the paper, by default, is in d-dimensional numerical space. Let $\delta(p, f)$ denote the Euclidean distance between two points (locations) p and f. For simplicity, we use Euclidean distance in the paper. Nevertheless, the techniques proposed in the paper can be easily extended to other metric distances.

[1] The maximal distance if the facility is the nearest neighbor.

The NN query is formally defined as follows.

Definition 1. ***Nearest Neighbor(NN) Query.*** *Given an object o and a set $\mathcal{F}$ of facilities, the nearest neighbor of o, denoted by $NN(o)$, is the facility $f \in \mathcal{F}$ with the closest distance to o; that is, $\delta(o, f) \leq \delta(o, f')$ for any facility f' in $\mathcal{F}$.*

Then we have the definition of all nearest neighbor query in which we aim to find the nearest neighbors for a set of objects.

Definition 2. ***All Nearest Neighbor (ANN) Query*** *Given a set $\mathcal{O}$ of objects and a set $\mathcal{F}$ of facilities, for each object $o \in \mathcal{O}$, we retrieve its nearest neighbor from $\mathcal{F}$.*

Example 1. In Fig. 2, there are three objects $\{o_1, o_2, o_3\}$ and three facilities $\{f_1, f_2, f_3\}$ in $\mathcal{O}$ and $\mathcal{F}$ respectively. Given the object o_1, its nearest neighbor regarding $\mathcal{F}$ is f_2, i.e., $NN(o_1) = f_2$. Fig. 2 shows the nearest neighbors for all objects, which can be obtained by issuing the all nearest neighbor query against $\mathcal{O}$ and $\mathcal{F}$.

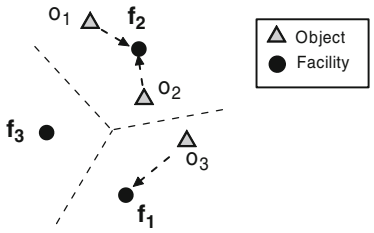


Fig. 2. NN and ANN Example

2.2 Voronoi Diagram

Voronoi diagram is a data structure that consists of a set of voronoi cells where each cell is a polygon. Each cell has an owner which is usually called as seed. Voronoi diagram provides us a nice feature for nearest neighbor search, which is if a point is in a voronoi cell, this point is guaranteed to be the nearest neighbor of the cell's seed. Although the cells are constructed in a way that guarantees the seed is the nearest neighbor of all the points fall in the cell, determining which cell a point falls in can be time consuming.

3 AVR-Tree

In this section, we will introduce a new data structure, termed AVR-Tree, to efficiently facilitate the NN and ANN queries. In Section 3.1, we will show the motivation of the AVR-Tree and then followed by the detailed description of the new indexing structure, including the maintenance of the AVR-Tree.

3.1 Motivation

As the facilities are usually organized by some hierarchical spatial indexes such as R-tree and the NN search follows the *branch-and-bound* travel paradigm, it is critical to estimate the minimal and maximal possible distances between a query object o and a minimal bounding rectangle (MBR) such that some non-promising entries can be excluded from computation at high level to significantly reduce the I/O and CPU costs.

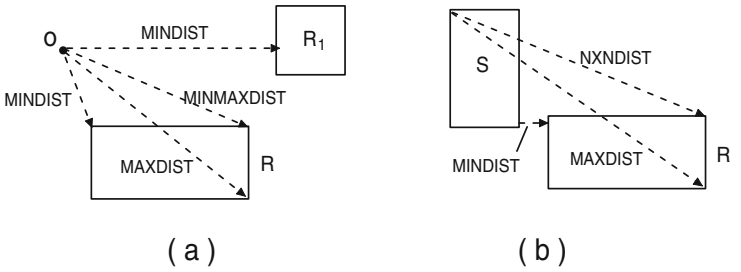


Fig. 3. Example for upper and lower bounds for NN distance

For a query object o, we use nearest neighbor distance, denoted by $nnd(o)$, to represent the distance between o and its nearest neighbor. Given a minimal bounding rectangle (MBR) R of a set of facilities, we use $nnd_{min}(o, R)$ ($nnd_{max}(o, R)$) to denote the lower (upper) bound of the nearest neighbor distance of o regarding facilities located in R. As shown in Fig. 3(a), an immediate approach is to use the minimal distance (MINDIST) and maximal distance (MAXDIST) between o and R as $nnd_{min}(o, R)$ and $nnd_{max}(o, R)$. See detailed calculation in [12]. Observe that each MBR's face is touching at least one facility within R, the concept of MiniMax Distance (MINMAXDIST) is proposed in [12] which can provide a tighter upper bound for $nnd_{max}(o, R)$ as shown in Fig. 3(a).

Example 2. In the example of Fig. 3(a), MINMAXDIST between o and R is smaller than MINDIST between o and R_1. This implies that none of the facilities in R_1 can be the nearest neighbor of o and hence R_1 is pruned without exploring its facilities. On the other hand, we cannot prune R_1 if MAXDIST is used as $nnd_{max}(o, R)$.

With similar rationale, a new pruning metric NXNDIST is proposed in [6], which can provide a tighter upper bound of the nearest neighbor distance between two MBRs. More specifically, as shown in Fig. 3(b), S and R represent MBRs of a set of objects and facilities respectively. For any object o within S, its nearest neighbor distance regarding the facilities within R is bounded by NXNDIST. The empirical study in [6] shows that NXNDIST is much more effective than MAXDIST between two MBRs.

Observe that the MBRs based pruning technique is less effective compared with the Voronoi technique, in [13], VOR-Tree combines the R-tree and Voronoi diagram techniques to speed up the nearest neighbor search. Nevertheless, since all the vertexes of the voronoi cell and delaunay neighbors are kept for each facility, the node capacity (i.e., fan-out) of VOR-Tree is small. For instance, the node capacity of VOR-Tree and R-tree is 40 and 204 respectively when page size is 4K. As shown in the empirical study, this results in expensive I/O costs in NN and ANN queries. Moreover, the intermediate entries of VOR-Tree cannot take advantage of the Voronoi diagram technique since they only keep the MBR of the facilities. This implies that they cannot derive more effective pruning metric compared with the MBR based pruning techniques.

Motivated by the above facts, in the paper we propose a new index approach, termed AVR-Tree, to achieve a better trade-off between the pruning capability and the overhead of the index. Before going to the detail, we first introduce two notations:

Definition 3. ***Voronoi Bounding Rectangle (VBR).*** *Given a facility f, the voronoi bounding rectangle (VBR) of a facility is the minimum bounding box of its voronoi cell, denoted by $f.VBR$. Given a set of facilities F, the VBR of F is the minimal bounding rectangle of the VBRs of all facilities in F, denoted by $F.VBR$.*

Besides the VBR, we also maintain the maximum nearest neighbor distance (MND) for each facility, which is defined as follow.

Definition 4. ***Maximum Nearest Neighbor Distance (MND).*** *Given a facility f, the Maximum Nearest Neighbor Distance (MND) of f , denoted by $f.MND$, is the furthest distance between f and all locations within its voronoi cell; that is, we have $\delta(o, f) \leq f.MND$ if f is the nearest neighbor of an object o. Similarly, we use $F.MND$ to denote the maximal MND values among the facilities in F.*

Since each voronoi cell is a convex polygon, it is immediate that we can derive $f.MND$ by finding the furthest voronoi cell vertex regarding the facility f.

Example 3. In Fig. 4(a), the polygon (v_1, v_2, v_3, v_4, v_5) is the voronoi cell of facility p_1. The VBR of f_1 ($f_1.VBR$) is shown in the example, as well as $p_1.MND$, which is the distance between p_1 and v_2.

Unlike the VOR-Tree which keeps all voronoi cell vertexes and delaunay neighbors for each facility, we only maintain the VBR and MND of a facility so that we can utilize the nice property of the voronoi cell with smaller index size (i.e., larger node capacity). Moreover, we can naturally keep the VBR and MND for a set of facilities and apply VBR and MND based pruning for intermediate entries in AVR-Tree.

In Section 3.2, we will introduce the data structure of AVR-Tree which integrates VBR and MND of the facilities to R-tree structure. Moreover, effective pruning techniques are proposed for NN and ANN queries in Section 4 and Section 5 respectively.

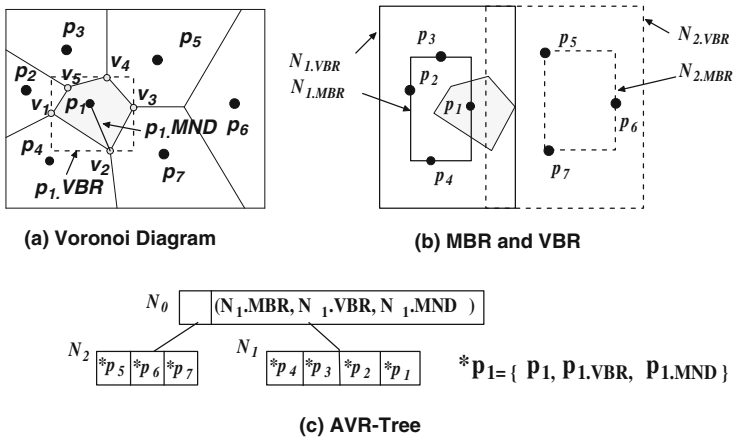


Fig. 4. AVR-Tree

3.2 AVR-Tree Structure

The Aggregate Voronoi R-tree (AVR-Tree for short) is an R-tree like structure which enriches with VBRs and MNDs of the facilities. A data entry of the AVR-Tree consists of a facility location and its VBR and MND. As to an intermediate entry, the MBR, VBR and MND are obtained from its descendant entries.

Fig. 4 illustrates an example of AVR-Tree for a set $\mathcal{F}$ of facilities $\{p_1, \ldots, p_7\}$, where there are 7 data entries $\{^*p_1, \ldots, ^*p_7\}$ and 3 intermediate entries $\{N_0, N_1, N_2\}$. Fig. 4(a) shows the Voronoi diagram of $\mathcal{F}$ and the data entries (e.g., $^*p_1$) can be constructed based on the corresponding voronoi cells. Fig. 4(b) depicts the MBRs and VBRs of the intermediate entries N_1 and N_2 where facilities $\{p_1, p_2, p_3, p_4\}$ are contained by N_1 and others are assigned to N_2. Note that the MNDs are also kept for N_1 and N_2. Fig. 4(c) shows the structure of the AVR-Tree.

3.3 AVR-Tree Construction

Construction of an AVR-Tree consists of two phases. In the first phase, we build an R-tree against the facilities following the standard R-tree algorithm [8]. In the second phase, the Voronoi diagram is constructed and the VBRs and MNDs information of the AVR-Tree entries are obtained in a bottom up fashion.

3.4 AVR-Tree Update

The insertion and deletion of a facility in an AVR-Tree is exactly the same as the R-tree except that we also need to recalculate the VBRs and MNDs of the facilities affected by the update, and then propagate the changes to their parent entries in a bottom up fashion.

4 Nearest Neighbor (NN) Query

In this Section, we introduce the AVR-Tree based nearest neighbor query. We first introduce the pruning technique, then the detailed algorithm.

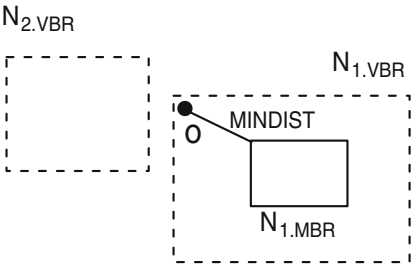


Fig. 5. Pruning Example for NN search

Theorem 1. *Given an object o and an AVR-Tree entry F (data entry or intermediate entry), F can be excluded from NN of o if $F.VBR \cap o = \emptyset$ or $F.MND < MINDIST(o, F.MBR$ [2]$)$.*

Proof. $F.VBR \cap o = \emptyset$ implies that $f.VBR \cap o = \emptyset$ for all facilities f within F since $F.VBR$ is the minimal bounding box of $\{f.VBR\}$ for all facilities f from F. On the other hand, an object o must fall in the voronoi cell of f if f is the NN of o. Since the voronoi cell of f is contained by $f.VBR$, we can safely claim that none of the facilities in F can be the nearest neighbor of o if $F.VBR \cap o = \emptyset$.

We prove the second case by the contradiction. Suppose a facility f from F is the NN of o, and then we have $\delta(o, f) \leq f.MND$. Consequently, we have $\delta(o, f) \leq F.MND$ since $f.MND \leq F.MND$. Clearly we have $MINDIST\ (o, F.MBR) \leq \delta(o, f)$ since $f \in F.MBR$, and hence $MINDIST(o, F.MBR) \leq F.MND$. This is against the assumption $MINDIST(o, F.MBR) > F.MND$. □

Example 4. Regarding the example in Fig. 5, we can immediately prune the entry N_2 since $N_2.VBR \cap o = \emptyset$. If $MINDIST(o, N_1.MBR) > N_1.MND$, N_1 can also be eliminated from NN candidate of the object o.

In Algorithm 1 we illustrate the details of the NN algorithm assuming the facilities $\mathcal{F}$ are organized by an AVR-Tree, represented by $\mathcal{F}_{AVR}$. The algorithm follows the *branch-and-bound* paradigm. A priority queue Q is employed to access the AVR-Tree entries where the key of an entry is its closest distance to the query object o. Consequently, algorithm can terminate when the first object is popped from Q (Line 1). For each intermediate entry popped from Q, Line 1 eliminates its non-promising child entries based on Theorem 1. Our empirical study demonstrates the efficiency of Algorithm 1 since a large number of non-promising entries can be pruned at low cost.

[2] If F is a data entry (i.e., facility), then the location of F is $F.MBR$.

Algorithm 1. ***NN*** $(o, \mathcal{F}_{AVR})$

```
Input  : o : an object o ,
         F_AVR : a set F of facilities organized by AVR-Tree
Output : the nearest neighbor of o in F
1  Q ← root of F_AVR ;
2  while Q ≠ ∅ do
3      E ← top element popped from Q ;
4      if E is a data entry then
5          return the facility f associated with E
6      else
7          for each child entry e of E do
8              if e.VBR ∩ o ≠ ∅ and MINDIST(o, e.MBR) ≤ e.MND then
9                  Push e into Q with key value MINDIST(o, e.MBR) ;
```

5 All Nearest Neighbor (ANN) Query

In this section, we investigate the problem of all nearest neighbor (ANN) query. We first introduce the AVR-Tree based pruning techniques, then the details of the ANN algorithm.

Theorem 2. *Given the MBR of a set of objects, denoted by $O.MBR$, and an AVR-Tree entry F (data entry or intermediate entry), F can be excluded from NN calculation of any object $o \in O.MBR$ if $F.VBR \cap O.MBR = \emptyset$ or $MINDIST\ (O.MBR, F.MBR) > F.MND$.*

Proof. The proof of this Theorem is similar to that of Theorem 1. We omit the details due to the space limitation.

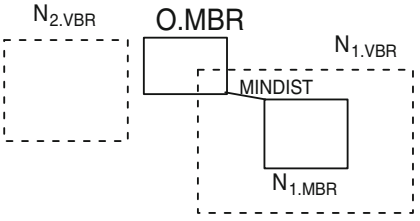


Fig. 6. Pruning Example for ANN query

Example 5. In Fig. 6 we can safely prune N_2 from the nearest neighbor computation for the objects in $O.MBR$ since $O.MBR \cap N_2.VBR = \emptyset$. Similarly, N_1 can be eliminated if $MINDIST\ (N_1.MBR, O.MBR) > N_1.MND$.

Algorithm 2 illustrates the details of the all nearest neighbor (ANN) query, where we assume a set $\mathcal{O}$ of objects and a set $\mathcal{F}$ of facilities are organized by R-tree $\mathcal{O}_R$ and AVR-Tree $\mathcal{F}_{AVR}$ respectively. Similar to spatial join [5,10], we apply the synchronized R-tree traversal paradigm to process ANN query in a level

by level fashion. In Algorithm 2, we use a tuple to maintain the object R-tree entry (intermediate or data entry), denoted by $T.O$, as well as a set of facility AVR-Tree entries (intermediate or data entries), denoted by $T.F$, which may contribute to the nearest neighbors for the objects within $T.O$. A FIFO queue, denoted by Q, is employed to maintain the tuples and is initialized as $<$ root of $\mathcal{O}_R$, root of $\mathcal{F}_{AVR}$ $>$ at Line 2. Line 2-2 iteratively process each tuple in the queue until it is empty. When a tuple T is popped from Q, we can come up with the nearest neighbor of the object if all entries in $T.O$ and $T.F$ are data entries (Line 2-2). Otherwise, Line 2 retrieves all child entries of $T.O$ to set $\mathcal{L}$. Similarly, the set $\mathcal{R}$ keeps all child entries of the entries in $T.F$ [3]. For each object entry $o_e \in \mathcal{L}$, we identify the facilities from $\mathcal{R}$ which may be the nearest neighbor of an object within o_e. Specifically, Theorem 2 is applied at Line 2 to eliminate non-promising facility entries regarding o_e. When Algorithm 2 terminates, the nearest neighbors of all objects in $\mathcal{O}$ are retrieved.

Algorithm 2. ***ANN***$(\mathcal{O}_R, \mathcal{F}_{AVR})$

```
   Input  : O_R : the objects organized by R-tree,
            F_AVR : the facilities organized by AVR-Tree
   Output : N : the nearest neighbors for all objects in O
1  Q ← tuple < root of O_R, root of F_AVR >;
2  while Q ≠ ∅ do
3      T ← the tuple popped from Q ;
4      if both T.O and T.F are data entries then
5          o ← object associated with T.O ;
6          Calculate distances between o and the facilities in T.F;
7          N ← < o, the nearest neighbor of o > ;
8      else
9          L := child entries of T.O ;
10         R := child entries of the entries in T.F ;
11         for each entry o_e ∈ L do
12             C = ∅ ;
13             for each entry f_e ∈ R do
14                 if f_e.VBR ∩ o_e.MBR ≠ ∅ and MINDIST(f_e.MBR, o_e.MBR)
                   ≤ f_e.MND then
15                     C := C ∪ f_e;
16             push < o_e, C > into Q ;
```

6 Performance Evaluation

We present results of a comprehensive performance study to evaluate the efficiency and scalability of the proposed techniques in the paper. Following algorithms are evaluated.

[3] Note that we have $L = T.O$ if $T.O$ is a data entry and facility entries are intermediate entries. The set $\mathcal{R}$ is handled in the same way.

R-tree the algorithms in which facilities are organized by R-tree. The NN and ANN queries are implemented based on the techniques proposed in [12] and [6] respectively.

VOR-Tree the algorithms in which facilities are organized by VOR-Tree [13].

AVR-Tree the algorithms in which facilities are organized by AVR-Tree. Particularly, the NN and ANN queries are implementation as described in Algorithm 1 and Algorithm 2 in the paper.

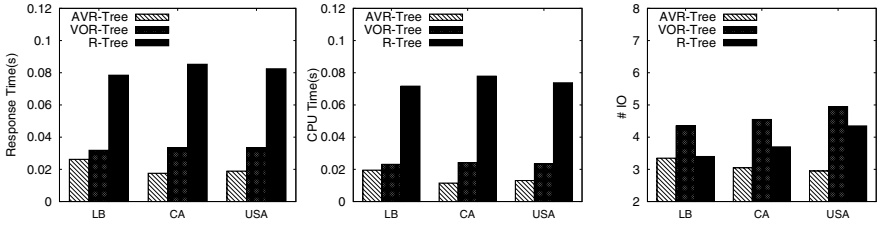
All algorithms in this paper are implemented in standard C++ with STL library support and compiled with GNU GCC. Experiments are performed on a PC with Intel Xeon 2.50GHz dual CPU and 2G memory running Debian Linux. The disk page size is fixed to 4096 bytes. The node capacity of the R-tree, AVR-Tree and VOR-Tree are 204, 102 and 40 respectively. Besides the MBR, we also keep the VBR (voronoi bounding rectangle) and MND (maximal nearest neighbor distance) information in each node of AVR-Tree. Therefore, the node capacity of AVR-Tree is smaller than that of R-tree. Similarly, we need to keep the voronoi neighbor vertexes and delaunay neighbor identifications for each location in VOR-Tree, and VOR-Tree has smallest node capacity. We use an LRU memory buffer with size 512K bytes.

Real Datasets. Three real spatial datasets, LB, CA and USA, are employed in the experiments. The facilities in CA are $45,671$ facility locations in California (`http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm`). Similarly, the facilities in USA are obtained from the U.S. Geological Survey (USGS) and consists of $205,323$ facility locations (`http://www.geonames.usgs.gov`). Meanwhile, the object locations of CA and USA are public available from `http://www.census.gov/geo/www/tiger`, where there are $62,556$ and 1M objects in CA and USA respectively. In LB, both facility and object locations are obtained from the Long Beach dataset from `http://www.census.gov/geo/www/tiger`, in which there are $53,145$ locations. In the above three datasets, all dimensions are normalized to domain $[0, 10000]$.

In the paper, the processing time, which includes the CPU time and I/O latency, is used to measure the efficiency of the algorithms. We also record the number of I/O accesses in the algorithms.

6.1 Evaluate Nearest Neighbor Search

In the first set of experiments, we evaluate the performance of three NN search algorithms based on R-tree, VOR-Tree and AVR-Tree against dataset LB, CA and USA, where $1,000$ query points are randomly chosen from $[0, 10000]^2$. The average query response time, CPU time and the number of I/O accesses for the three algorithms are reported in Fig. 7. It is shown that AVR-Tree outperforms R-tree and VOR-Tree on both CPU time (Fig. 7(b)) and I/O cost (Fig. 7(c)), and hence has the fastest query response time (Fig. 7(a)). As shown in Fig. 7(c) VOR-Tree incurs the largest number of I/O accesses due to its small node capacity. Nevertheless, its overall performance (i.e., query response time) is better

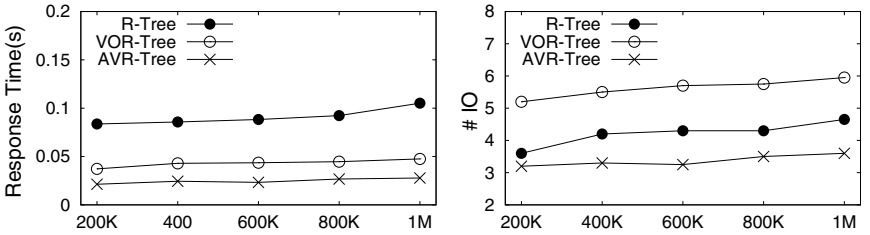


(a) *Response Time* (b) *CPU Time* (c) *# IO Accesses*

Fig. 7. NN Query on Different Datasets

than that of R-tree since it can take advantage of voronoi diagrams and use much less CPU time as shown in Fig. 7(b).

In the second set of experiments, we evaluate the scalability of the three algorithms where the facility locations are randomly chosen from *USA* object locations, with size ranging from 200K to 1M. Fig. 8 shows that the performance of the three algorithms are scalable towards the growth of the number of facilities.

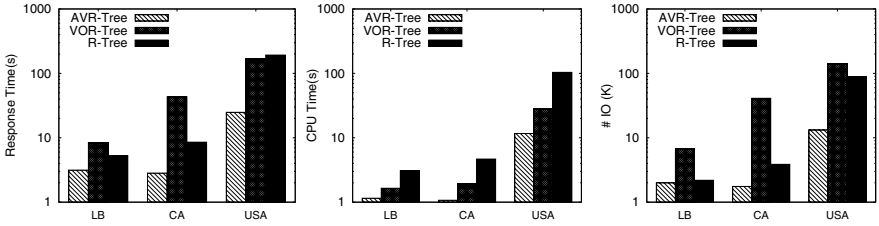


(a) *Response Time* (b) *# IO Accesses*

Fig. 8. Effect of the number of facilities

6.2 Evaluate All Nearest Neighbor (ANN) Query

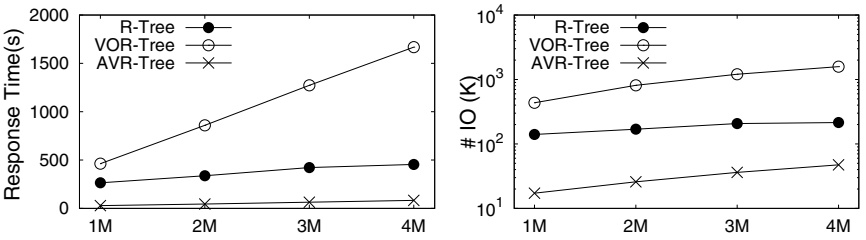
Fig. 9 evaluates the performance of R-tree, VOR-Tree and AVR-Tree Algorithms for all nearest neighbor (ANN) query, where three datasets *LB*, *CA* and *USA*



(a) *Response Time* (b) *CPU Time* (c) *# IO Accesses*

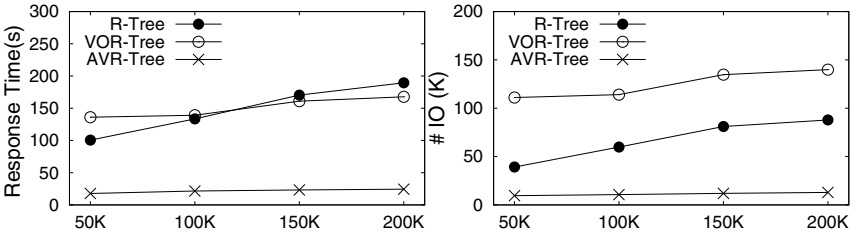
Fig. 9. ANN Query on Different Datasets

are employed. It is shown that the performance of AVR-Tree significantly outperforms R-tree and VOR-Tree. In *USA*, the query response time of AVR-Tree is 8 times faster than that of R-tree and VOR-Tree. Similar to NN query in Section 6.1, Fig. 9(b) and Fig. 9(c) show that VOR-Tree is more time efficient compared with R-tree, but invokes more I/O accesses due to its small node capacity. AVR-Tree demonstrates superior performance in comparison with R-tree and VOR-Tree because it can achieve a good trade-off between the pruning effectiveness and indexing overhead (e.g., node capacity).



(a) *Response Time* (b) *# IO Accesses*

Fig. 10. Effect of the number of objects



(a) *Response Time* (b) *# IO Accesses*

Fig. 11. Effect of the number of facilities

We also study the scalability of the three algorithms towards the growth of the number of facilities and objects in the ANN query. In Fig. 10, a set of object locations are randomly chosen from $[1, 100000]^2$ with size ranging from 1M to 4M. It is reported that the scalability of VOR-Tree is poor because it can only apply the MBR based pruning on facility and object entries at the intermediate level and the node capacity is small due to a large amount of information kept for each facility. As expected, AVR-Tree always ranks first when the number of objects grows. Similar observation is reported in Fig. 11, where a number of facilities are randomly chosen from *USA*.

6.3 Index Construction

In this subsection, we evaluate the index size and index construction time of R-tree, on the facilities of the three datasets *LB*, *CA* and *USA*. Fig. 12 reports the size of the three indexing structures on *LB*, *CA* and *USA*. As expected,

VOR-Tree has the largest index size, followed by AVR-Tree and R-tree. The construction time of the three index structures is illustrated in Fig. 13. As the dominant cost of VOR-Tree and AVR-Tree construction is the R-tree construction, their performance is slightly slower than that of R-tree. Note that the voronoi diagram and delaunay graph computation algorithms are public available from Qhull (`http://www.qhull.org/`).

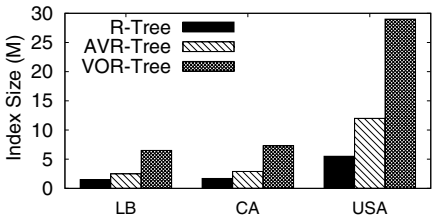


Fig. 12. Index Size

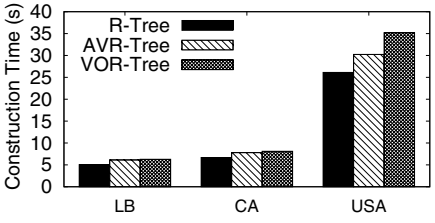


Fig. 13. Construction Time (s)

7 Related Work

The family of the nearest neighbor search problems consists of nearest neighbor (NN) search [14,8] or top k nearest neighbor search [12,3], all nearest neighbor (ANN) search [15,6], reverse nearest neighbor [11] and approximate nearest neighbor [1] and etc. Our work focuses on the NN and ANN queries on location data. The problem of NN query on location data has a wide spectrum applications and has been extensively studied in the literature. Roussopoulos *et al.* [12] propose to use R-tree [8] as the index structure for nearest neighbor search in low dimensional space. [3] proposes a tree-based index to solve nearest neighbor search in high dimensional space. Recently, M. Sharifzadeh *et al.* [13] propose a new structure, VOR-Tree, where it is an R-tree with voronoi cells stored in its data entries. The data entries also store the information of the neighboring cells of each cell.

Our work also relates to ANN search. One of the earliest work proposed by Braumuller *et al.* [4] is to perform one NN search on the facility data set for each data in the object data set. Zhang *et al.* [15] propose an R-tree based approach to search the nearest neighbors as a batch. Corral et al. [7] and Hjaltason et al. [9] propose a technique based on R*-tree structure [2]. Corral *et al.* [7] also propose a pruning rule and two updating strategies to efficiently solve the search problem. Recently, Y. Chen et al. [6] propose a tighter upper bound distance between two MBRs so that the pruning is more effective in order to retrieve all nearest neighbors more efficiently.

8 Conclusion

In this paper, we introduce a new structure, termed AVR-Tree, to organize location data, which utilizes the R-tree and Voronoi diagram techniques. Efficient

nearest neighbor (NN) and all nearest neighbor (ANN) algorithms are proposed in the paper and comprehensive experiments are conducted to demonstrate the effectiveness and efficiency of the algorithms.

Acknowledgement. Ying Zhang is supported by ARC DP110104880, ARC DP130103245 and UNSW ECR grant PS27476. Wenjie Zhang is supported by ARC DP120104168 and ARC DE120102144. Xuemin Lin is supported by ARC DP0987557, ARC DP110102937, ARC DP120104168 and NSFC61021004.

References

1. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. J. ACM 45(6), 891–923 (1998)
2. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The r*-tree: An efficient and robust access method for points and rectangles. In: SIGMOD Conference, pp. 322–331 (1990)
3. Berchtold, S., Ertl, B., Keim, D.A., Kriegel, H.-P., Seidl, T.: Fast nearest neighbor search in high-dimensional space. In: ICDE, pp. 209–218 (1998)
4. Braunmüller, B., Ester, M., Kriegel, H.-P., Sander, J.: Efficiently supporting multiple similarity queries for mining in metric databases. In: ICDE, pp. 256–267 (2000)
5. Brinkhoff, T., Kriegel, H.-P., Seeger, B.: Efficient processing of spatial joins using r-trees. In: SIGMOD Conference, pp. 237–246 (1993)
6. Chen, Y., Patel, J.M.: Efficient evaluation of all-nearest-neighbor queries. In: ICDE, pp. 1056–1065 (2007)
7. Corral, A., Manolopoulos, Y., Theodoridis, Y., Vassilakopoulos, M.: Algorithms for processing k-closest-pair queries in spatial databases. Data Knowl. Eng. 49(1), 67–104 (2004)
8. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD Conference (1984)
9. Hjaltason, G.R., Samet, H.: Incremental distance join algorithms for spatial databases. In: SIGMOD Conference, pp. 237–248 (1998)
10. Huang, Y.-W., Jing, N., Rundensteiner, E.A.: Spatial joins using R-trees: Breadth-first traversal with global optimizations. In: VLDB 1997 (1997)
11. Korn, F., Muthukrishnan, S.: Influence sets based on reverse nearest neighbor queries. In: SIGMOD Conference, pp. 201–212 (2000)
12. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In: SIGMOD Conference, pp. 71–79 (1995)
13. Sharifzadeh, M., Shahabi, C.: Vor-tree: R-trees with voronoi diagrams for efficient processing of spatial nearest neighbor queries. PVLDB 3(1), 1231–1242 (2010)
14. Weber, R., Schek, H.-J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB, pp. 194–205 (1998)
15. Zhang, J., Mamoulis, N., Papadias, D., Tao, Y.: All-nearest-neighbors queries in spatial databases. In: SSDBM, pp. 297–306 (2004)

Top-k Neighborhood Dominating Query

Xike Xie[1], Hua Lu[1], Jinchuan Chen[2], and Shuo Shang[1]

[1] Department of Computer Science, Aalborg University, Denmark
{xkxie,luhua,sshang}@cs.aau.dk
[2] Key Labs of Data Engineering and Knowledge Engineering MOE,
Renmin University of China, China
jcchen@ruc.edu.cn

Abstract. In many real-life applications, spatial objects are associated with multiple non-spatial attributes. For example, a hotel may have price and rating in addition to its geographic location. In traditional spatial databases, spatial objects are often ranked solely based on their distance to a given query location, e.g., in a nearest neighbor search. In another line of research, domination based skyline queries are used to return best objects according to multi-criteria on non-spatial attributes. In this paper, we study how to rank spatial objects with respect to their non-spatial attributes within their spatial neighborhoods. To enable a general ranking, we design a ranking function that inherits the advantages of dominance relationship and integrates them with spatial proximity. Further, we propose an effective index structure, and a branch and bound solution that executes the ranking efficiently via the index. We conduct extensive empirical studies on real and synthetic datasets. The results demonstrate the high efficiency of our proposal compared to straightforward alternatives.

1 Introduction

Spatial objects are associated with multiple non-spatial attributes in many real-life applications. Non-spatial attributes capture the quality or properties of spatial objects whose locations are typically represented in coordinate values (e.g., longitude and latitude). Without loss of generality, we abstract a spatial object o by two sequences of attributes: 2-dimension spatial attributes and d-dimension non-spatial attributes. Formally, we have $o = (o.\eta, o.\psi)$, where $\eta = (x, y)$ denotes the spatial location, and $\psi = (\psi[1], \ldots \psi[d])$ denotes the non-spatial attributes which we also call *quality attributes*.

In traditional spatial databases, spatial objects are retrieved solely based on spatial locations and the implied spatial distances. Typical examples are range queries, nearest neighbor queries, and spatial joins. On the other hand, dominance relationship has been extensively studied and used to compare objects in skyline queries [2] [8] [16] [9] [19] and data analysis [15] [10] [3]. Given two objects p and q, p is said to *dominate* q ($p \succ q$), if p is no worse than q in all quality attributes and better than q in at least one quality attribute. Without loss of generality, we assume that all quality attributes are defined on numeric domains, and "better than" means "larger than". The formal definition of dominance is given as follows.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 131–145, 2013.

Definition 1. (Dominance) *Object p dominates object q, denoted as $p \succ q$, if* $\forall i$, $p.\psi[i] \geq q.\psi[i]$ *and* $\exists j, p.\psi[j] > q.\psi[j]$.[1]

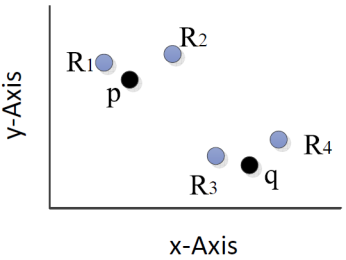
Given a set P of objects, a skyline query returns all those objects in P that are not dominated. The key advantage of the dominance concept based skyline query is that it does not require any special ranking function to compare multiple attributes. Nevertheless, the dominance concept can be employed to rank objects according the number of objects they dominate [19], or that of objects dominating them [11].

Although spatial locations and non-spatial attributes are heterogeneous in nature, it is useful to combine them to formulate novel types of queries that serve practical user needs. For example, hotel investors are interested in investing in those hotels that dominate their neighbor hotels because such advantaged hotels have higher potential to make profit. To find such advantaged hotels, both spatial locations and quality attributes should be involved in a query. However, neither a traditional spatial query nor a pure skyline query can meet such practical needs that integrate the two heterogeneous aspects. A spatial query does not consider the quality attributes, whereas a skyline query ignores the impact of the neighborhood determined by objects' spatial locations.

An example is shown in Figure 1. If we run a skyline query on these hotels listed in Figure 1 (a), $\{p, q, R_3\}$ will be returned since they are not dominated by others while all others are dominated by them. However, from a investor's perspective, p is more interesting as it dominates more neighbor hotels than q does, as shown in Figure 1 (b). Specifically, p dominates both R_1 and R_2 in its neighborhood whereas q only dominates R_4 in its neighborhood. On the other hand, a pure spatial query (e.g., a range query, a nearest neighbor query, or any pure counting query based on them) cannot differentiate p and q to serve the investor's interest.

Restaurant	Number of Rooms	Stars
q	400	4
p	300	5
R_1	200	4
R_2	300	3
R_3	500	2
R_4	300	4

(a) non-spatial attributes



(b) spatial attributes

Fig. 1. Example

To bridge the gap between practical needs and existing techniques, we study a generic ranking mechanism for spatial objects with quality attributes in this paper. Particularly, we focus on the *Top-k neighborhood dominating query*, which returns k spatial objects

[1] Although the dominance definition employs ">", the proposed techniques in this paper can also be applied to scenarios where lower values are preferred. As pointed out in the literature [2], a simple negation operator on the original values will make such scenarios compatible.

with highest ranking scores. The score of an object is defined in a generic and flexible manner to reflect its dominating capability in its neighborhood. Users are given the flexibility to tailor the score function to meet various scenarios that include the one exemplified above.

Our contributions in the paper are summarized as follows.

- We define a general score function for ranking spatial objects with respect to their neighborhood dominating capability, and give a basic solution (Section 2).
- We derive the upper/lower bounds for the score function to speed up the ranking evaluation (Section 3).
- We design the Ψ-augmented R-tree to further improve the ranking efficiency (Section 4).
- We demonstrate the superiority of our methods through extensive experiments (Section 5).

In addition, we review related work in Section 6 and conclude the paper in Section 7. The Notations used throughout this paper are listed in Table 1.

Table 1. Notations

Notation	Meaning
O	a set of objects $(o_1, o_2, \ldots, o_n)$
$\|O\|$	the size of database O
D	Domain of spatial attributes
Ψ	Domain of non-spatial attributes
$\|a, b\|$	the Euclidean distance between a and b
$\|E\|$	the number of objects inside R-tree node E
$o.\eta(x, y)$	the spatial attributes of o ($\langle x, y\rangle$)
$o.\psi$	the quality attributes of o
$p \succ q$	p dominates q or $p.\psi$ dominates $q.\psi$
$p \prec q$	p is dominated by q or $p.\psi$ is dominated by $q.\psi$
$\phi(o)$	ranking score of object o
ϕ^-/ϕ^+	the upper / lower bounds of the ranking score
ω_E	the augmented distribution function of R-tree node E
ω_E^-/ω_E^+	the upper / lower bounds of ω_E

2 Problem Definition and Basic Algorithm

We define the query in Section 2.1, and give a basic algorithm in Section 2.2.

2.1 Problem Definition

We start by briefly reviewing several score functions. Based on that, we propose our own score functions. Yiu and Mamoulis [19] quantify the dominating capability of an object s by the number of objects it dominates:

$$\phi'(s) = \Big|\{o \in O \big| s.\psi \succ o.\psi\}\Big| \tag{1}$$

In other words, the dominance score $\phi'(s)$ is the number of objects dominated by s. However, the spatial attributes are not considered in this definition.

Yiu et al. [17,18] propose to assign different weights $w(.)$ to different dimensions of non-spatial attributes and integrate them into a score function. As a flavor to the score function, the spatial neighborhood can be emphasized by a general ripple coefficient, $2^{-\frac{|p,q|}{\epsilon}}$ [18]. $|p,q|$ denotes the Euclidean distance between p and q, and ϵ is a user-specified parameter for the spatial range. The ripple coefficient gives a higher value if p and q are close and a lower value otherwise. The score function with the ripple coefficient is as follows.

$$\phi''(s) = AGG_i\Big\{max\{w(o.\psi[i]) \cdot 2^{-\frac{|s,o|}{\epsilon}}\}\Big\} \tag{2}$$

However, users have to specify the weights to integrate all non-spatial dimensions in the score function above. Consequently, the key benefit of dominance and skyline queries diminishes. Our purpose in this paper is to define a ranking (score) function ϕ that integrates the advantages of both ϕ' and ϕ''. For two objects s and o, we can define the *partial score* of s with respect to o as:

$$\textbf{Partial Score: } \phi(s,o) = \underbrace{2^{-\frac{|s,o|}{\epsilon}}}_{Spatial\ Factor} \cdot \underbrace{\tau(s,o)}_{Dominance\ Indicator} \tag{3}$$
$$where \quad \tau(o,s) = \begin{cases} 1 & s.\psi \succ o.\psi \\ 0 & otherwise \end{cases}$$

Let the product of spatial factor and dominance factor be $\phi(s,o)$, meaning how much weight for object o to add to s's dominance, the *score function* of s is the summation of partial score function for all other objects in the database O:

$$\textbf{Ranking Score: } \phi(s) = \sum_{o \in O} \phi(s,o) \tag{4}$$

Our score function ϕ inherits ϵ-ripple from ϕ'' (spatial factor) and the dominance quantification from ϕ' (dominance factor). Unlike ϕ'', ϕ does not have to assign different weights to different dimensions, which preserves the advantage of skyline query. Also, ϕ' can be viewed as a special case of ϕ while ϵ is infinite. Since the limit of $2^{-\frac{|p,q|}{\infty}}$ is 1, we have:

$$\lim_{\epsilon \to \infty} \phi(s) = \sum_{o \in O} \tau(s,o) = \Big|\{o \in O \big| s.\psi \succ o.\psi\}\Big| = \phi' \tag{5}$$

Thus, by defining ϕ-function, we bridge the gap between the dominance query and the neighborhood dominance query. Based on this generic score function, we give the formal problem definition as follows:

Definition 2. *Given a database O of spatial objects, a* ***top-k neighborhood dominating query*** *(k-NDQ in short) retrieves a set $S \subseteq O$ of k objects such that $\forall o_i \in S, \forall o_j \in O \setminus S$, $\phi(o_i) \geq \phi(o_j)$.*

2.2 Basic Solution

Intuitively, the query can be answered by iterating all objects in O and computing exact $\phi(s)$ for each encountered object s. Meanwhile, a global max-heap H_k is used to maintain the current top-k results in terms of the scores. After all object in O has been processed, the query returns the k objects in H_k.

To compute the score $\phi(s)$, a straightforward way is to iterate over all other objects in O one by one. This brute-force way is called Basic and is formalized in Algorithm 1. Apparently, the basic algorithm is expensive as it examines all objects but s in database O in order to compute the score for s. In the following sections, we propose more efficient ways to compute $\phi(s)$.

Algorithm 1. Basic

```
1: function BASIC(Object s)
2:     φ(s) ← 0;
3:     for each object o ∈ O ∧ o ≠ s do
4:         Compute φ(s, o);
5:         Add φ(s, o) to φ(s);
6:     Update H_k with φ(s) if necessary;
```

3 Efficient kNDQ Evaluation

In this section, we propose an efficient algorithm to compute $\phi(s)$. It employs a R-tree to significantly reduce the number of objects to be accessed. The main idea is to estimate the upper/lower bounds (*ULBounds* in short) for the current object s's score. If its upper bound of ranking score $\phi^+(s)$ is below the current top-k score γ_k, the computation of $\phi(s)$ is immediately stopped. By using ULBounds, considerable score computation cost can be saved.

In Section 3.1, we propose a branch-and-bound framework for computing $\phi(s)$. After that, we further discuss two key components of the algorithm: upper/lower bounds and the partial order property in Section 3.2.

3.1 kNDQ Framework

This framework uses two global parameters: a max-heap H_k for storing current top-k scores and γ_k is the k-th highest score in the heap. Initially, H_k is empty, γ_k is 0. Suppose spatial objects in O are organized in a hierarchical structure (e.g., a R-tree). The ranking score of object s can be computed in a top-down manner, starting at the R-tree root node. Subsequently, tree nodes are accessed recursively. When accessing a tree node E, its *tree node partial score* $\phi(s, E)$ is computed.

Definition 3. (Tree Node Partial Score) *Given an object s and R-tree node E, the score of s with respect to E is*

$$\phi(s, E) = \sum_{e \in E} \phi(s, e) \tag{6}$$

Definition 3 is recursive. If E is a leaf node, $\phi(s, E)$ returns all partial scores of s with respect to all objects in E, i.e., $\phi(s, E) = \sum_{o \in E} \phi(s, o)$. If $\phi(s, E)$ can be well estimated by its upper/lower bounds without retrieving $o \in E$, we can use the score bounds to update current $\phi(s)$. If its upper bound $\phi(s)^+ < \gamma_k$, there is no need to access the subtree of E. The computation is significantly reduced.

To compute ULBounds of tree node partial scores, we store in each tree node with a vector of aggregated values $\langle count, \psi^l, \psi^u \rangle$[2], where $\psi^l[i] = min_{o \in E}\{o.\psi[i]\}$ and $\psi^u[i] = max_{o \in E}\{o.\psi[i]\}$. How to derive the ULBounds is detailed in Section 3.2. Next, we give the branch-and-bound framework for computing $\phi(s)$ in the query processing.

The pseudocode of the framework is shown in Algorithm 2. A max-heap is used to keep the entries to be visited. Entries are sorted according to the upper bounds of partial scores, as higher valued scores tend to contribute more to $\phi(s)^+$. Thus, the branch and bound process would stop earlier. At each iteration, a node E is visited. First, we treat ULBounds of $\phi(s, E)$ as the *Original Form* and reduce them from $\{\phi^-(s), \phi^+(s)\}$ (line 7). Second, we expand E and derive ULBounds of $\phi(s, e)$ for all child nodes $\{e \in E\}$ (line 9-12, 14-17). The summation of $e \in E$'s ULBounds is called *Expanded Form*, as it is the result of expanding E.

Algorithm 2. kNDQ

```
1:  function kNDQ(RootNode R, object s)
2:      H is a max-heap ranking on the upper bounds of partial score;
3:      {φ⁻(s), φ⁺(s)} ← {φ⁻(s, R), φ⁺(s, R)};
4:      push_heap(H, R);
5:      while H is not empty do
6:          E ← pop_heap(H, R);
7:          Subtract φ⁻(s, E) from φ⁻(s); Subtract φ⁺(s, E) from φ⁺(s);
8:          if E is a non-leaf node then //Expand E
9:              for each entry e ∈ E do
10:                 Add φ⁻(s, e) to φ⁻(s); Add φ⁺(s, e) to φ⁺(s);
11:                 if γ_k ≻ φ⁺(s) then
12:                     return;
13:         else // E is a leaf node; Expand E
14:             for each object o ∈ E do
15:                 Add φ⁻(s, o) to φ⁻(s); Add φ⁺(s, o) to φ⁺(s);
16:                 if γ_k ≻ φ⁺(s) then
17:                     return;
18:     if γ_k ⊁ φ(s) then
19:         Update H_k and γ_k;
```

Each time E is accessed, we transform ULBounds of $\phi(s, E)$ from original form into expanded form. Meanwhile, ULBounds of $\phi(s)$ become tighter. Actually, they can converge to $\phi(s)$ after all iterations. In the sequel, we study how to derive the ULBounds and prove that the expanded form is tighter than the original form (*partial order property*).

[2] Such an aggregated R-tree is implemented based on the R*-tree [1].

3.2 Spatial Dominance Upper/Lower Bounds

We derive the ULBounds for a tree node partial score $\phi(s, E)$. They can be plugged in Algorithm 2 to speed up the query processing. We discuss different ULBounds according to different types of E. The spatial factor could be founded by the minimum or maximum distance between s and E. The dominance factor could be bounded by the number of objects inside node E, denoted by $|E|$. Then, for each bound, we derive the two forms, original form and expanded form, as aforehand mentioned. The reasonability of the transition between two forms is also illustrated.

ULBounds for Leaf Nodes

Lemma 1. *Given an object s and a leaf node e, the upper bound of $\phi(s, e)$ is:*

$$\textit{Original Form: } \phi^+(s,e) = 2^{-\frac{|s,e|_{min}}{\epsilon}} \cdot |e| \tag{7}$$

$$\textit{Expanded Form: } \phi^+(s,e) = \phi(s,e) = \sum_{o \in e} \begin{cases} 2^{-\frac{|s,o|}{\epsilon}} & if\ s \succ o \\ 0 & if\ s \not\succ o \end{cases} \tag{8}$$

Proof. For original form, we overestimate $\phi(s, e)$'s spatial factor to $2^{-\frac{|s,e|_{min}}{\epsilon}}$, and overestimate $\phi(s, e)$'s dominance factor to $|e|$, assuming that s dominates all objects in e. Hence, the original form is $\phi(s, e)$'s upper bound. The expanded form of $\phi^+(s, e)$ degenerates into the partial score $\phi(s, e)$, which can be considered as a trivial upper bound.

Lemma 2. *Given an object s and a leaf node e, the lower bound of $\phi(s, e)$ is:*

$$\textit{Original Form: } \phi^-(s,e) = \begin{cases} 2^{-\frac{|s,e|_{max}}{\epsilon}} & if\ s \succ e.\psi^u \\ 0 & otherwise \end{cases} \tag{9}$$

$$\textit{Expanded Form: } \phi^-(s,e) = \phi(s,e) = \sum_{o \in e} \begin{cases} 2^{-\frac{|s,o|}{\epsilon}} & if\ s \succ o \\ 0 & if\ s \not\succ o \end{cases} \tag{10}$$

Proof. For the original form, we underestimate $\phi(s)$'s spatial factor by $2^{-\frac{|s,e|_{max}}{\epsilon}}$. If s dominates all objects in e ($s \succ e.\psi^u$), the dominance factor equals to 1. Otherwise the dominance factor of $\phi(s, o)$ is underestimated as 0. Hence, the original form is $\phi(s, e)$'s lower bound. The expanded form of $\phi^-(s, e)$ equals to $\phi(s, e)$, so it is a trivial lower bound.

ULBounds for Non-leaf Nodes

Lemma 3. *Given an object s and a non-leaf node E, the upper bound of $\phi(s, E)$ is:*

$$\textit{Original Form: } \phi^+(s,E) = 2^{-\frac{|s,E|_{min}}{\epsilon}} \cdot |E| \tag{11}$$

$$\textit{Expanded Form: } \phi^+(s,E) = \sum_{e \in E} \begin{cases} 2^{-\frac{|s,e|_{min}}{\epsilon}} \cdot |e| & if\ s \succ e.\psi^u \\ 0 & if\ s \prec e.\psi^l \end{cases} \tag{12}$$

Lemma 4. *Given an object s and a non-leaf node E, the lower bound of $\phi(s, E)$ is:*

$$\textit{Original Form: } \phi^-(s,E) = \begin{cases} 2^{-\frac{|s,E|_{max}}{\epsilon}} \cdot |E| & if\ s \succ E.\psi^u \\ 0 & otherwise \end{cases} \tag{13}$$

$$\textit{Expanded Form: } \phi^-(s,E) = \sum_{e \in E} \begin{cases} 2^{-\frac{|s,e|_{max}}{\epsilon}} \cdot |e| & if\ s \succ e.\psi^u \\ 0 & if\ s \prec e.\psi^l \end{cases} \tag{14}$$

The proofs of Lemma 3 and 4 are omitted due to page limit.

Partial Order Property. The expanded form of ULBounds must be tighter than the original form counterparts. This is guaranteed by the following lemma.

Lemma 5. *For a parent node E and its children $\{e \in E\}$, we have:*

$$\begin{cases} \sum_{e \in E} \phi^+(s,e) \leq \phi^+(s,E) & \textit{(Upper Bound Partial Order)} \\ \sum_{e \in E} \phi^-(s,e) \geq \phi^-(s,E) & \textit{(Lower Bound Partial Order)} \end{cases} \tag{15}$$

Proof. Let's consider upper bound partial order.
1) E is a leaf node:

$$\text{(Expanded form)} \sum_{o \in e} \begin{cases} 2^{-\frac{|s,o|}{\epsilon}} & if\ s \succ o \\ 0 & if\ s \not\succ o \end{cases} \leq \sum_{o \in e} 2^{-\frac{|s,o|}{\epsilon}}$$
$$\leq \sum_{o \in e} 2^{-\frac{|s,e|_{min}}{\epsilon}} = 2^{-\frac{|s,e|_{min}}{\epsilon}} \cdot |e| \text{ (Original Form)}$$

2) E is a non-leaf node:

$$\text{(Expanded form)} \sum_{e \in E} \begin{cases} 2^{-\frac{|s,e|_{min}}{\epsilon}} \cdot |e| & if\ s \succ e.\psi^u \\ 0 & if\ s \prec e.\psi^l \end{cases} \leq \sum_{e \in E} 2^{-\frac{|s,e|}{\epsilon}}$$
$$\leq \sum_{e \in e} 2^{-\frac{|s,E|_{min}}{\epsilon}} = 2^{-\frac{|s,E|_{min}}{\epsilon}} \cdot |E| \text{ (Original Form)}$$

The lower bound partial order can be proved in a similar manner. Thus, the lemma is proved.

With the partial order property (Lemma 5), we can infer that the more entries we visit, the tighter ULBounds will be. Thus, it is guaranteed that by expanding tree nodes in Algorithm 2, the ULBounds gradually converge. Specially, when a leaf node e is expanded, the upper/lower partial scores of s with respect to e converge to the partial score $\phi(s, e)$. However, the converge of ULBounds can be slow, if the dominance factor is only roughly estimated by aggregated values $\langle \psi^l, \psi^u \rangle$. In the sequel, we study how to derive tighter ULBounds by better estimating the dominance factor.

4 Ψ-Augmented R-Tree and Ψ-Augmented ULBounds

In this section, we improve the estimation of the dominance factor by constructing an auxiliary structure, Ψ-augmented R-tree. Section 4.1 introduces the index structure as well as its construction. Section 4.2 investigates how to use the index to improve the query efficiency.

4.1 Dominance Histogram and Ψ-Augmented R-Tree

Rather than just storing the range of $E.\psi$, we augment each R-tree node E with the distribution of non-spatial values for objects belonging to E. We denote the distribution function of E by ω_E. Formally,

$$\omega_E : \Psi^d \rightarrow N \tag{16}$$

Given a range $[\psi_1, \psi_2]$ $(\psi_1, \psi_2 \in \Psi)$, $\omega_E[\psi_1, \psi_2]$ returns the number of objects in E's subtree whose non-spatial attributes are within the range.

$$\omega_E[\psi_1, \psi_2] = \left|\{o \mid o.\psi \in [\psi_1, \psi_2] \wedge o \in E\}\right| \tag{17}$$

In implementation, ω_E can be represented by a set of discrete histogram bars. For each histogram bar b, it stores its range $[\psi_{b^l}, \psi_{b^u}]$ as well as the number of objects in the range, denoted by $b.count$ (or $\omega_E[\psi_{b^l}, \psi_{b^u}]$ equivalently). We call ω_E as the *dominance histogram* of node E.

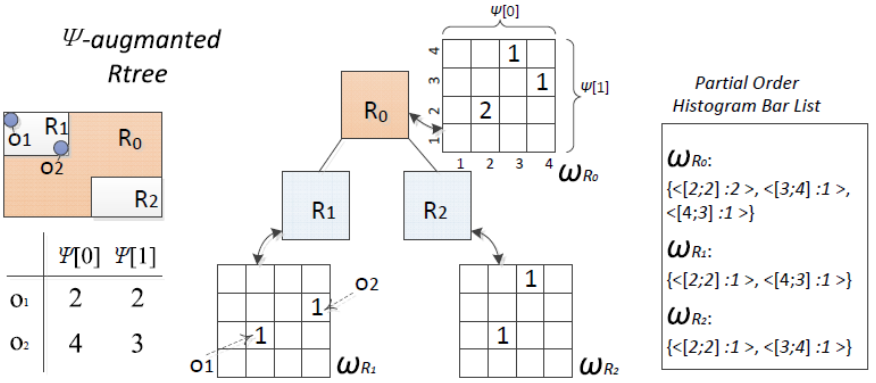


Fig. 2. Ψ-augmented R-tree

We show a dominance histogram with 2 non-spatial dimensions in Figure 2. Each R-tree node R_i is augmented with a histogram ω_{R_i}. For each dimension of Ψ, we partition it into equal-width intervals. In Figure 2, both dimension $\Psi[1]$ and $\Psi[2]$ are partitioned into 4 intervals (4×4 bars in total). Then, we count the number of objects belonging to each histogram bar. For example, leaf node R_1 has two objects o_1 and o_2. In ω_{R_1}, two bars are marked with "1", indicating the existence of o_1 and o_2. The parent node R_0's histogram ω_{R_0} can be collected by merging ω_{R_1} and ω_{R_2}. A non-leaf node's bar count equals to the summation of corresponding children's bar counts. For example, $\omega_{R_0}.[2;2] = 2$, which is the summation of $\omega_{R_1}.[2;2]$ and $\omega_{R_2}.[2;2]$. Formally,

$$\omega_E[\psi_{b^l}, \psi_{b^u}] = \sum_{e \in E} \omega_e[\psi_{b^l}, \psi_{b^u}] \tag{18}$$

Suppose $\mathcal{B}$ is the fanout of R-tree, then the total number of nodes in R-tree is $N = \sum_{i=1 \wedge \frac{|O|}{\mathcal{B}^i} \geq 1} \frac{|O|}{\mathcal{B}^i}$ [4]. If each dimension is split into m equal-width intervals, the space complexity of the tree is $O(Nm^d)$ and the time complexity is $O(N\mathcal{B}m^d)$.

We notice that in multi-dimensional space the non-spatial attributes are often distributed sparsely. Instead of keeping all the histogram bars, we can maintain a *partial order histogram bar list* for each R-tree node. The elements of the bar list are sorted according to the partial order on "Ψ-coordinates". For example, bar $\omega_{R_0}.[2;2]$ is a Ψ-coordinate. To distinguish from $\prec$, we use "$\prec_c$" to represent the dominance relationship between two Ψ-coordinates.

Definition 4. $\prec_c$**.** *Given two Ψ-coordinates $a = [a_1; \ldots; a_d]$ and $b = [b_1; \ldots; b_d]$, we say $a \prec_c b$, if $\forall i, a_i \leq b_i$ and $\exists 1 \leq j \leq d, a_j < b_j$.*

In ω_{R_0}, $[2;2] \prec_c [3;4]$ and $[2;2] \prec_c [4;3]$, so they are sorted as $\langle [2;2], [3;4], [4;3] \rangle$. Then, to collect ω_{R_0}'s partial order list, we merge the sorted lists of ω_{R_1} and ω_{R_2}. Let the length of the partial order list be $O(l)$ (usually $l \ll m^d$). By using partial order histogram bar lists, we reduce the space and time complexity to $O(Nl)$ and $O(N\mathcal{B}l)$, respectively.

Ψ-Augment a R-Tree. Given a constructed R-tree, we can augment it with the partial order histogram in a recursive manner, as detailed in Algorithm 3. The algorithm returns the histogram bar list of the input node. Initially, the input is set to be the root of R-tree. The algorithm tests if the input tree node is a leaf node or not. If it is a leaf, line 4-10 calculates the partial order list. Line 7 and 10 can be done in $Olog(l)$, because the list is sorted. Otherwise, it is a non-leaf node, all its children's lists are merged and returned. The cost of merging $O(\mathcal{B})$ sorted lists is $O(\mathcal{B}l)$.

Algorithm 3. Ψ-augment

```
1: function Ψ-AUGMENT(Node E)
2:     if E is a non-leaf node then
3:         return Merge_{e∈E} Ψ-augment(e);
4:     else  //E is a leaf node
5:         for each object o ∈ E do
6:             project o.ψ into bar b[ψ_{b^l}, ψ_{b^u}], where o.ψ ∈ [ψ_{b^l}, ψ_{b^u}];
7:             if b exists in E's partial order list then
8:                 b.count ++
9:             else
10:                insert b into list and set b.count to be 1;
11:        return E's partial order list;
```

4.2 Ψ-Augmented Upper/Lower Bounds

Now we discuss how the Ψ-augmented R-tree is used to estimate the dominance factor in processing kNDQ. Due to the limited granularity of histogram bars, the dominance factor can be estimated by a interval. Given the histogram constructed on node E, we use $B_E^-[\psi^l, \psi^u]$ to represent bars covered by interval $[\psi^l, \psi^u]$, and $B_E^+[\psi^l, \psi^u]$ to represent bars overlapping with interval $[\psi^l, \psi^u]$:

$$\begin{cases} B_E^-[\psi^l, \psi^u] = \{b_i \mid [\psi_{b^l}, \psi_{b^u}] \subseteq [\psi^l, \psi^u]\} \\ B_E^+[\psi^l, \psi^u] = \{b_i \mid [\psi_{b^l}, \psi_{b^u}] \cap [\psi^l, \psi^u] \neq \emptyset\} \end{cases} \tag{19}$$

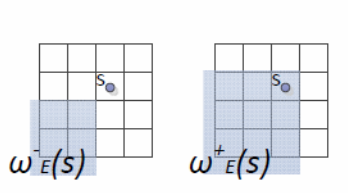


Fig. 3. $\omega^{-}(s)$ and $\omega^{+}(s)$

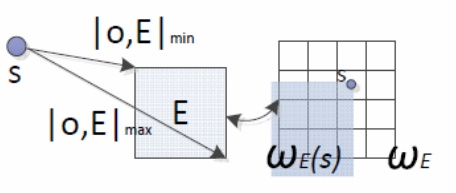


Fig. 4. Upper/lower Bound

Then, we can count the We use $\omega_E^{+}(s)$ (or $\omega_E^{-}(s)$) to denote $\omega_E^{+}[0, s.\psi]$ (or $\omega_E^{-}[0, s.\psi]$). An example is shown in Figure 3, where $\omega_E^{+}(s)$ is the number of objects over shaded bars. In other words, $\omega_E^{+}(s)$ denotes the largest number of objects s can possibly dominate inside node E. Then, $\omega_E^{+}(s)$ and $\omega_E^{-}(s)$ can be written as:

$$\begin{cases} \omega_E^{-}(s) = \sum_{b \in B_E^{-}[0,s.\psi]} b.count \\ \omega_E^{+}(s) = \sum_{b \in B_E^{+}[0,s.\psi]} b.count \end{cases} \tag{20}$$

We can obtain $\omega_E^{+}(s)$ and $\omega_E^{-}(s)$ by scanning E's partial order histogram bar list. Given object s, and the histogram bar width W_i on i-th non-spatial dimension, b's Ψ-coordinate can be calculated by $\{b_i = \lfloor \frac{s.\psi[i]}{W_i} \rfloor\}_{i=1}^{d}$. Meanwhile, we collect b's range $[\psi_b^l, \psi_b^u]$. To retrieve values for $\omega_E^{+}(s)$ and $\omega_E^{-}(s)$, we sequentially scan E's histogram bar list, find bars whose ranges are dominated by ψ_b^l and ψ_b^u, and then summarize their counts. The collected results are $\omega_E^{-}(s)$ and $\omega_E^{+}(s)$, respectively. The time complexity is $O(l)$.

We use $\omega_E^{+}(s)$ and $\omega_E^{-}(s)$ to estimate the dominance factor of s with respect to node E. We use $|s, E|_{min}$ and $|s, E|_{max}$ to estimate the spatial factor, as shown in Figure 4. The bounds in Lemmas 1 2 3 and 4 can thus be tightened. The improved ULBounds are as follows.

Ψ-Augmented ULBounds

Lemma 6. *Given object s and a node E, the Ψ-augmented upper bound of $\phi(s, E)$ is:*

$$\textit{Original Form: } \phi^{+}(s, E) = 2^{-\frac{|s,E|_{min}}{\epsilon}} \cdot \omega_E^{+}(s) \tag{21}$$

$$\textit{Expanded Form: } \phi^{+}(s, E) = \sum_{e \in E} 2^{-\frac{|s,e|_{min}}{\epsilon}} \cdot \omega_e^{+}(s) \tag{22}$$

Lemma 7. *Given object s and a node E, the Ψ-augmented lower bound of $\phi(s, E)$ is:*

$$\textit{Original Form: } \phi^{-}(s, E) = 2^{-\frac{|s,E|_{max}}{\epsilon}} \cdot \omega_E^{-}(s) \tag{23}$$

$$\textit{Expanded Form: } \phi^{-}(s, E) = \sum_{e \in E} 2^{-\frac{|s,e|_{max}}{\epsilon}} \cdot \omega_e^{-}(s) \tag{24}$$

The proofs of Lemma 6 and 7 are omitted due to page limits. To integrate them into Algorithm 2, we need to prove that they abide by the partial order property.

Partial Order Property for Ψ-Augmented ULBounds

Lemma 8. *The partial order property for Ψ-augmented upper/lower bounds holds.*

Proof. Let's consider the Ψ-augmented upper bound:

$$\text{(Expanded form)} \sum_{e \in E} 2^{-\frac{|s,e|_{min}}{\epsilon}} \cdot \omega_e^+(s) \leq \sum_{e \in E} 2^{-\frac{|s,E|_{min}}{\epsilon}} \cdot \omega_e^+(s)$$

$$= 2^{-\frac{|s,E|_{min}}{\epsilon}} \cdot \sum_{e \in E} \omega_e^+(s) = 2^{-\frac{|s,E|_{min}}{\epsilon}} \cdot \omega_E^+(s) \text{(Original Form)} \quad (25)$$

The partial order property for Ψ-augmented lower bound can be proved in a similar manner. Thus, the lemma is proved.

5 Experiments

Experimental setup is discussed in Section 5.1, where default parameters are bolded. Results are reported in Section 5.2.

5.1 Setup

Datasets. We use both synthetic datasets and real datasets for experiments. For all datasets, we normalize the spatial attributes to the Euclidean space $[0, 10000]^2$; and the non-spatial attributes to space $[0, 1]^d$.

For synthetic datasets, we generate a series of datasets, containing 50K, **100K**, and 200K following the way introduced in previous work [14]. We assume non-spatial attributes are independent with dimensions 3, **4**, and 5.

We also use a real dataset from AllStays.com[3] that maintains hotels, resorts, campgrounds, etc. around the world. We clean the dataset following the way introduced in [12]. Then, we obtain 30,918 hotel records with the schema (longitude, latitude, review, stars, price). We then normalized all 30,918 hotel records as described above, and call the normalized dataset **USH**.

Query. The parameter ϵ is set to 10, **50**, and 100. The parameter k is set to 50, **100**, and 200.

Methods. To evaluate kNDQ, we list several competitors: *Basic*, *kNDQ-SD*, *kNDQ**. Basic uses Algorithm 1. kNDQ-SD adopts Algorithm 2 with spatial dominance ULBounds (Section 3). kNDQ* adopts Algorithm 2 with Ψ-augmented ULBounds (Section 4).

Miscellaneous. The Ψ-augmented R-tree is implemented on R*-tree [13]. The entire tree is accommodated in the main memory. We set the tree fanout to be 25, according to the results reported elsewhere [7]. For dominance histogram, the bar width is set to be 0.5. If a dataset has 4 non-spatial attributes, each dominance histogram bar will have $\frac{25}{(1/0.5)^4} = 1.56$ objects on average. For higher dimensional data, objects will be more sparsely distributed among histogram bars. Thus, the bar width is reasonably small. All programs were implemented in C++ and run on a Core2 Duo 3.40GHz PC enabled by MS Windows 7 Enterprise.

[3] `http://www.allstays.com/`

5.2 Results

Synthetic Datasets. We test the query performances of Basic, kNDQ-SD, and kNDQ* by varying the size of database O in Figure 5. Compared to the fast increasing query time (T_q) of Basic, the time cost of kNDQ* increases in a relatively stable pace. Specially, when $|O| = 100K$, the time cost of kNDQ* is less than one quarter of that of Basic. The gap is enlarging while $|O|$ is increasing.

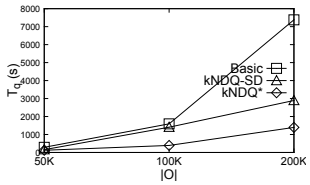


Fig. 5. T_q(s) vs. $|O|$

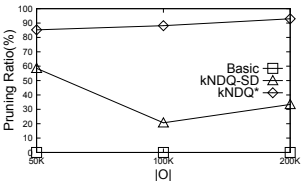


Fig. 6. Pruning Ratio vs. $|O|$

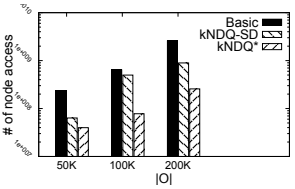


Fig. 7. T_q(# of node access) vs. $|O|$

To understand the superiority of kNDQ*, we study the effectiveness of upper/lower bounds. Referring to Figure 6, the pruning ratio of kNDQ* is stably above 88%, which is much higher than its counterparts. The superior pruning power is also reflected in Figure 7, where kNDQ* accesses one order of magnitude less tree nodes than others.

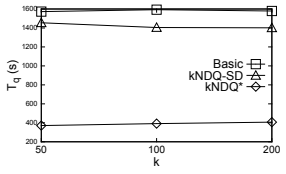


Fig. 8. T_q(s) vs. k

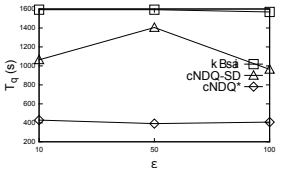


Fig. 9. $T_q(s)$ vs. ϵ

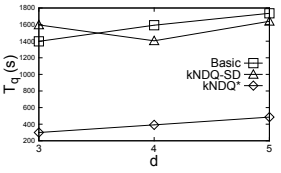


Fig. 10. T_q(s) vs. d

We further study the query performance by varying query parameter k and ϵ. With respect to k, all methods perform stably, while kNDQ* is the most efficient one, according to the results in Figure 8. For ϵ, both kNDQ* and Basic decrease slightly according to the results in Figure 9. A higher value of ϵ corresponds to a lower value of spatial factor of Equation 4. In other words, a higher ϵ implies that objects are harder to be found in "neighborhood" and thus have higher chances to be pruned.

We then study the impact of dimensionality d, and show the results in Figure 10. As d increases, the query time of kNDQ* increases stably. According to the result in Figure 11, the pruning ratio stays stable with respect to the increase of d.

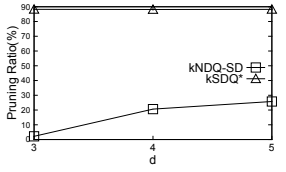


Fig. 11. Pruning Ratio vs. d

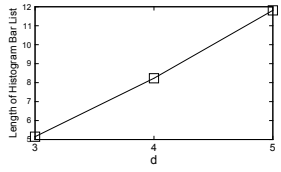


Fig. 12. Avg. Histogram Size per Node vs. d

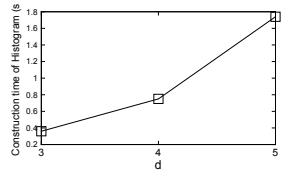


Fig. 13. Construction Time of Histogram vs. d

Also, we check the impact of d over the histogram size. From Figure 12, the average histogram size (per node) increases linearly with the dimensionality. Without histogram bar list, it would take $(\frac{1}{0.5})^5 = 32$ units to store the bars when $d = 5$. Thus, the bar list representation saves about 60% storage cost. The construction time increases super linearly as the results in Figure 13 indicate. Compared to the benefits in the query efficiency, the histogram construction is well worth the effort.

Real Datasets. Similar trends are observed from real datasets. The query time of kNDQ* increases stably with k (in Figure 14) and decreases with ϵ (in Figure 15). Compared to Basic and kNDQ-SD, kNDQ* achieves higher efficiency by accessing considerably less R-tree nodes, as shown in Figure 16.

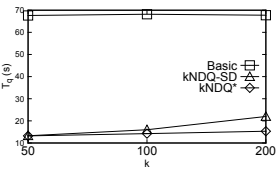


Fig. 14. T_q(s) vs. k (USH)

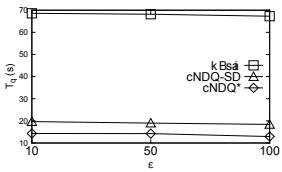


Fig. 15. T_q(s) vs. ϵ (USH)

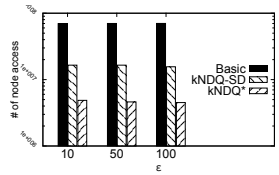


Fig. 16. T_q(# of node access) vs. ϵ (USH)

6 Related Work

Our work is related to previous works on dominance relationship and spatial ranking.

Dominance Relationship. Dominance relationship has been studied extensively in skyline queries. Borzonyi et al. [2] propose the external memory solution for skyline queries. Kossmann et al. [8] and Tan et al. [16] investigate indexes for efficient skyline query evaluation. Li et al. [9] use a data cube to facilitate dominance relationship analysis. Yiu et al. [19] study top-k dominating queries, which retrieves k objects dominating the highest number of objects.

Dominance Relationship in Spatial Databases. Sharifzadeh et al. [15] define a spatial dominance relationship based on spatial distances. They study on how to use Voronoi diagram to achieve higher efficiency. Huang et al. [5] propose a route skyline query which returns points of interests for query point moving on a pre-defined route. Continuous skyline queries for moving objects are studied in [6] [20]. However, all those works do not rank spatial objects. Lu et al. [12] rank objects by considering their distances to the nearest dominators.

Ranking Spatial Data. Du et al. [3] define the influence score of an spatial object as the weighted summation of its reverse nearest neighbors. Li et al. [10] study different queries involving both dominance relationship and spatial proximity. However, their approach handles the two aspects separately, and therefore is not applicable to ranking objects. Yiu et al. [17, 18] use an aggregated score function for non-spatial attributes and combine it with spatial proximity. Unlike this paper, their approach requires extra preferences to be specified to combine non-spatial attributes into an aggregated value.

7 Conclusion

In this paper, we study a novel type of query: top-k neighborhood dominating query (kNDQ). Adopting a flexible and general score function that takes into consideration both spatial locations and non-spatial quality attributes, kNDQ is able to capture different practical user needs in querying spatial objects. To support efficient query evaluation, we design an effective index structure, Ψ-augmented R-tree. With the augmented information at each index node, we efficiently estimate the ranking score by deriving effective upper/lower bounds without spending unnecessary cost on computing exact scores. Based on the proposed index, we design a branch and bound solution. Extensive experiments on synthetic and real datasets demonstrate the efficiency of our proposals.

References

1. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-tree: An efficient and robust access method for points and rectangles. In: SIGMOD (1990)
2. Borzonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proc. ICDE, pp. 421–430 (2001)
3. Du, Y., Zhang, D., Xia, T.: The optimal-location query. In: Medeiros, C.B., Egenhofer, M., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 163–180. Springer, Heidelberg (2005)
4. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD (1984)
5. Huang, X., Jensen, C.S.: In-route skyline querying for location-based services. In: Kwon, Y.-J., Bouju, A., Claramunt, C. (eds.) W2GIS 2004. LNCS, vol. 3428, pp. 120–135. Springer, Heidelberg (2005)
6. Huang, Z., Lu, H., Ooi, B.C., Tung, A.K.H.: Continuous skyline queries for moving objects. TKDE 18(12), 1645–1658 (2006)
7. Hwang, S., Kwon, K., Cha, S.K., Lee, B.S.: Performance evaluation of main-memory r-tree variants. In: Hadzilacos, T., Manolopoulos, Y., Roddick, J., Theodoridis, Y. (eds.) SSTD 2003. LNCS, vol. 2750, pp. 10–27. Springer, Heidelberg (2003)
8. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: Proc. VLDB, pp. 275–286 (2002)
9. Li, C., Ooi, B.C., Tung, A., Wang, S.: Dada: A data cube for dominant relationship analysis. In: SIGMOD (2006)
10. Li, C., Tung, A.K.H., Jin, W., Ester, M.: On dominating your neighborhood profitably. In: Proc. VLDB, pp. 818–829 (2007)
11. Lu, H., Yiu, M.L.: Identifying the Most Endangered Objects from Spatial Datasets. In: Winslett, M. (ed.) SSDBM 2009. LNCS, vol. 5566, pp. 608–626. Springer, Heidelberg (2009)
12. Lu, H., Yiu, M.L.: On computing farthest dominated locations. TKDE (2011)
13. Hadjieleftheriou, M.: Spatial index library version 0.44.2b
14. Borzonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE (2005)
15. Sharifzadeh, M., Shahabi, C.: The spatial skyline queries. In: Proc. VLDB, pp. 751–762 (2006)
16. Tan, K.-L., Eng, P.-K., Ooi, B.C.: Efficient progressive skyline computation. In: VLDB (2001)
17. Yiu, M.L., Dai, X., Mamoulis, N., Vaitis, M.: Top-k spatial preference queries. In: ICDE (2007)
18. Yiu, M.L., Lu, H., Mamoulis, N., Vaitis, M.: Ranking queries on spatial data with quality preferences. TKDE (2011)
19. Yiu, M.L., Mamoulis, N.: Efficient processing of top-k dominating queries on multi-dimensional data. In: VLDB (2007)
20. Zheng, B., Lee, K.C.K., Lee, W.-C.: Location-dependent skyline query. In: Proc. MDM, pp. 148–155 (2008)

OptRegion: Finding Optimal Region for Bichromatic Reverse Nearest Neighbors

Huaizhong Lin, Fangshu Chen, Yunjun Gao, and Dongming Lu

College of Computer Science and Technology, Zhejiang University, P.R. China
linhz@zju.edu.cn, youyou_chen@foxmail.com,
{gaoyj,ldm}@zju.edu.cn

Abstract. The MaxBRNN problem is to find an optimal region such that setting up a new service site within this region might attract the maximal number of customers by proximity. It has many practical applications such as service location planning and emergency schedule. Typical real-life applications are most in planar space, and the data volume of the problem is huge and thus an efficient solution is highly desired. In this paper, we propose an efficient algorithm, namely, OptRegion, to tackle the MaxBRNN problem in a two-dimensional space. Our methods employ three optimization techniques, i.e., sweep line, pruning strategy (based on upper bound estimation), and influence value computation (of candidate points), to improve the search performance. Extensive experimental evaluation using both real and synthetic datasets confirms that OptRegion outperforms the existing ones significantly under all problem instances.

1 Introduction

Given a database, an RNN (Reverse Nearest Neighbor) query returns the data points that have a given query point as their nearest neighbor. A BRNN (Bichromatic Reverse Nearest Neighbor) is the bichromatic version of RNN, in which all data points consist of the service point set $\mathcal{P}$ and the customer point set $\mathcal{O}$. For a service point $p \in \mathcal{P}$, a BRNN query finds all the points $o \in \mathcal{O}$ whose nearest neighbor in $\mathcal{P}$ is p. Those customer points o in $\mathcal{O}$ constitute the influence set of p and the influence value of p equals to the cardinality of the influence set. For example, in Fig.1, for a service point p_2, its BRNN, i.e., the influence set, is $\{o_2, o_3\}$.

The MaxBRNN problem [5,6] aims to find the region S in which all the points have the maximum influence value, namely the cardinality of BRNN set of all points p in S is maximized in a space. The MaxBRNN can be regarded as an optimal region search problem and has attracted much research efforts.

The MaxBRNN problem has many interesting real life applications in service location planning and emergency schedule. For example, in Fig. 1, there are five customer points o_1 to o_5 and four stores p_1 to p_4 in a city. Now a company wants to set up a new store and the objective is to find a location that can attract as many customers as possible under the assumption that the customers are more interested in visiting a convenient store based on their distances. We draw a circle for each customer point o_i ($1 \le i \le 5$), centered at o_i and the distance between o_i and its nearest store as radius.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 146–160, 2013.

The MaxBRNN problem is to find the optimal region, which is the intersection of three circles of o_2, o_3, and o_5, to set up a new store.

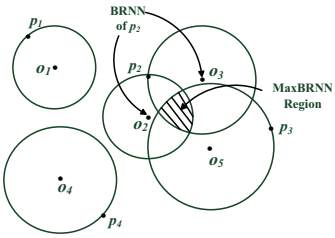


Fig. 1. An example of BRNN and MaxBRNN problem

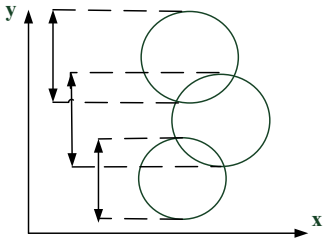


Fig. 2. The *y*-intervals of NLRs

There have been several algorithms [6,11,18,19] proposed to deal with the MaxBRNN problem in the literature. However, all these algorithms degrade significantly as the data-set becomes very large. Although the MaxBRNN problem has some applications in high dimensional spaces, most applications are in planar spaces, and the data volume of the problem is huge, and hence an efficient solution is highly desired.

In this paper, we propose an efficient algorithm called OptRegion to solve the MaxBRNN problem in a two-dimensional space. Our methods employ three optimization techniques, i.e., sweep line, pruning strategy (based on upper bound estimation), and influence value computation (of candidate points), to improve the search performance.

To sum up, our major contributions can be summarized as follows:

- We propose an efficient algorithm, namely, OptRegion, to the MaxBRNN problem, which can be applied for arbitrary L_p-norm space. Also, the sweep line technique is adopted in the algorithm to find overlapping circles quickly.
- We propose an effective pruning strategy, by which the majority of candidate points can be pruned without evaluation.
- We conduct extensive experiments with both real and synthetic data sets to demonstrate the performance of our proposed algorithm.

The rest of the paper is organized as follows. A survey of related work is given in section 2. Section 3 formulates the MaxBRNN problem. Section 4 describes the algorithm OptRegion in detail, and analyzes its efficiency. Section 5 evaluates the proposed algorithm and the pruning strategy through extensive experiments with real and synthetic data, and we conclude the paper in section 6.

2 Related Work

There are two types of RNN queries, namely, monochromatic and bichromatic RNN [9]. In the monochromatic case, all points are of the same category. In the bichromatic case, the points are of two different categories, such as services and customers. The original RNN problem has been studied in the literature [12,13,16] and the proposed algorithms are mainly based on some space-partition and pruning stratgies. In recent

years, the RNN problems have been studied extensively in the road network and the continuous environment[3,4,8,10,15].

The MaxBRNN problem, which maximizes the number of potential customers for the new service, was first introduced by Cabello et al. in [5,6], where they call it MAXCOV problem and present a solution for the two-dimensional Euclidean space. They also study other optimization criteria in BRNN queries: MINMAX, which minimize the maximum distance to the associated customers, and MAXMIN, which maximize the minimum distance to the associated customers.

The MaxBRNN is challenging in that there exists an infinite number of candidate locations where the new service may be built. The Algorithm MaxOverlap in [18] utilizes a technique called region-to-point transformation, which is also adopted in our OptRegion algorithm. It transforms the optimal region search problem to an optimal intersection point search problem in order to avoid searching an exponential number of regions. Nevertheless, the MaxOverlap does not scale well because the computation of optimal intersection points is expensive.

Wong et al. in [17] extend the MaxOverlap algorithm [18] for L_p-norm in the two and three-dimensional space. To the best of our knowledge, there is no previous work that study MaxBRNN query in more than 3 dimensional spaces.

Algorithm MaxSegment in [11] tries to speed up finding the optimal intersection point by checking intersection arcs of circles. However, since they don't use any pruning strategy, there may be a lot of intersection arcs that need to be checked, which is very time-consuming.

Zhou et al in [19] present an efficient algorithm called MaxFirst to solve the MaxBRNN problem. The MaxFirst utilizes the branch-and-bound principle, and partitions the space into quadrants recursively and computes the upper and lower bounds of the size of a quadrant's BRNN. The algorithm then retrieves only in those quadrants that potentially contain an optimal region. Experimental results show that MaxFirst is much more efficient than the MaxOverlap. Nonetheless, in some situations, there may also be a lot of quadrants need to be processed.

3 Problem Statement

Suppose we have a set $\mathcal{P}$ of service points and a set $\mathcal{O}$ of customer points in a two-dimensional space D. Each point $o \in \mathcal{O}$ has a weight $w(o)$, which is used to represent the number of customers or importance.

For a point $o \in \mathcal{O}$, kNN(o,$\mathcal{P}$) represents the set of the top k points in $\mathcal{P}$ that are nearest to o. for a point s in D ($s \in \mathcal{P}$ or not), BRkNN(s,$\mathcal{O}$,$\mathcal{P} \cup \{s\}$) represents the set of points in $\mathcal{O}$ that take s as one of their k nearest neighbors in $\mathcal{P} \cup \{s\}$. For simplicity, we take k=1 in the following discussion, which can be easily extended to the case k>1.

Definition 1. (influence set/value) given a point s, we define the influence set of s to be BRNN(s,$\mathcal{O}$,$\mathcal{P} \cup \{s\}$). the influence value of s is equal to $\sum_{o \in \text{BRNN}(s,\mathcal{O},\mathcal{P} \cup \{s\})} w(o)$.

Definition 2. (Nearest Location Region, NLR) Given a customer point o, the nearest location region R of o is defined to be the region centered at o and containing all the point s with $d(o,s) \leq d(o,\text{NN}(o,\mathcal{P}))$. NN($o$,$\mathcal{P}$) is the nearest neighbor of o in $\mathcal{P}$ and $d(x,y)$ is the distance between points x and y. The weight of R equals to the weight of o.

In the definition 2, we adopt the notation NLR to capture the general case when considering L_p-norm space. Minkowski distance can be adopted in computing $d(x,y)$.

When considering euclidean space, i.e., the L_2-norm space, the NLR is a circle around o with radius $d(o,\mathrm{NN}(o,\mathcal{P}))$.

Definition 3. (MaxBRNN) Given a set $\mathcal{P}$ of service points and a set $\mathcal{O}$ of customer points in a two-dimensional space $\mathbb{D}$. we want to find an optimal region S such that all points in S have the maximum influence value.

For example, in Fig. 1, the optimal region S is the intersection of three NLRs o_2, o_3, and o_5, and the influence value of S is the sum of the three NLRs' weights. Informally speaking, the MaxBRNN returns the region with maximum overlapped NLRs.

4 Algorithms for OptRegion

We also adopt the region-to-point transformation [18] to transform the optimal region search problem into finding the maximum influence intersection point between any two NLRs. The maximum influence intersection point can subsequently be mapped into the optimal region. Let S be the optimal region returned by the MaxBRNN query. If S is the intersection of more than one NLR, then there must exist two NLRs and at least one of their intersection points is contained in S.

OptRegion consists of three steps. First, NLRs are constructed for every customer point. Then, all intersection NLRs are detected, and the upper bound of influence values for all NLRs are estimated. Last, the exact influence values of candidate points are computed and the point with maximum influence value is found.

We first describe several techniques used in OptRegion in section 4.1 and 4.2, then give the overall algorithm in section 4.3. We omit the detailed description of our influence value computation technique here, since it is similar to the algorithm MaxSegment in [11] despite some differences in details and expressions.

4.1 Sweep Line

The method to find intersection NLRs in algorithm MaxOverlap [17,18] and MaxFirst [19] is as follows. First, an R*-tree is built for all NLRs. Then a range query against the R*-tree for each NLR is performed to find its all intersection NLRs. But it will take a lot of time to construct R*-tree and perform range query, especially when there is a large amount of customer points.

We adopt the sweep line approach to find all intersection NLRs, which localize the search range of intersection NLRs to speed up the processing. Sweep line approach is widely adopted in a variety of computational geometry problems, such as line segment intersection, Voronoi diagram construction etc [2].

In sweep line and subsequent prune strategy techniques, we model an NLR by its minimum bounding rectangle (MBR), which means the result is a kind of upper bound estimation. The exact result can be gained in the influence value computation process.

We define the *y*-interval of an NLR to be its orthogonal projection onto the *y*-axis, as shown in fig. 2. When the *y*-intervals of a pair of NLRs do not overlap, then they cannot intersect. Hence, only the pairs of NLRs whose *y*-intervals are overlapping need to be tested for intersection. It is obvious that there exists a horizontal line that intersects both NLRs whose *y*-intervals are overlapping. So, to find these pairs we imagine sweeping a line downwards over the plane, starting from a position above all NLRs. While we sweep the imaginary line, we keep track of all NLRs intersecting with it, so that we can find the intersection pairs we need.

The line is called the sweep line and the status of the sweep line is the set of NLRs intersecting with it, as shown in Fig. 3. We need a status structure to maintain the status of the sweep line. This is an ordered sequence of NLRs intersecting with the sweep line. All NLRs in the status structure are ordered by their *x*-coordinates of the left-most points. The status structure must be dynamic: as NLRs start or stop to intersect with the sweep line, they must be inserted into or deleted from the structure on the fly. We use the balanced binary search tree [14] in status structure implementation. The status structure changes while the sweep line moves downwards, but not continuously. Only at particular points, called event points, the update is required. Here the event points are the top and bottom point of NLRs, as shown in Fig. 3.

Lemma 1. To find intersection NLRs correctly, the status structure need to be updated only when the sweep line reaches the event points.

Proof. Considering an NLR *R*, there are two cases: 1) When the sweep line reaches an event point which is the top point of an NLR *R*, NLR *R* starts intersecting with the sweep line. Before *R* is inserted into the status structure, it is tested for intersection against the NLRs in status structure, which finds intersection NLRs that have been swept up to now. Other intersection NLRs with *R* below sweep line will be found and added to the intersection NLR list of *R* subsequently while the sweep line goes down; 2) When the sweep line reaches the bottom event point of *R*, it can be removed from the status structure safely, for all intersection NLRs with *R* have been found and *R* no longer intersects with the sweep line. □

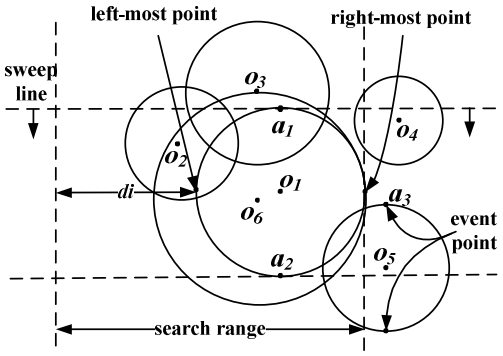


Fig. 3. An example of sweep line

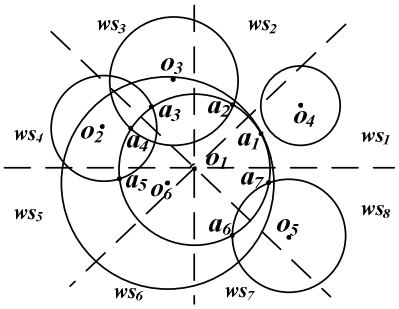


Fig. 4. An example of OptRegion

We maintain an event queue to store all the event points, which are the top and bottom points of all NLRs. The event points are ordered by their *y*-coordinates downwards. If two event points have the same *y*-coordinate, the process order can be arbitrary and the result is the same.

Lemma 2. Let *xleft* and *xright* be the *x*-coordinate of the left-most and right-most point of an NLR *R*, and *di* be the maximum diameter of NLRs swept up to now (as shown in Fig. 3). The NLRs whose left-most points are out of the range [*xleft*−*di, xright*] will not intersect with *R*.

Proof. We prove by contradiction. Suppose that an NLR R_i intersects with *R* and the *x*-coordinate x_a of the left-most point is out of the range [*xleft*−*di, xright*]. There are two cases:

1) If x_a<*xleft*−*di*: let *u* be the diameter of R_i. Since R_i intersects with *R* and the left-most point of R_i lies to the left of *R*, it can be argued that the *x*-coordinate of R_i's right-most point must be no less than that of *R*'s left-most point, namely, *xleft*≤x_a+*u*. Combined with x_a<*xleft*−*di,* we can get *di*<*u*, which contradicts with the fact that *di* is the maximum diameter of NLRs swept up to now.
2) If x_a>*xright*: it obviously contradicts with the assumption that R_i intersects with *R*. □

Based on lemma 2, we can perform a range query of [*xleft*−*di*, *xright*] in the status structure to find all NLRs intersecting with NLR *R* at the moment *R* is inserted into the status structure. This range query finds those intersection NLRs that have been swept up to now. Other intersection NLRs with *R* below sweep line will be found and added to the intersection NLR list of *R* subsequently while the sweep line goes down. By this way we localize the search of intersection NLRs by only testing pairs of NLRs for which there is a horizontal line that intersects with both NLRs. When the sweep line reaches the bottom event point of *R*, then all intersection NLRs with *R* have been found and *R* can be deleted from the status structure safely.

For example, in Fig.3, R_i is the NLR of customer point o_i. When the sweep line reaches point a_1 (the top event point of NLR R_1), we can get the intersection NLR list of R_1 at this moment is I_{R1} ={R_2, R_3, R_4, R_6}, for their left-most points locate in R_1's search range. At the same time, R_1 is added to the intersection NLR list of R_2, R_3, and R_4. After R_1 is inserted into the status structure, the sweep line moves down and reaches point a_3, the set I_{R1} is updated to be I'_{R1}={R_2, R_3, R_4, R_5, R_6}. Finally, when the sweep line reaches a_2, we delete R_1 from the status structure and the resulting intersection NLR list of R_1 is I'_{R1}. Here, we have to mention that the resulting I'_{R1} is a superset of the exact intersection NLR list, e.g., R_4 in I'_{R1} is actually not intersecting with R_1. The exact result can be gained in the subsequent influence value computation process.

Based on the above discussion, the sweep line algorithm is given below. We construct an event queue in line 1. The sweeping process is described in line 3-9. When a top point of NLR *R* is encountered, it is tested for intersection NLRs and inserted into the status structure. When a bottom point is encountered, *R* is deleted from the status structure. By the end of the algorithm, the construction of intersection NLR lists for all NLRs are accomplished.

Algorithm 1. SweepLine

input : $\mathcal{O}$:= set of NLRs of all customers
output: $\boldsymbol{I}$:= intersection NLR lists for every NLR

```
1   construct the event queue
2   initialize an empty status structure
3   for each event point in the event queue by downward y-coordinate order
4       let R be the NLR of the event point
5       if the event point is the top point of R
6           find the intersection NLRs in the status structure and record in both
            intersection list of R and intersection NLRs
7           insert R into the status structure
8       else //the bottom point of R
9           delete R from the status structure
10  return I
```

4.2 Pruning Strategy

We divide an NLR into eight subspaces evenly, illustrated as NLR R_1 in Fig. 4, by four dotted lines, horizon, vertical, and two lines intersect with horizon line by 45° angle. For each subspace, we sum the weight of the NLRs intersect with the current NLR and locate in that subspace (including the NLR itself), noted as ws_1 to ws_8. Although the partitioning scheme with eight subspaces is described here, other partitioning scheme such as no partitioning, 4 or 16 partitioning can also be used. The selection of partitioning scheme is based on the balance between pre-computing cost and pruning power. With more partitioning subspaces, the pruning power is better but we need much more precomputing cost.

Lemma 3. The weight sum ws_i ($1 \le i \le 8$) is the upper bound of the maximum influence value of the intersection points in that subspace. $wsMax = \max\{ws_i, 1 \le i \le 8\}$ is the upper bound of maximum influence value of the intersection points in an NLR.

Lemma 4. Suppose *max* is the maximum influence value found so far, then any NLRs with *wsMax* less than *max* can be safely pruned without further consideration.

Proof. During the computation of the *wsMax* value, we conduct upper bound estimation of the value, namely, it is a little greater than or equal to the actual maximum influence value of the intersection points in an NLR. There are two cases of upper bound estimation:

1) Every NLR is modeled by its MBR, which will certainly lead to overestimation. For example, in Fig.5(a), although the MBR of R_1 and R_2 intersect with each other, the circles don't intersect actually. Therefore, ws_1 and ws_2 of R_1 is greater than the actual influence value of intersection points in this subspace. For R_3, it doesn't locate in the third subspace, but it is counted in the computation of ws_3, which makes ws_3 a little greater, for point *a* locates in the subspace.

2) When we compute the ws_i value, we sum the weight of the NLRs that intersect with the current NLR and locate in the subspace. However, in most cases, the NLRs located in the same subspace may not all contain a same point, in other word, ws_i is

the upper bound estimation of the maximum influence value of the intersection points in that subspace. For example, in Fig.5(b), NLR R_2, R_3, and R_4 all intersect with R_1 and locate in the third subspace, so ws_3 is 4 (including R_1 itself). However, because R_2 doesn't intersect with R_3 and R_4, the actual optimal intersection points are a_3 or a_4 whose maximum influence value is 3, which is smaller than ws_3.

Thus, any NLRs with *wsMax* less than *max* will certainly contain no intersection point with influence value greater than *max* and the NLRs can be pruned safely. □

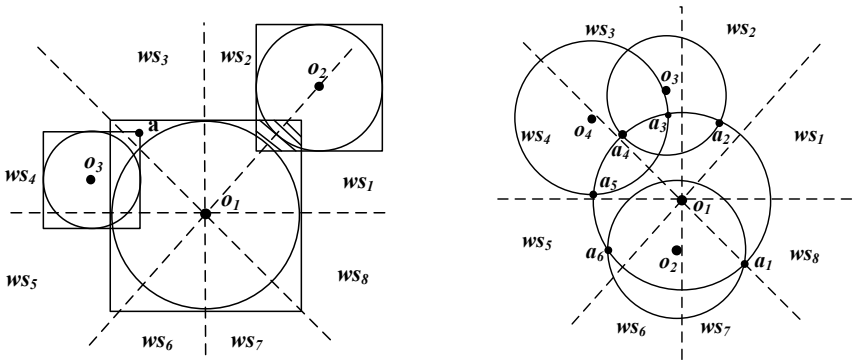


Fig. 5. Upper bound estimation

While performing upper bound estimation, the intersection points of NLRs are not necessary to be computed. We only need to decide in which subspaces an intersection NLR may be located with respect to the current NLR by comparing of the customer points' coordinate values, avoiding the time-consuming computation of intersection points and corresponding influence values [18].

Based on lemma 3, after we have computed the *wsMax* value of all NLRs, NLRs are processed by the descending *wsMax* order. The NLRs with greater *wsMax* have higher probability containing the optimal intersection point and there are more chances to prune NLRs.

4.3 OptRegion

Definition 4. (Intersection Arc) If NLR R intersects with NLR R_i, then the arc in the perimeter of R within the intersection region is called intersection arc in R. The weight of the intersection arc is equal to the weight of R_i, i.e. $w(o_i)$.

We illustrate briefly the influence value computation technique and the algorithm OptRegion by an example in Fig.4. After the sweep line process, we find that R_1 intersects with R_2 to R_6, and we compute that ws_1=3, ws_2=4, ws_3=4, ws_4=4, ws_5=3, ws_6=2, ws_7=3, ws_8=3, so the *wsMax* value of R_1 is 4. Similarly, the *wsMax* of R_2 to R_6 are: 4,4,3,3,4. So, we will first deal with NLR R_1 with the *wsMax* value 4.

NLR R_1 intersects with R_6 by arc a_1a_1 in R_1 (it is the whole circle of R_1), with R_3 by arc a_2a_4, with R_2 by a_3a_5, and with R_5 by a_6a_7. To find the intersection point with maximum influence value among a_1 to a_7, we only need compute the maximum overlap weight of the arc a_1a_1, a_2a_4, a_3a_5, a_6a_7, and plus the weight of R_1. Suppose the weight

of each o_i is equal to 1. By sweeping around the circle of R_1 once, we can easily find the points a_3 and a_4 have the maximum influence value 4, which is corresponding to the optimal region overlapped by R_1, R_2, R_3,and R_6. Having found the intersection point with influence value 4, namely, *max*=4, we can prune all the other NLRs without further influence value computation since there is no NLR with the *wsMax* value larger than *max*.

We give our algorithm OptRegion below. All NLRs are constructed in line1. We build a kd-tree for all service points in $\mathcal{P}$. Then we use the algorithm ANN in [1] to find the nearest service point over the kd-tree for each customer point in $\mathcal{O}$.

In line 2-3, algorithm OptRegion calls algorithm SweepLine to find all intersecting NLRs and construct an intersection NLR list for each NLR, then we estimate the upper bound of maximum influence value of the intersection points in an NLR.Variable *s* and *max* are global variables representing the intersection point with the maximum influence value found so far and its influence value. Line 5 to 7 initialize *s* and *max* to be the point and weight of the single NLR with largest weight, with the purpose to correctly deal with the case where the optimal region is contained in only one NLR. In the above example, *s* is set to be R_1 and *max* to be 1.

In line 8-11, NLRs are processed by descending *wsMax* order, in an attempt to prune more NLRs that the estimated maximum influence value are less than the maximum influence value found so far.

In line 11, the algorithm tries to find the exact maximum influence value of all intersection points in R by sweeping around the perimeter of R counterclockwise. Based on the intersection NLR list of NLR R, it is easy to compute the endpoints of intersection arcs for all intersection NLRs. By this way, we map all the customer points that contribute to the influence value of the candidate points to circular arcs in the perimeter of NLR. Then, we sweep around the circle of R to compute the influence value of each intersection point. The details of the process can refer to [11].

Algorithm 2. OptRegion

input : $\mathcal{O}$:= set of customer points
$\mathcal{P}$:= set of service points
output: $\boldsymbol{S}$:= optimal region presented by the overlapping NLRs

```
1   for each o ∈ O construct an NLR for o
2   call SweepLine
3   compute the wsMax for all NLRs
4   sort all the NLRs by the wsMax value
5   choose the NLR R with the largest weight
6   s←any point in R
7   max←w(R)
8   for every R in the NLR set by descending wsMax order
9     if the wsMax of R is less or equals to max
10      break
11    influence value computation for all intersection points in R, if the
      computed value is greater than max, the max is replaced and s is replaced
      with the intersection point for the corresponding max
12  find the intersection of all NLRs containing s, put the id of these NLRs into S
13  return S
```

4.4 Analysis

Time Complexity: The algorithm OptRegion has the following three steps.

Step 1 (NLR construction): We use kd-tree [7] to perform nearest neighbor query. We build a kd-tree for all service points in $\mathcal{P}$, which requires $O(|\mathcal{P}|\log|\mathcal{P}|)$ time and $O(|\mathcal{P}|)$ space. Then we use the algorithm ANN in [1] to find the nearest service point over the kd-tree, which requires $O(\log|\mathcal{P}|)$ time for each customer point in $\mathcal{O}$. Thus, the construction of NLRs can be done in $O(|\mathcal{P}|\log|\mathcal{P}|+|\mathcal{O}|\log|\mathcal{P}|)$.

Step 2 (Find intersection NLRs and estimate upper bound of influence value): Let d be the maximum number of NLRs whose MBR intersects with a given NLR's MBR, and c be the maximum NLRs intersecting with a sweep line. In the sweep line algorithm, it takes $O(|\mathcal{O}|\log|\mathcal{O}|)$ time to sort the event queue. For each NLR, it takes $O(d+\log c)$ time to test intersection, and $O(\log c)$ time to insert into and delete from the status structure. So, the overall execution time for sweep line algorithm is $O(|\mathcal{O}|(d+\log|\mathcal{O}|))$. For each NLR, its MBR intersects with at most d MBRs, thus it takes $O(d)$ time to process each NLR, i.e., $O(|\mathcal{O}|d)$ time to estimate upper bound of influence value for all NLRs. Thus, this step requires $O(|\mathcal{O}|(d+\log|\mathcal{O}|))$ time.

Step 3 (Find the optimal intersection point): first we need to sort all NLRs by descending *wsMax* order, which requires $O(|\mathcal{O}|\log|\mathcal{O}|)$ time. Actually, we need not fully sort all NLRs, here we only need a priority queue that can return the NLR with next maximum *wsMax* value. As soon as the *wsMax* value returned is below the maximum influence value found so far, remained NLRs are all pruned without further processing. Let α ($0\leq\alpha<1$) be the prune rate. There are at most d intersection arcs in each NLR, the influence value computation takes $O(d)$ time for each NLR. Thus, this step requires $O((1-\alpha)|\mathcal{O}|d)$ time.

From the analysis above, we can get the following theorem.

Theorem 1. The overall time complexity of our algorithm is $O(|\mathcal{P}|\log|\mathcal{P}|+|\mathcal{O}|(\log|\mathcal{P}|+d+\log|\mathcal{O}|))$.

Correctness: Intuitively, the correctness of OptRegion is guaranteed by the following three aspects: 1) Based on Lemma 1 and 2, the algorithm sweep line can find all the intersection pairs of NLRs as the sweep line goes down; 2) Based on Lemma 4, the pruning strategy prune the NLRs correctly; 3)The influence value computation process can compute the influence value of intersection points exactly.

The correctness of OptRegion can be concluded as theorem 2. We omit the detailed proof here to save space.

Theorem 2. Algorithm OptRegion returns the MaxBRNN region correctly.

5 Performance Study

We have conducted extensive experiments to evaluate the performance of our algorithm OptRegion. We compare our algorithm with the state-of-the-art algorithm MaxOverlap[18], MaxSegment[11], and MaxFirst[19] for the MaxBRNN problem. All experiments show that our algorithm outperforms other algorithms significantly.

The algorithm OptRegion is implemented in C++, in which we reused part of the C++ code of MaxOverlap. The C++ code of MaxOverlap is got from the authors. We implement the algorithm MaxSegment and MaxFirst in C++ according to the description in their papers. All experiments are carried out on a Linux machine with an Intel Core2 Duo 2.9 GHz CPU and 2GB memory.

The performance evaluation is performed using both synthetic and real datasets. The synthetic datasets follow uniform and Gaussian distributions, and the number of customer points ranges from 50K to 200K. The customer dataset and the service dataset have the same distribution. Two real world datasets LB and CA, which contain 2D points representing geometric locations in Long Beach Country and California respectively, are also used in the experiments. The LB and CA datasets are available at http://www.rtreeportal.org/spatial.html. For real datasets, the number of customer point is in the range from 10K to 40K. We partition the real datasets into two parts, the first 40K points are customers and the remaining 20K points are services. We set the cardinality of $\mathcal{P}$ (service set) to be half the cardinality of $\mathcal{O}$ (customer set) for both synthetic and real datasets, since in reality the number of services is always much fewer than that of customers.

The weight of each customer point in both synthetic and real datasets is set to 1. When the weights of customer points are other than 1, the experimental results are similar and we omit it here.

We first conduct experiments to evaluate the pruning strategy with different partitioning methods in section 5.1. In section 5.2, to gain insight of the performance promotion of our algorithm, we compare algorithms from two aspects of running time: 1) overall running time, which includes the execution time for all three steps in section 4.4; 2) running time without preprocessing, which includes the time of finding intersection NLRs, estimating upper bound of influence value, and finding the optimal intersection point, i.e. the step 2 and step3 in section 4.5.

5.1 Effectiveness of Pruning Strategies

The upper bound estimation of influence value in OptRegion is based on the space partitioning of an NLR. Different partitioning will have different pruning effect. We explore three strategies here, i.e. 1-partition, 4-partition and 8-partition, in which an NLR is partitioned into 1, 4, and 8 subspaces respectively. We depict the pruning effect and running time of different strategies in Fig. 6 and Table 1 respectively, which show that 4 and 8-partition are superior to 1-partition. The experiments are conducted on the synthetic dataset with 80K customer points. Compared to 1-partition, both 4 and 8-partition have more powerful pruning effect. We have to compute exact influence value of more than 1300 NLRs against all 80K NLRs when 1-partition is used, whereas only 253 and 140 NLRs really need exact influence value computation when 4 and 8-partition are used respectively. The prune rate α is 99.83% for 8-partition.

From Table 1 we can find that, as the increase of subspace number, the time percentage of exact influence value computation decreases, but the time percentage of upper bound estimation increases. There are almost no differences between the running time of 4 and 8-partition. Although 8-partition strategy can prunes more NLRs, the pruning process itself need more computation. Except the 1-partition strategy, we can choose either of the other two pruning strategies.

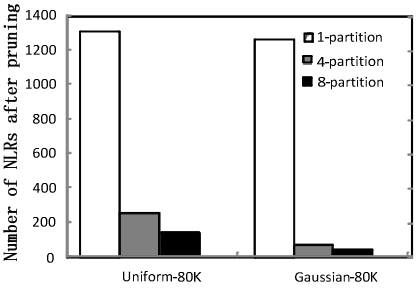


Fig. 6. Pruning effect of different strategies

Table 1. Running time of different pruning strategies

		1-partition	4-partition	8-partition
uniform 80K	number of NLRs not pruned	**1302**	**253**	**140**
	time percentage of upper bound estimation	**70.04%**	**78.20%**	**82.76%**
	time percentage of influence compute	16.30%	4.71%	1.31%
	total time (without preprocessing)	1.67s	1.38s	1.45s
Gaussian 80K	number of NLRs not pruned	**1261**	**69**	**40**
	time percentage of upper bound estimation	**76.60%**	**82.10%**	**82.77%**
	time percentage of influence compute	7.90%	0.45%	0.38%
	total time (without preprocessing)	1.58s	1.3s	1.29s

5.2 Results on MaxBRNN

Overall Running Time: The experiment results are given in Fig. 7 for synthetic and real datasets with different point set cardinalities. Our algorithm OptRegion outperforms MaxOverlap, MaxSegment, and MaxFirst significantly for all circumstances. The performance promotion is up to about one order of magnitude. As the number of point set increases, the overall running time of the other algorithms increase dramatically, whereas OptRegion has a relatively much smaller increase rate, showing its good scalability.

There are four major advantages in OptRegion that contribute to the remarkable performance improvement:

1) We use kd-tree instead of R^*-tree to store the service points and perform nearest neighbor query. The preprocessing time of other algorithms are almost the same and increase dramatically with the increase of point number. They all adopt the same R^*-tree index structure. Although the time complexity by using kd-tree is the same as R^*-tree, we find that the kd-tree is much more efficient for nearest neighbor query than R^*-tree through experiments. The kd-tree is overlap free and the partition is even, which make the algorithm need not to search useless branches. We adopt an existing *k* nearest neighbors query library (http://www/cs/umdedu/~mount/ANN/) in our implementation. The library is an implementation of the algorithm ANN in [1];

2) We adopt sweep line technique to find intersection NLRs, which localizes the search range and has much more simple computation;

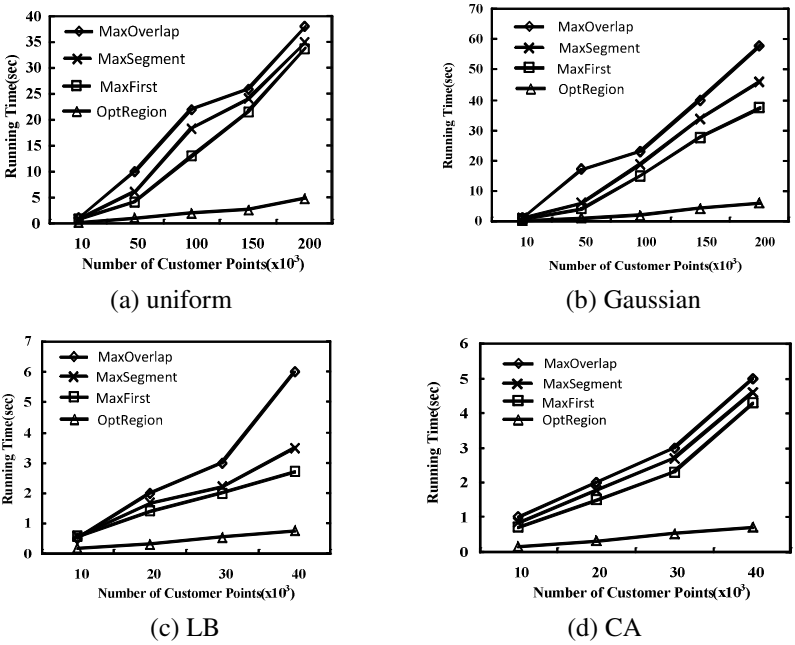


Fig. 7. Overall running time

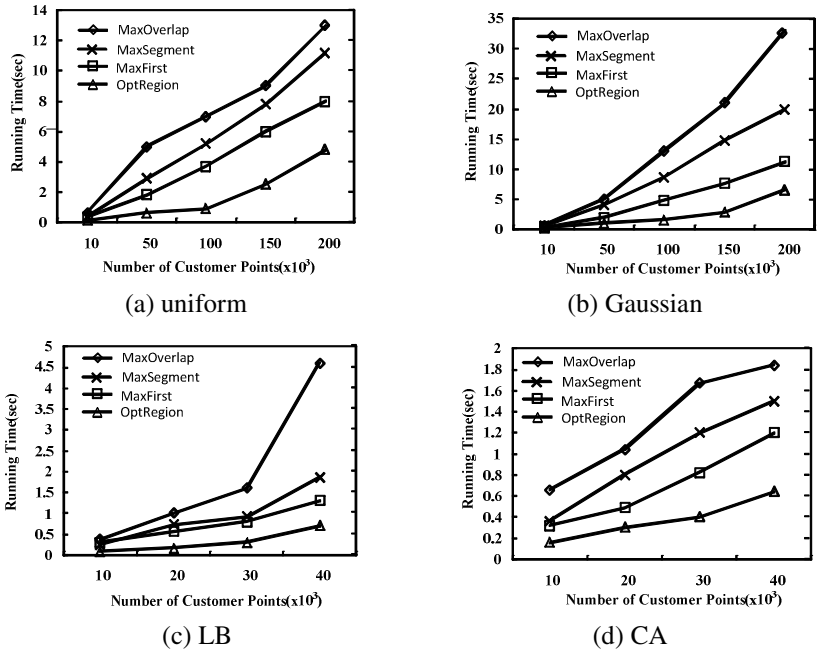


Fig. 8. Running time without preprocessing

3) We adopt a pruning strategy, which is based on the upper bound estimation of the influence value for candidate points. A large amount of NLRs can be pruned without further processing;

4) We adopt a novel approach to fast compute the exact influence value of candidate intersection points, which need much less computation cost than other algorithms.

Running Time without Preprocessing: The experiment results are given in Fig.8. Without the preprocessing time, OptRegion is about one time faster than MaxFirst and several times faster than MaxOverlap and MaxSegment.

Comparing Fig. 8(a) and 8(b), we observe that data distribution affects the algorithms performance. All algorithms spend more time on datasets with Gaussian distribution than uniform distribution, for there are much more intersection NLRs and intersection points in the dense area of Gaussian distribution dataset.

From Fig. 8(c) and 8(d), we find that the result on real world datasets is consistent with the synthetic datasets. For the dataset CA, OptRegion is about 50% improvement over MaxFirst and about 75% and 80% improvement over MaxSegment and MaxOverlap. For dataset LB, the result is almost the same except that MaxOverlap degrades dramatically when the cardinality is 40K.

6 Conclusion and Future Work

In the paper, we propose an efficient algorithm called OptRegion to address the MaxBRNN problem, which has many applications in real life. OptRegion employs several efficient techniques, such as sweep line, upper bound estimation, and fast influence value computation. Extensive experiments using both real and synthetic data sets verify the effectiveness and efficiency of our proposed OptRegion algorithm. The experimental results show that OptRegion is one order of magnitude faster than the-state-of-the-art competitors in all cases.

There are several directions in our future work. First, we will explore the MaxBRNN problem in the high dimensional space and the road network. Second, the current MaxBRNN problem assumes that every service point has infinite capability for customers. However, a service point (for example, a restaurant) can only accommodate a certain number of customers in reality. It is necessary to consider the problem under the limited capability assumption.

Acknowledgments. This work was supported in part by NSFC grant 61003049, the Fundamental Research Funds for the Central Universities grant 2012QNA5018, the Zhejiang Province Funds for Science and Technology Innovation Team grant 2010R50040, the Public Technology Research Program of Zhejiang Province grant 2010C33151, the cultural relic protection science and technology project of Zhejiang Province, and the Key Project of Zhejiang University Excellent Young Teacher Fund (Zijin Plan).

References

[1] Arya, S., Mount, D.-M., Netanyahu, N.-S., Silverman, R., Wu, A.-Y.: An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. JACM 45(6), 891–923 (1998)

[2] Berg, M., Cheong, O., Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications, 3rd edn., pp. 32–55. Springer (2009)

[3] Bernecker, T., Emrich, T., Kriegel, H.-P., Renz, M., Zankl, S., Züfle, A.: Efficient probabilistic reverse nearest neighbor query processing on uncertain data. In: VLDB, pp. 669–680 (2011)

[4] Cheema, M.-A., Zhang, W., Lin, X., Zhang, Y., Li, X.: Continuous reverse k nearest neighbors queries in Euclidean space and in spatial networks. In: VLDB, vol. 21(1), pp. 69–95 (2012)

[5] Cabello, S., Díaz-Báñez, J.M., Langerman, S., Seara, C., Ventura, I.: Reverse facility location problems. In: CCCG (2005)

[6] Cabello, S., Díaz-Báñez, J.M., Langerman, S., Seara, C., Ventura, I.: Facility location problems in the plane based on reverse nearest neighbor queries. Eur. J. Oper. Res. 202(1), 99–106 (2010)

[7] Friedman, J.-H., Bentley, J.-L., Finkel, R.-A.: An algorithm for finding best matches in logarithmic expected time. ACM TOMS, 209–226 (1977)

[8] Gao, Y., Zheng, B.H., Chen, G., Li, Q., Guo, X.F.: Continuous visible nearest neighbor query processing in spatial databases. In: VLDB, vol. 20(3), pp. 371–396 (2011)

[9] Korn, F., Ukrishnan, S.-M.: Influence sets based on reverse nearest neighbor queries. In: SIGMOD, pp. 201–212 (2000)

[10] Kang, J.-M., Mokbel, M.F., Shekhar, S., Xia, T., Zhang, D.H.: Continuou evaluation of monochromatic and bichromatic reverse nearest neighbors. In: ICDE, pp. 781–790 (2007)

[11] Liu, Y.-B., Wong, R.-C., Wang, K., Li, Z.-J., Chen, C.: A New Approach for Maximizing Bichromatic Reverse Nearest Neighbor Search. In: KAIS (to appear)

[12] Singh, A., Ferhatosmanoglu, H., Tosun, A.: High Dimensional Reverse Nearest Neighbor Queries. In: CIKM, pp. 91–98 (2003)

[13] Stanoi, I., Agrawald, D.: Reverse nearest neighbor queries for dynamic databases. In: ACM SIGMOD DMKD, pp. 44–53 (2000)

[14] Sedgewick, R., Brown, M.H.: Data Structures and Algorithems, 1st edn. Addison-Wesley, Balanced Trees (1983)

[15] Shang, S., Yuan, B., Deng, K., Xie, K., Zhou, X.: Find the most accessible locations: reverse path nearest neighbor query in road networks. In: ACM GIS, pp. 181–190 (2011)

[16] Tao, Y., Papadias, D., Lian, X.: Reverse KNN search in arbitrary dimensionality. In: VLDB, pp. 744–755 (2004)

[17] Wong, R.-C., Tamer Özsu, M., Fu, A.-W., Yu, P.-S., Liu, L., Liu, Y.: Maximizing bichromatic reverse nearest neighbor for Lp-norm in two- and three-dimensional space. In: VLDB, pp. 893–919 (2011)

[18] Wong, R.-C., Özsu, M.T., Yu, P.-S., Fu, A.-W., Liu, L.: Efficient method for maximizing bichromatic reverse nearest neighbor. In: VLDB, pp. 1126–1137 (2009)

[19] Zhou, Z., Wu, W., Li, X., Lee, M.-L.: Wynne Hsu: MaxFirst for MaxBRkNN. In: ICDE, pp. 828–839 (2011)

Generalization-Based Private Indexes for Outsourced Databases

Yi Tang[1,2], Fang Liu[1,2], and Liqing Huang[1,2]

[1] School of Mathematics and Information Science
Guangzhou University, Guangzhou 510006, China
[2] Key Laboratory of Mathematics and Interdisciplinary Sciences of Guangdong Higher Education Institutes
Guangzhou University, Guangzhou 510006, China

Abstract. Ensuring data protection and enhancing selective query performance over encrypted data are two closely linked challenges for outsourced databases. It needs to develop indexes over encrypted data to support secure and efficient selective queries on server side. However, the plaintext-associated information hidden in those indexes may introduce inference attacks when comparing with different encrypted tuple sets. In this paper, we investigate a kind of inference attacks by linking query results from different database users. The inferences are based on implicit equality relations hidden in query results. To defend against this attack, we develop a generalization-based method to construct secure and private indexes. We design a combined metric to measure the inference resistance of our proposed method. This measure is quantized by the entropy values and attribute value diversities in query results. We have conducted some experiments to validate our proposed method.

1 Introduction

As is common knowledge, storing data encrypted is a basic requirement for outsourcing databases. How to execute selective queries securely and efficiently over encrypted data becomes a main concern in database research community. To avoid storing too many different encrypted versions of a single tuple on servers, each tuple is generally encrypted with a single key by default in early efforts on outsourced databases [1][5]. This implies that a certain user may have the full rights to access any encrypted tuples if he has the decryption key. The data owners need to participate in the post-retrieval process to provide different views to different users.

The selective encryption methods use different keys to encrypt different data portions such as tuples or attributes [7]. To avoid users from managing too many keys, the keys can be derived from user hierarchy [2] and the traditional encryption method can be changed into some appropriate methods such as the attribute-based encryption (ABE) method [14]. Although these methods provide an effective and possible way to combine encryption with access control, the fulfillment of access control depends on the readability of decrypted data.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 161–175, 2013.

This means that some decryption efforts on client side are wasted. It needs purified indexes locate encrypted tuples as precisely as possible to reduce the extra decryption costs.

The granularity of data encryption in outsourced databases can be a tuple or an attribute. Constructing the indexes directly on the encrypted values is not an ideal choice because of the poor query efficiencies. It is generally to introduce auxiliary attributes to index encrypted values [5][9]. These auxiliary attributes reflect mapping images of plain attribute values. The preimage of each image can be a single value or a set of values. And then the constructed indexes can be called as value-based indexes or bucket-based indexes [11].

The introduced auxiliary attribute values is of benefit to locate encrypted tuples accurately. However, those values only shadow real the attribute values. They are inextricably linked to the original attribute values and may introduce potential disclosure of confidential information [1][11][12]. Securely combining the index and selective encryption is still immature.

In the scenario of outsourced databases, on one hand, the database service provider is not required to guarantee a strict separation among data portions available to different users. On the other hand, if index values on a certain attribute value cannot be distinguished on the tuples with different access control lists, the equality relations on the plain attribute values among those tuples will be demonstrated explicitly. This may lead an adversary user can infer attribute values in some tuples that he cannot access even if these tuples are encrypted [11].

The inference on the explicit equality of index tags can be eliminated by introducing different index values. We note that query results implicitly contain some equality relations among attribute values. Once the adversary user obtains query results for other users by colluding with the service provider, he can exploit the implicit equality to infer the attribute values he cannot access.

In this paper, we will address the issue of protection against this kind of collusion-based inferences by mitigating the implicit equality relations among query results. We try to close the gap between indexing for efficient querying and selective encryption for access controls. We propose a generalization-based method to construct private indexes to defend against collusion-based inferences. We define a combined metric, quantized by the entropy values and attribute value diversities, to measure the degree of privacy protected among those query results.

The rest of this paper is structured as follows. In Section 2, we give our basic models and introduce the collusion-based inference attack over encrypted data. In Section 3, we explore the reason why the collusion-based inference attack can be achieved. In Section 4, we introduce the notion of generalization and propose the notion of (k, α)-secure index to develop a generalization-based index. In Section 5, the method to construct the (k, α)-secure index is proposed. In Section 6, we conduct some experiments to validate our proposed method. And finally, the conclusion is drawn in Section 7.

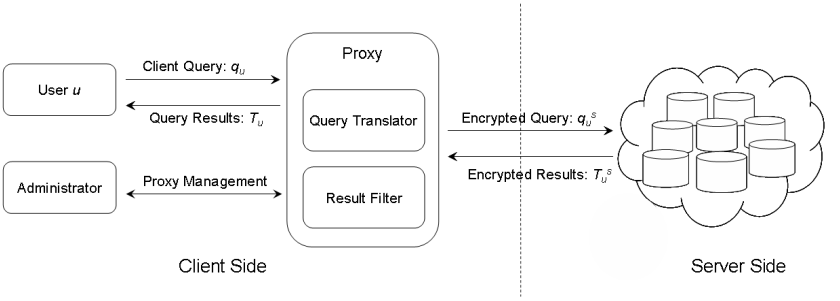


Fig. 1. The outsourced databases scenario

2 Background

We consider the typical outsourced database scenario described in DAS model [5]. According to this model, the plain relation $R = (\mathsf{A}_1, \mathsf{A}_2, ..., \mathsf{A}_n)$ is stored as corresponding encrypted relation $R^s = (etuple, \mathsf{A}_1^s, \mathsf{A}_2^s, ..., \mathsf{A}_n^s)$ on server where the attribute *etuple* denotes the encrypted tuple in R and A_i^s denotes the index for A_i in R^s with $i = 1, 2, ..., n$.

As illustrated in Fig. 1, when user u initiates a plain query q_u, a proxy at client side will translate it into a corresponding encrypted version q_u^s and deliver it to the remote server. After executing q_u^s over encrypted data on server side, the query result T_u^s will be returned to the proxy as replies. The proxy will decrypt all tuples in T_u^s, filter the decrypted tuples according to the conditions in q_u, and finally return results to user u.

Three parties are involved in the scenario where the users and the proxy are on client side and a provider on server side provides outsourced database services. In our threat model, the proxy is trust and secure, and the user can access any outsourced data based on his access rights. The service provider is honest but curious, sometimes a bit greedy. This means that the provider can provide the service he claims to be able to provide but he may leak some stored encrypted tuples out to others for curiosity or benefits. When no ambiguity is possible, we also call the service provider as the server.

Table 1(a) and 1(b) show an original relation with an access control list and corresponding encryption version on server side, respectively. For example, the tuple t_1, $\langle 8, beef, 22\rangle$, in Table 1(a) is stored as the encrypted tuple t_1^s, $\langle e_1, \mathsf{id}_u(8), \mathsf{id}_u(beef), \mathsf{id}_u(22)\rangle$, in Table 1(b) where $e_1 = encrypted(\langle 8, beef, 22\rangle)$ and $\mathsf{id}_u()$ is a mapping function, privately defined by user u, for attributes. This mapping function is used to construct value-based attribute indexes. The value-based index mapping is an injective function and it maps different attribute value to different images. The keyed hash function and the block ciphers can be used to construct value-based index.

It seems perfect when executing query over encrypted tuples is with user-specified index. However, the service provider is a pure storage service provider, he has no obligation to design appropriate storage constraints to separate tuple

Table 1. A Relation in plain and encrypted with ACL and value-based index

(a) Original Relation with ACL

tid	ACL	Mon	Cmdty	Qty
t_1	u	*8*	*beef*	*22*
t_2	u,v	*8*	*pork*	*25*
t_3	v,w	*8*	*ham*	*25*
t_4	u,v	*9*	*pork*	*23*
t_5	w	*9*	*beef*	*22*

(b) Encrypted Relation with Value-based Index

tid^s	$etuple$	Mon^s	Cmdty^s	Qty^s
t_1^s	e_1	$\mathsf{id}_u(8)$	$\mathsf{id}_u(beef)$	$\mathsf{id}_u(22)$
t_2^s	e_2	$\mathsf{id}_u(8)\mathsf{id}_v(8)$	$\mathsf{id}_u(pork)\mathsf{id}_v(pork)$	$\mathsf{id}_u(25)\mathsf{id}_v(25)$
t_3^s	e_3	$\mathsf{id}_v(8)\mathsf{id}_w(8)$	$\mathsf{id}_v(ham)\mathsf{id}_w(ham)$	$\mathsf{id}_v(25)\mathsf{id}_w(25)$
t_4^s	e_4	$\mathsf{id}_u(9)\mathsf{id}_v(9)$	$\mathsf{id}_u(pork)\mathsf{id}_v(pork)$	$\mathsf{id}_u(23)\mathsf{id}_v(23)$
t_5^s	e_5	$\mathsf{id}_w(9)$	$\mathsf{id}_w(beef)$	$\mathsf{id}_w(22)$

(c) Encrypted Relation with Conflict-free Value-based Index

tid^s	$etuple$	Mon^s	Cmdty^s	Qty^s
t_1^s	e_1	$\mathsf{id}_u(8,\ S_{u1})$	$\mathsf{id}_u(beef,\ S_{u1})$	$\mathsf{id}_u(22,\ S_{u1})$
t_2^s	e_2	$\mathsf{id}_u(8,\ S_{u2})\mathsf{id}_v(8,\ S_{v1})$	$\mathsf{id}_u(pork,\ S_{u1})\mathsf{id}_v(pork,\ S_{v1})$	$\mathsf{id}_u(25,\ S_{u1})\mathsf{id}_v(25,\ S_{v1})$
t_3^s	e_3	$\mathsf{id}_v(8,\ S_{v2})\mathsf{id}_w(8,\ S_{w1})$	$\mathsf{id}_v(ham,\ S_{v1})\mathsf{id}_w(ham,\ S_{w1})$	$\mathsf{id}_v(25,\ S_{v2})\mathsf{id}_w(25,\ S_{w1})$
t_4^s	e_4	$\mathsf{id}_u(9,\ S_{u2})\mathsf{id}_v(9,\ S_{v1})$	$\mathsf{id}_u(pork,\ S_{u1})\mathsf{id}_v(pork,\ S_{v1})$	$\mathsf{id}_u(23,\ S_{u1})\mathsf{id}_v(23,\ S_{v1})$
t_5^s	e_5	$\mathsf{id}_w(9,\ S_{w2})$	$\mathsf{id}_w(beef,\ S_{w1})$	$\mathsf{id}_w(22,\ S_{w1})$

sets on access rights. A set of encrypted tuples may be leaked intentionally or unintentionally. This means that an adversary user could potentially get some encrypted tuples that he cannot access. Though the adversary cannot take the plain values by decryption, the same index values could open a door to draw inferences on those tuples and thus the index-based inference attack could be launched.

As an instance for the index-based inference, considering the tuples t_1^s and t_2^s in Table 1(b). They are the encrypted versions of t_1 and t_2 in Table 1(a), respectively.

According to the access control mechanism, user u can access both tuples and user v can only access t_2^s. However, if v obtains the entire index value of $t_2^s.\mathsf{Mon}^s$, he may infer that the value of $\mathsf{id}_u(8)$ represents the index for attribute Mon with value *8*. This is because he realizes that both $\mathsf{id}_u(8)$ and $\mathsf{id}_v(8)$ appear in the field of attribute Mon^s of tuple t_2^s while he knows that $t_2.\mathsf{Mon}$ is with 8. If he could obtain the tuple t_1^s, which is with Mon^s value $\mathsf{id}_u(8)$, he can precisely infer that $t_1.\mathsf{Mon}$ is with value *8* though he has not the right to access t_1.

The reason why the index-based inference attack could be launched is because of the conflict between the equal attribute values and the unequal access control lists in some tuples. The tuples t_1 and t_2 are such tuples that are conflicting over attribute Mon. Note that the attribute values can be hidden by corresponding

index tag values, this inference can be prevented if the explicit equality relation between index tags is destroyed. The notion of *salt* is introduced in constructing the conflict-free mapping function id() [11].

Table 1(c) demonstrates an encrypted table with salted indexes. Recall the tuples t_1 and t_2 we discussed previously, it is easy to find that the user v can still realize that both $\mathsf{id}_u(8,S_{u2})$ and $\mathsf{id}_v(8, S_{v1})$ represent the attribute Mon index with 8, but he cannot directly infer any other tuple is with attribute Mon value 8 except the tuple t_2 which he can access.

However, this method cannot prevent the collusion between the server and users. Note that a query operation is on a set of constraints and it implies that any tuple in query results satisfies the constraints. For example, the returned result for the query q, **select** $*$ **from** R **where** $R.\mathsf{Mon} = 8$, implies that all those returned tuples are with attribute Mon value 8 whether executing decryption or not. Thus, the collusion-based inference attack exploited the implicit equality in tuples could be launched. For example, the collusion between the user v and the server s may lead v to decide $t_1.\mathsf{Mon}$ precisely based on the query q although he is not granted to access t_1.

3 The Collusion-Based Inference

In this section, we will explore the collusion-based inference which is based on the implicit equality relations among the single-attribute-single-value equality query results.

Let t and t^* be two different tuples in a relation R. Suppose their corresponding access control lists are denoted by $t.ACL$ and $t^*.ACL$, respectively.

Definition 1. *Two tuples t and t^* are said to be ACL-related if $t.ACL \neq t^*.ACL$ and $t.ACL \cap t^*.ACL \neq \phi$.*

Definition 2. *Given an attribute A in relation R, two tuples t and t^* are said to be A-related if $t.A = t^*.A$.*

Definition 3. *Two tuples t and t^* are said to be conflicting tuples over attribute A, denoted by $t \sim_A t^*$, if they are A-related and ACL-related.*

For the relation in Table 1(a), t_1 and t_2 are conflicting over attribute Mon because they are ACL-related and Mon-related.

Definition 4. *User u can collude with the server s if s can return u any query result included at least one tuple that u can access.*

Theorem 1. *Given two tuples t and t^* with $t \sim_A t^*$ and $t.ACL - t^*.ACL \neq \phi$. If user u, $u \in t.ACL - t^*.ACL$, can collude with server s, then u can precisely infer the value of $t^*.A$.*

The proof is sketched as the following.

Let $a = t.\mathsf{A}$. Consider query q, **select** $*$ **from** R **where** $R.A = a$. After user u initiates q, the server s will return tuple set T_u, $\{t_{u,1}, t_{u,2}, ..., t_{u,n_u}\}$, and we have

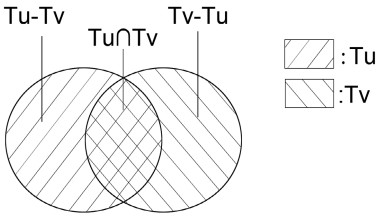


Fig. 2. Two tuple sets and their intersection

$t \in T_u$. Let $v \in t.ACL \cap t^*.ACL$. User v can also initiate query q, and then s returns tuple set T_v, $\{t_{v,1}, t_{v,2}, ..., t_{v,n_v}\}$. It is apparently that $t \in T_v$ and $t^* \in T_v$. If u can collude with s, he can get tuple set T_v, and hence he can determine that $t^*.\mathsf{A}$ is the same as $t.\mathsf{A}$.

In fact, user u can determine the value of $t^{'}.\mathsf{A}$ for any tuple $t^{'} \in T_v - T_u$. The collusion-based inference is a kind of link attack based on implicit equality. As illustrated in Fig. 2, user u connects T_v with T_u by colluding with server s. He can infer values in $T_v - T_u$ by the intersection $T_v \cap T_u$ and the implicit equality contained in T_v. To defend against this kind of link attack, we consider the generalization method [3].

Our idea is to construct the index on the generalized attribute value. Thus, even if the user u colludes with the server s and takes the set $T_v - T_u$, he cannot precisely infer any corresponding attribute values in set $T_v - T_u$. Hence, we need to measure the degree of information in T_v that user u can be used to infer a certain attribute values in $T_v - T_u$.

4 Generalization-Based Indexes

4.1 Generalization Mapping

Attributes can be classified into two types, the categorical and the continuous. The categorical attributes take values on discrete domains whereas the continuous take on continuous domains. Note that a continuous domain can be partitioned into a set of buckets and these buckets can be viewed as a discrete domain, we assume all attributes are categorical.

Although attributes take values on specific discrete domains, it is possible to construct general domains for some attributes. It means some specific attribute values can be mapped to a general value in a general domain.

Formally, a generalization on attribute A means that there is a generalization mapping gm_{A} from the original domain D to a general domain G such that $\forall a \in D, \exists g \in G, gm_{\mathsf{A}}(a) = g$. a is said to be a preimage of g and the set of preimages of g in tuple set T is defined as $Preimage(g) = \{t.\mathsf{A} | t \in T \wedge gm_{\mathsf{A}}(t.\mathsf{A}) = g\}$.

4.2 Measuring Privacy Protection of Generalization

The generalization replaces original values with generalized values and hence keeps the privacy of those specific values. We will propose some formulas to measure the privacy protected by a certain generalization mapping.

Given a tuple set T, a generalization mapping gm_{A}, and a generalization instance g, the entropy of the tuple set T_g, $\{t|t \in T \wedge gm_{\mathsf{A}}(t.\mathsf{A}) = g\}$, can be computed by

$$H(T_g) = -\sum_{t \in T_g \wedge t.\mathsf{A}=a} \frac{m_a}{m_g} \cdot \log \frac{m_a}{m_g}, \tag{1}$$

where m_g is the total number of tuples whose attribute values can be generalized to the instance g, and m_a is the number of tuples t with $t.\mathsf{A} = a$.

Let us consider the generalization version query, **select** * **from** R **where** $gm(R.\mathsf{A}) = g$. After user u executes this query, the server s will return tuple set $T_{u,g}$, $\{t_{u,1}, t_{u,2}, ..., t_{u,n_{ug}}\}$. Similarly, when user v executes, s returns set $T_{v,g}$, $\{t_{v,1}, t_{v,2}, ..., t_{v,n_{vg}}\}$. In the case of $T_{u,g} \cap T_{v,g} \neq \phi$, when user u colludes with s, he can get the set $T_{v,g}$ and realize that the attribute A values of all tuples in $T_{v,g}$ are generalized to g. Note that the user u knows the tuples in set $T_{u,g} \cap T_{v,g}$ and he always tries to infer the attribute A values of tuples in $T_{v,g} - T_{u,g}$ based on the known tuples. Therefore the question is transformed as how much information hidden in the set $T_{v,g}$ for the user u.

Let $gm_{\mathsf{A}}(a) = g$. Let m_g^- denote the number of tuples in $T_{v,g} - T_{u,g}$ and $m_g^\cap$ denote the number of tuples in $T_{v,g} \cap T_{u,g}$. Let m_a^- denote the number of tuples $t : t \in T_{v,g} - T_{u,g}$ with $t.\mathsf{A} = a$ and $m_a^\cap$ denote the number of tuples $t : t \in T_{v,g} \cap T_{u,g}$ with $t.\mathsf{A} = a$. We have

$$\begin{aligned}
H(T_{v,g}) &= -\sum_a \frac{m_a^- + m_a^\cap}{m_g^- + m_g^\cap} \log \frac{m_a^- + m_a^\cap}{m_g^- + m_g^\cap} \\
&= -\sum_a \frac{m_a^-}{m_g^- + m_g^\cap} \log \frac{m_a^- + m_a^\cap}{m_g^- + m_g^\cap} - \sum_a \frac{m_a^\cap}{m_g^- + m_g^\cap} \log \frac{m_a^- + m_a^\cap}{m_g^- + m_g^\cap} \\
&= -\frac{m_g^-}{m_g^- + m_g^\cap} \sum_a \frac{m_a^-}{m_g^-} \log \frac{m_a^- + m_a^\cap}{m_g^- + m_g^\cap} - \frac{m_g^\cap}{m_g^- + m_g^\cap} \sum_a \frac{m_a^\cap}{m_g^\cap} \log \frac{m_a^- + m_a^\cap}{m_g^- + m_g^\cap} \\
&= -\frac{m_g^-}{m_g^- + m_g^\cap} \sum_a \frac{m_a^-}{m_g^-} (\log \frac{m_a^- + m_a^\cap}{m_g^- + m_g^\cap} - \log \frac{m_a^-}{m_g^-}) - \frac{m_g^-}{m_g^- + m_g^\cap} \sum_a \frac{m_a^-}{m_g^-} \log \frac{m_a^-}{m_g^-} \\
&\quad - \frac{m_g^\cap}{m_g^- + m_g^\cap} \sum_a \frac{m_a^\cap}{m_g} (\log \frac{m_a^- + m_a^\cap}{m_g^- + m_g} - \log \frac{m_a^\cap}{m_g}) - \frac{m_g}{m_g^- + m_g} \sum_a \frac{m_a^\cap}{m_g} \log \frac{m_a^\cap}{m_g} \\
&= \frac{m_g^-}{m_g^- + m_g^\cap} D_{KL}(T_{v,g} - T_{u,g} || T_{v,g}) + \frac{m_g^-}{m_g^- + m_g^\cap} H(T_{v,g} - T_{u,g}) \\
&\quad + \frac{m_g^\cap}{m_g^- + m_g^\cap} D_{KL}(T_{v,g} \cap T_{u,g} || T_{v,g}) + \frac{m_g^\cap}{m_g^- + m_g^\cap} H(T_{u,g} \cap T_{v,g}),
\end{aligned}$$

where $D_{KL}(T_{v,g}-T_{u,g}||T_{v,g})$ is the Kullback-Leibler divergence [6] between $T_{v,g}-T_{u,g}$ and $T_{v,g}$, and $D_{KL}(T_{v,g} \cap T_{u,g}||T_{v,g})$ is the Kullback-Leibler divergence between $T_{v,g} \cap T_{u,g}$ and $T_{v,g}$.

Since the user u only takes the plain tuples in $T_{u,g} \cap T_{v,g}$, the hidden entropy $H_u(T_{v,g})$ in tuple set $T_{v,g}$ for the user u can be computed by

$$H_u(T_{v,g}) = H(T_{v,g}) - \frac{m_g^{\cap}}{m_g^{-} + m_g^{\cap}} H(T_{v,g} \cap T_{u,g}). \tag{2}$$

This formula could be used to measure the privacy protected in $T_{v,g}$ with respect to user u.

4.3 Generalization-Based Indexes

Definition 5. *Two tuples t and t^* are said to be generalized A-related if $gm_{\mathsf{A}}(t.\mathsf{A}) = gm_{\mathsf{A}}(t^*.\mathsf{A})$.*

For the relation in Table 1(a), when defining the generalization mapping over attribute Mon as $gm_{\mathsf{Mon}} : \{8,9\} \rightarrow \{[8,9]\}$, all the tuples in this table are generalized Mon-related.

Definition 6. *Two tuples t and t^* are said to be generalized conflicting tuples over attribute A, denoted by $t \sim_{gm_A} t^*$, if they are ACL-related and generalized A-related.*

A generalization-based index mapping maps a generalization value into an index tag value. Actually, it maps a set of attribute values into a single value. If two tuples are generalized conflicting, an adversary user could launch an index-based inference attack to infer the ranges of attribute values he cannot access. To mitigate this kind of inference, we can construct a conflict-free generalization-based index mapping by introducing the *salt*s as discussed in [11].

Definition 7. *The generalization mapping gm_{A} is α-secure if for any generalized conflicting tuple t and t^* with $\forall u \in t.ACL - t^*.ACL$ and $\forall v \in t.ACL \cap t^*.ACL$, $H_u(T_{v,g}) \geq \alpha$ where $g = gm_{\mathsf{A}}(t.\mathsf{A})$.*

Intuitively, we can use the α-secure generalization mapping to construct indexes against collusion-based inference. However, the lack of diversity in attribute value combinations can still lead a successful inference.

Recall the computation of $H_u(T_{v,g})$, the adversary user u can access plain tuples in set $T_{u,g} \cap T_{v,g}$ and he also knows the preimage set of g. Since he can obtain the encrypted version of tuples in $T_{v,g} - T_{u,g}$ by colluding with the server, he can construct a candidate plain set, say $T^c_{v-u,g}$, for $T_{v,g} - T_{u,g}$ by assigning preimage combinations and then compute $H^c_u(T_{v,g})$ according to the formula (2). If only one preimage combination makes $H^c_u(T_{v,g}) \geq \alpha$ satisfied, he can immediately determine the real set of $T_{v,g} - T_{u,g}$. We call the number of preimage combinations that satisfy $H^c_u(T_{v,g}) \geq \alpha$ as the diversity of the set $T_{v,g} - T_{u,g}$ and denote it as $\|T_{v,g} - T_{u,g}\|_{\mathsf{d}}$.

Table 2. Encrypted Relation with Conflict-free Generalization-based Index over Mon

ACL	tid^s	$etuple$	Mon_g^s
u	t_1^s	e_1	$\mathsf{id}_u([8,\ 9],\ \mathrm{S}_{u1})$
u,v	t_2^s	e_2	$\mathsf{id}_u([8,\ 9],\ \mathrm{S}_{u2})\mathsf{id}_v([8,\ 9],\ \mathrm{S}_{v1})$
v,w	t_3^s	e_3	$\mathsf{id}_v([8,\ 9],\ \mathrm{S}_{v2})\mathsf{id}_w([8,\ 9],\ \mathrm{S}_{w1})$
u,v	t_4^s	e_4	$\mathsf{id}_u([8,\ 9],\ \mathrm{S}_{u2})\mathsf{id}_v([8,\ 9],\ \mathrm{S}_{v1})$
w	t_5^s	e_5	$\mathsf{id}_w([8,\ 9],\ \mathrm{S}_{w2})$

Definition 8. *The generalization mapping gm_A is (k, α)-secure if for any generalized conflicting tuple t and t^* with $\forall u \in t.ACL - t^*.ACL$ and $\forall v \in t.ACL \cap t^*.ACL$, $H_u(T_{v,g}) \geq \alpha$ and $\|T_{v,g} - T_{u,g}\|_d \geq k$ where $g = gm_A(t.A)$.*

Table 2 demonstrates an encrypted relation with conflict-free generalization-based index over Mon corresponding to the relation in Table 1(a). We can show that the generalization mapping, $gm_{\mathsf{Mon}} : \{8, 9\} \rightarrow \{[8, 9]\}$, is (2, 0.25)-secure.

4.4 Generalization with Noise Tuples

Generalization fundamentally relies on spatial locality. In the case of attribute value generalization, it means that each tuple must have enough number of neighbor tuples whose corresponding attribute values can be involved in a same generalization class. However, the real data distribution is not as ideal as we expected. An outlier tuple is a tuple with markedly different features from others when they are generalized into a same equivalence class. The outlier tuples often lead generalization to a higher hierarchy.

To mitigate the negative effects introduced by outliers, some generalization methods adopt the suppression strategy [3]. For the scenario of outsourced databases, the suppression strategy will destroy the logical completeness of tuples. We consider the strategy of adding artificial noise tuples to datasets.

When constructing a (k, α)-secure generalization mapping, the parameter α is a key factor in determining generalization hierarchy. For example, to satisfy $H_u(T_{v,g}) \geq \alpha$, a general way is to increase the generalization mapping level, i.e., increase the number of distinct attribute values with image g. If outliers are involved in $T_{v,g}$, especially involved in $T_{v,g} - T_{u,g}$, it may need higher generalization level to satisfy $H_u(T_{v,g}) \geq \alpha$. However, if we add some artificial noise tuples with distinct attribute values but with the same mapping images as outliers, the inequality $H_u(T_{v,g}) \geq \alpha$ would be satisfied at lower generalization level.

5 Constructing (k, α)-Secure Conflict-Free Indexes

In this section, we discuss two key procedures, the `Generalize`() and the `Salt`(), for constructing (k, α)-secure conflict-free generalization-based indexes.

5.1 Attribute Generalization

```
procedure Generalize(Table T, Attribute A, Threshold k, α, β)
  CandSet = {t.A|t ∈ T}
  while CandSet ≠ φ
    pick a ∈ CandSet
    construct initial generalization mapping gm for a
    while TRUE and CandSet ≠ φ
      g = gm(a)
      CandSet ← CandSet − Preimage(g)
      if (gm is (k, α)-secure over Preimage(g))
        gm_A(a) ≜ gm(a) where a ∈ Preimage(g)
        break
      else
        add noise tuples with probability β
        increase generalization level with probability 1 − β
      endif
    endwhile
  endwhile
  if gm is (k, α)-secure
    return gm_A
  else
    return FAILED
  endif
endprocedure
```

The procedure Generalize() is for constructing (k, α)-secure generalization mapping over attribute A. We first initial a candidate value set *CandSet* for generalizing attribute A values. After choosing value a from *CandSet*, we construct a generalization mapping on a according to a predefined generalization hierarchy. Then a sequence of (k, α)-secure tests is launched. If all of the tests are passed, we will choose next candidate value to repeat construct-and-test procedure. If some of the tests are failed, we will choose the increasing-generalization-level strategy or the adding-noise-tuples strategy to repeat construct-and-test procedure according to the generalization parameter β.

5.2 Conflict-Free Indexes

We construct conflict-free generalization-based indexes by introducing random values, the *salt*s, as a parameter for index function id().

The procedure Salt() is for constructing conflict-free generalization-based indexes. We first construct the set of tuple-user pairs according to the tuple set and ACL lists. For each generalization image g, we construct user set U_g, $U_g = \{u|t \in T \wedge gm(t.\mathsf{A}) = g \wedge u \in t.ACL\}$. And then construct a user u associated tuple set $T_{u,g}$ and partition this set $T_{u,g}$ into a sequence of equivalence classes. Tuples in the same equivalence classes have the same ACL. Let $T_{u,g,t_1}, T_{u,g,t_2}, ..., T_{u,g,t_k}$ be such classes, where $T_{u,g} = \cup_{i=1}^{k} T_{u,g,t_i}$ and $T_{u,g,t_i} \cap T_{u,g,t_j} \neq \phi$ for $1 \leq i, j \leq k$ and $i \neq j$. It is easy to show that for

any pair of equivalence class T_{u,g,t_i} and T_{u,g,t_j}, we have $t_i \sim_{gm(\mathsf{A})} t_j$. And then different *salt*s are distributed to tuples in different classes to construct index mapping values.

```
procedure Salt(Table T, Attribute A, GenMapping gm)
  TU = {(t,u)|t ∈ T ∧ u ∈ t.ACL}
  while TU is not empty
    pick (t*,u*) ∈ TU;
    g = gm(t*.A)
    U_g = {u|(t,u) ∈ TU ∧ gm(t.A) = g}
    for each u ∈ U_g
      T_{u,g} = {t|(t,u) ∈ TU ∧ gm(t.A) = g}
      generate equivalence class subsets: T_{u,g,t_i}, 1 ≤ i ≤ n_u
      generate random numbers: S_{u_i}, 1 ≤ i ≤ n_u;
      for each t ∈ T_{u,g}
        i = arg(t ∈ T_{u,g,t_i})
        t.A^s_g = id_u(g, S_{u_i})
        TU ← TU − {(t,u)}
      endfor
    endfor
  endwhile
endprocedure
```

6 Experiments and Discussion

6.1 The Datasets

To evaluate the behavior of our proposed method, we need two types of materials for experiments, the data tuples and the authorized users for tuples.

For the data tuples, we first generate a relational table with 800000 tuples following the TCP-H benchmark specifications, and then randomly select $3000, 8000, 13000$, and 18000 tuples to construct tables Data3k, Data8k, Data13k, and Data18k, respectively. Each table contain the same three attributes, including $10000, 9999$, and 1000 distinct integers, respectively.

For the authorized users for tuples, we extract the authors coauthored with Professor Jiawei Han from the DBLP repository. In particular, we extract the top m most productive authors and construct authors set of size n from the repository. We view the constructed authors set as the authorized users set, i.e., the ACL lists for tuples. In our experiments, we set m as 40, 90, 140, and 190, respectively, and correspondingly, we set n as 60, 124, 204, and 297, respectively. We denote our constructed ACL lists as $ACL1$, $ACL2$, $ACL3$, and $ACL4$, respectively.

6.2 The Results

In our conducted experiments, we set $k = 2$ and the generalization mapping images are denoted by intervals. The secure threshold α is from 0.5 to 3 with step size 0.5, and the generalization parameter β is chosen in $\{0.4, 0.7, 1\}$.

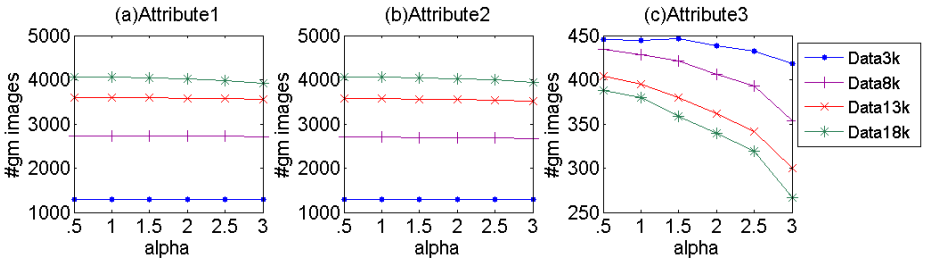


Fig. 3. The number of gm images on α (case of $ACL1$ and $\beta = 0.4$)

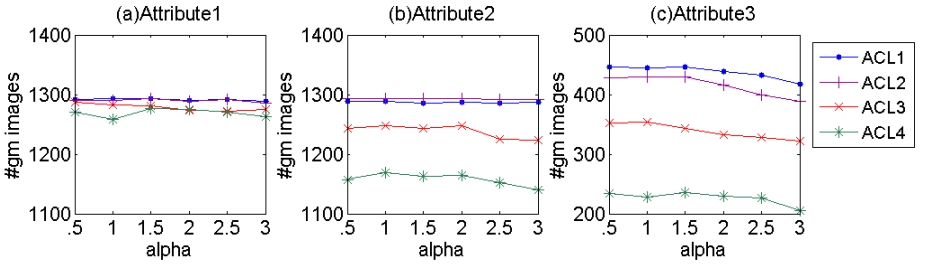


Fig. 4. The number of gm images on α (case of $Data3k$ and $\beta = 0.4$)

Fig.3 demonstrates the relationship between the number of generalization mapping, gm, images the secure threshold α for the case of the access control list $ACL1$ and the generalization parameter $\beta = 0.4$. We find that the number of gm images is decreasing as the α increases. This implies that more original attribute values are mapped to a single images averagely in order to satisfy the larger α.

When comparing with the Fig.3(a) and (b), we find the number of gm images are similar in all four datasets in our experiments. It means that the number of images depends on the sizes of attribute values domains. Intuitively, the larger size of dataset, larger number of images. However, the number of images is not consistent with the size of dataset. For example, Fig.3(c) shows that the larger size of dataset, the smaller number of images. This is because of the generalization mapping images in our conducted experiment are on intervals. Given an access control list, the larger dataset with smaller size of domain may introduce more outlier attribute values. This figure implies that the number of gm images depends on the size of attribute domain when giving a certain access control list.

Fig.4 demonstrates the trend of the number of gm images on the secure threshold α in the case of $Data3k$ and $\beta = 0.4$. As shown in Fig.3, the larger value of α leads smaller number of the gm images. It also shows that the access control list with more users makes the number of gm images smaller given a dataset. This implies that the introduced complexity of larger access control list require more different original values mapping to a single generalization value.

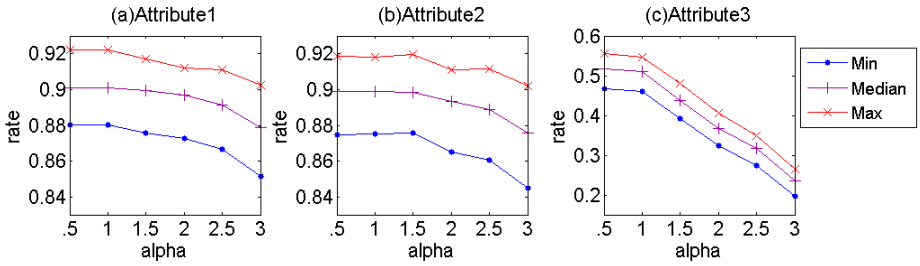


Fig. 5. The hitted ratios on α (case of $Data13k$ and $ACL1$)

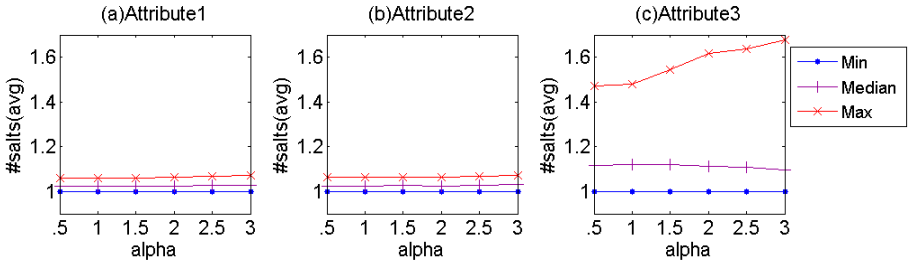


Fig. 6. The number of *salt*s and α (case for $Data13k$ and $ACL1$)

Generalization maps specified values into a general value. It means that when initiating a query from client, the returned tuple set will contain some tuples unsatisfied original query conditions. The hitted ratio reflects the ratio of tuples satisfied original conditions. For a single-attribute-single-value query, i.e., the query **select** * **from** R **where** $R.\mathsf{A} = a$, the hitted ratio can be computed by $\frac{m_a}{m_g}$. In our experiments, we randomly generate single-attribute-single-value queries for different users and compute $Max, Median$, and Min hitted ratios for user queries in different attributes. Figure 5 demonstrates the relation between hitted ratios and α with $Data13k$ and $ACL1$. It shows that the hitted ratio drops when the generalization level becomes higher.

To defend against index-based inference, it needs different attribute index values for conflicting tuples. Fig.6 demonstrates that in our conducted experiments, limited number of salts can be used for users to prevent index-based inference.

6.3 Related Work

Outsourcing data to third parties out of the control of data owners requires storing data encrypted on remote servers. To avoid storing many different encrypted versions of a same tuple on servers, encrypting each tuple with a single key is a common knowledge. Since the early efforts on outsourced databases [1][5] are focused on how to translate the client-side plain queries into corresponding

server-side encrypted versions, they assume that all the tuples are encrypted by a same key. It implies that a certain user may have the full rights to access any encrypted tuples if he gets the decryption key.

The selective encryption methods use different keys to encrypt different data portions such as tuples or attributes [7]. To avoid users from managing too many keys, the keys can be derived from user hierarchy [2]. And also, the traditional ciphers are replaced with the attribute-based encryption (ABE) method to encrypt data [14]. However, the access controls provided by these methods depend on the readability of decrypted data. This means that some decryption efforts on client side are wasteful.

Other efforts are on developing new ciphers for keyword searching on encrypted data. However, either the symmetric encryption scheme [10] or the asymmetric encryption scheme [13] cannot prevent the curious service provider locating the positions with the same method. We note that locating encrypted tuples implies execute comparison operations over encrypted data on server without decryption. The partially [8] or fully [4] homomorphic encryption methods can be used to perform the comparison. But, as mentioned previously, if the comparison results could be distinguished on server, the curious service provider could also manipulate in the same way to obtain the results of comparison.

To improves the speed of encrypted data retrieval operations on server, several index techniques are proposed. The CryptDB scheme [9] defines layers of encryption for different types of database queries. For executing a specific query, layers of encryption are removed by decrypting to an appropriate layer and the tuple index is directly on the encrypted data. This method may lead many sensitive values be stored to the level defined by the weakest encryption scheme. No inference attacks are considered in this scheme. The DAS (Database as a Service) model [5] proposes a bucketization method to construct the index. This index is defined on an auxiliary attribute which is associated with the corresponding original attribute. However, there is no formal security analysis about this kind of index. Value-based index is discussed in [1]. Comparing with the bucketization index, the value-based index locate encrypted data in high accuracy but also disclose many other useful information such as the data distribution.

To our knowledge, the authors in [11] firstly address the inferences of encrypted data in outsourced databases. They discuss a kind of inference attack introduced by the value-based index. The addressed inferences are on the explicit equality relations among tuples with different access rights. But they do not address the collusion-based inference on the implicit equality relations among retrieved tuples.

Our proposed generalization-based index can be viewed as a kind of bucketization index. Comparing with the index in [5], we define a combined measure to measure the inference resistance against the collusion-based inference. We also introduce the notion of salt which is proposed in [11] to construct conflict-free generalization-based index.

7 Conclusion

Ensuring data privacy is fundamental for outsourced databases. Indexes over encrypted data provide efficient selective queries on server side. However, the plaintext-associated information hidden in those indexes may introduce inference attacks. In this paper, we investigate a kind of inference attacks based on implicit equality relations hidden in query results. To defend against this attack, we develop a generalization-based method to construct private indexes. We design a combined metric, quantized by the entropy values and attribute value diversities, to measure the inference resistance for our proposed methods. We have conducted some experiments to validate our proposed method.

References

1. Damiani, E., Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing Confidentiality and Efficiency in Untrusted Relational DBMSs. In: Proceedings of ACM CCS 2003, pp. 93–102 (2003)
2. Damiani, E., Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Key Management for Multi-user Encrypted Databases. In: Proceedings of StorageSS 2005, pp. 74–83 (2005)
3. Fung, B., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving Data Publishing: a Survey of Recent Developments. ACM Computing Surveys 42(4), 14:1–14:53 (2010)
4. Gentry, C.: Fully Homomorphic Encryption Using Ideal Lattices. In: Proceedings of STOC 2009, pp. 169–178 (2009)
5. Hacigumus, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over Encrypted Data in the Database-Service-Provider Model. In: Proceedings of ACM SIGMOD 2002, pp. 216–227 (2002)
6. Kullback, S., Leibler, R.: On Information and Sufficiency. Annals of Mathematical Statistics 22(1), 79–86 (1951)
7. Miklau, G., Suciu, D.: Controlling Access to Published Data Using Cryptography. In: Proceedings of VLDB 2003, pp. 898–909 (2003)
8. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
9. Popa, R., Redfield, C., Zeldovich, N., Balakrishnan, H.: CryptDB: Protecting Confidentiality with Encrypted Query Processing. In: Proceedings of SOSP 2001, pp. 85–100 (2011)
10. Song, D., Wagner, D., Perrig, A.: Practical Techniques for Searches on Encrypted Data. In: Proceedings of IEEE S&P 2000, pp. 44–55 (2000)
11. Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Private Data Indexes for Selective Access to Outsourced Data. In: Prodeedings of WPES 2011, pp. 69–80 (2011)
12. Wang, H., Lakshmanan, L.: Efficient Secure Query Evaluation over Encrypted XML Databases. In: Proceedings of VLDB 2006, pp. 127–138 (2006)
13. Yang, G., Tan, C.H., Huang, Q., Wong, D.S.: Probabilistic Public Key Encryption with Equality Test. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 119–131. Springer, Heidelberg (2010)
14. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. In: Proceedings of INFOCOM 2010, pp. 534–542 (2010)

Distributed AH-Tree Based Index Technology for Multi-channel Wireless Data Broadcast*

Yongtian Yang[1], Xiaofeng Gao[1,**], Xin Lu[2], Jiaofei Zhong[3], and Guihai Chen[1]

[1] Dept. of Computer Science and Engineering, Shanghai Jiao Tong Univ., China
[2] School of Software, Shanghai Jiao Tong Univ., China
[3] Dept. of Mathematics and Computer Science, Univ. of Central Missouri, USA

Abstract. Alphabetic Huffman Tree (AH-Tree) is an appropriate data structure to index data set with skewed access frequencies, which fits the feature of web-based wireless data broadcast service to a mass number of mobile clients. In this paper we solve a long-time open question to construct an arbitrary k-ary AH-Tree with Hu-Tucker algorithm [1] by dynamic programming, whose time complexity is $O(kn^2)$, where n is cardinality of the data set. We then build a distributed AH-Tree index sequence with a special control-table shrinking technique. Next, we introduce a pyramid index allocation method, which is scalable to any available broadcast channel. We prove the correctness of our algorithm, analyze the time complexity of tree-construction process, and compare our design with B^+-Tree index by numerical experiments. Both mathematical analysis and simulation results prove the efficiency of our design. To the best of our knowledge, we are the first to propose a detailed, fast, and distributed k-ary AH-Tree index with allocation protocol, which has both theoretical and practical significance in this area.

Keywords: Huffman Tree, Distributed Index, Data Broadcast.

1 Introduction

Wireless data broadcast is an efficient data dissemination technology to a mass number of mobile clients with battery-constraint portable wireless devices (i.e., mobile phones, smart phones, and PDAs). Due to the nature of wireless communication, instead of point-to-point query-reply mode, a server broadcasts public information like traffic conditions, live TV streams, weather forecasts, and tourist services, etc., over multiple channels periodically. Each mobile client can access onto the channel, wait for the required data items, and download its required data packet sequence each at a time slot.

Intuitively, the criteria to evaluate the performance of a wireless data broadcast system are the downloading time and energy consumption of mobile devices.

* This work has been supported in part by the National Natural Science Foundation of China (Grant numbers 61073152, 61133006, and 61202024), China 973 project (2012CB316200), Shanghai Educational Development Foundation (Chenguang Grant No.12CG09), and the Natural Science Foundation of Shanghai (Grant No.12ZR1445000).

** Corresponding author.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 176–192, 2013.

Correspondingly, *access latency* and *tuning time* are two widely accepted system evaluation standards. The former denotes the time interval from when a client sends a request to the time when it receives the required datum, while the latter denotes the activating time of the mobile device during data retrieval process.

Indexing technology is one of the most effective methods to reduce tuning time. Each mobile device has two modes: active mode and doze mode. Its energy consumption during active mode is far greater than that in doze mode. Indices help to reduce the active time of a mobile device significantly. Clients can follow the direction of indices on broadcasting channels, turn off during the waiting period, and turn on again right before the required data item appears.

There have been a lot of works discussing efficient indexing schemes, which can be classified into three categories: hashing-based indexing [2], tree-based indexing (*e.g.*, B^+ tree [3], Huffman tree [4]), and table-based indexing (*e.g.*, exponential index [5]). Among them, tree-based indices are the most widely implemented structures according to their easy-searching and fast-constructing characteristics. Additionally, researchers prefer to choose a balanced tree as a base for their index design, since it is easier to control the tree height and bound the index size to avoid large increase of access latency.

However, data broadcast system serves mobile clients with hundreds or thousands of data items, whose visiting frequencies vary hugely according to the empirical statistics of human behavior with preference difference. Based on the investigation of website popularity [6], we find that a few data items have very high popularity; a medium number of data items have middle-of-the-road visiting frequencies; while a huge number of data items actually have very low preference. Such phenomenon implies that balanced tree is probably not an appropriate index structure for web service since each searching path has similar length, bringing longer tuning time on average.

To overcome the aforementioned shortcoming, the *Alphabet Huffman Tree* (AH-Tree) is a good choice. In a typical AH-Tree, the higher the frequency of a data item, the shorter the path from root to the corresponding leaf index. Many literatures have studied AH-Tree index during the past two decades. In [1], Hu and Tucker first proposed a binary AH-Tree algorithm with time complexity $O(n \log n)$, where n is the size of data items. Their design is based on a complex queue technology, which cannot be directly extended to construct k-ary AH-Tree with arbitrary k [7]. Later, Shivakumar et al. [8] extended Hu-Tucker Algorithm into k-ary AH-Tree, and first implemented it as indices to broadcast environment. Nevertheless, they did not describe the algorithm clearly. They only mentioned a skeleton of how to build a k-ary AH-Tree, lack of specification of internal node construction with k branches. The time complexity of such design would be high up to $O(n^3)$ if we directly follow their description without creating any particular queue structure, which is impractical for real-world applications. Similarly, [4], [9], and [10] discussed AH-Tree index for data broadcast problem respectively, none of which provided complete tree-construction process with time complexity analysis. They only gave simple explanation according to

the description in [8], almost in the same manner. Therefore, how to construct an arbitrary k-ary AH-Tree by Hu-Tucker algorithm remains an open problem.

Additionally, researchers tended to modify the index tree into a distributed index sequence to improve the performance [3,10,11]. This scheme relies on the use of control table. However, we found that the control table also results in overhead in space, which becomes an unneglectable factor as we found half of the control table is redundant. Moreover, the control table contains as much redundancy as the useful information. How to eliminate these redundancy becomes crucial.

In this paper, we propose an efficient AH-Tree construction with the help of dynamic programming. Our algorithm can build arbitrary k-ary AH-Tree index with bounded tree height in $O(kn^2)$ time. We then modify this tree into a distributed index sequence with control table design to further reduce the searching steps. Considering the influence of control table size, we further propose a general and effective scheme to eliminate redundant entries in control tables to reduce the overall index packet length, which save almost 50% of the storage. Finally, we use the dynamic pyramid scheme for index and data allocation. We prove the correctness and complexity of our design theoretically and illustrate the performance of the broadcast system by numerical experiments. Both theoretical proofs and simulation results validate the efficiency of our design. To the best of our knowledge, we are the first to propose the detailed design for k-ary Hu-Tucker algorithm with time complexity analysis. Our design does not rely on any special data structure and queue design, which can be easily implemented in any practical system. The simulation results show that our design gains significant growth regard to skew distributed data. However, it does not over perform B-Tree with regard to uniform distribution [12].

The rest of this paper is organized as follows. Section 2 summarizes the related work in this research area. Section 3 illustrates the problem formulation and the architecture of broadcast system. In Sec. 4 we introduce the dynamic programming to construct k-ary AH-Tree index and provide the correctness proofs, while in Sec. 5 we complete the process to build a distributed index sequence with control tables. Section 6 describes the index allocation method. In Sec. 7 we compare our design with the latest work in [12] and prove the efficiency of our construction. Finally, Section 8 gives conclusion and future works.

2 Related Works

The key research topics in wireless data broadcast are basically focusing on how to deign index structures and how to allocate data onto channels, in order to reduce access latency and tuning time [13].

A series of research works deal with data scheduling problem to decrease access latency, without implementing indexing technology [14]. As a result, the tuning time is as long as the access latency, which still leads to high power consumption for mobile devices.

Traditional disk-based indexing techniques have been modified to meet the requirement of data broadcast systems, which can be classified into three categories: hashing-based [2], tree-based (*e.g.*, Huffman tree [4]), and table-based (*e.g.*, ex-

ponential index [5]) schemes. Hashing-based schemes use hash functions to distribute data onto channels. For instance, Yao *et al.* [2] proposed MHash to facilitate skewed access probabilities and reduce access latency. Table-based schemes include exponential index proposed by Xu *et al.* [5], which shares links in different search tables and allows users to start searching at any index node. However, this scheme may not perform well under non-uniform access probabilities. Tree-based schemes are sometimes faster to design and easier to maintain, thus achieving more attentions. One common tree-based index, *i.e.* B^+-tree distributed index (*BTD*) was extended to satisfy different system requirements. Gao *et al.* [11] redesigned BTD and built a complete multi-channel broadcasting system with non-uniform data access probabilities and unequal data sizes.

When it comes to multi-channel data broadcasting, how to allocate index and data will produce heavy impact on the performance of each scheme. A certain allocation method could be helpful to a specific index structure, but at the same time it might reduce the efficiency of another index scheme. Several works [10,15] deal with data allocation for multi-channel data broadcast. One work [16] proposed an index allocation method named TMBT, which creates a virtual BTD for each data channel and multiplexes them on the index channel.

Huffman tree is a skewed index tree that takes into account the data access probability, where more popular data have shorter search paths from the root of the tree [8]. However, most existing works discussed Huffman tree for a certain data type with special constraints and features. Recently, Zhong *et al.* [9] proposed a uniform AH-Tree indexing scheme to satisfy all possible environments.

Based on the observation that the previous schemes can be further improved, we propose a novel AH-Tree based indexing approach under multi-channel environment, where data items have different access probabilities. Our scheme is further refined to minimize both average access latency and tuning time, while the performance compared to other related schemes is also provided. Simulation results confirm the efficiency of our scheme.

3 Problem Formulation and System Architecture

We consider multi-channel wireless data broadcast with a server and numbers of clients. The server first retrieves data from its local database and then calls the Index Generator modular to index the data. Next, the Channel Allocation modular allocates channels to data and index. We take index-data separation mode in this paper to reduce the possible switches among channels. After that, the server periodically broadcasts data and index sequences in a fixed range over wireless channels. Clients can access the data at anytime by tuning onto the channels. In this paper, we focus on Index Generator and Channel Allocator. We propose distributed AH-Tree based index sequence in Index Generator modular and pyramid index allocation scheme in the Channel Allocation modular.

Let $D = \{d_1, d_2, \ldots, d_t\}$ be the data set to broadcast, where t is the number of data item. Associated with D, $P = \{p_1, p_2, \ldots, p_t\}$ is the access frequency set, where p_i is the access frequency of d_i. Also, since each d_i may have different size, we use s_i, measured by KB, to represent the size of d_i and $S = \{s_1, s_2, \ldots, s_t\}$,

There are $A = m + n$ channels in the system. We assign m channels to the data and n channels to the index. The channel set is $\mathbf{C} = \{C_1, C_2, \ldots, C_A\}$.

For clarity, we summarize the symbols with their meanings in Table 1. Some of them will be described in the following sections.

Table 1. Symbol Description

Sym	Description	Sym	Description
D	Data set. $D = \{d_1, \cdots, d_t\}$	$\mathbf{C}$	Channels. $\mathbf{C} = \{C_1, \cdots, C_A\}$
L	Level of T	l	Cut level of T
T	An AH-Tree	t	Number of data items
k	Maximum branch no. for T	B_i^j	The j^{th} index at i^{th} level of T
N	Node set of index tree	D_k	The k^{th} datum in T
A	Available channels $A = m + n$	Δ_i	The i^{th} sub-tree at level $l + 1$
m	Number of data channels	PATH(B_i^j)	A path from B_1^1 to B_i^j
MAX(B_i^j)	Maximum key B_i^j domains	n	Number of index channels

4 Basic Index Technology

In this section, we describe the index technique used in the Index Generator modular and an AH-Tree index technique is proposed.

4.1 Tree Construction

First, let us introduce the two-stage construction process of the k-ary AH-Tree [1,8]. In the first stage, we build an optimal k-ary Huffman tree without

Algorithm 1. AH-Tree Construction

1: **Input:** D, P;
2: **Output:** Node set N of AH-Tree T.
3: Create t leave nodes for $d_i \in D$ and push them into N. Set $I = \{1, \cdots, t\}$.
4: **while** $|I| > 1$ **do**
5: **if** $\sum_{i=1}^{k} p_{n_i} = \min(\sum_{i=1}^{k} p_{x_i})$, where n_i, $x_i \in I$ **and** no leaves among $n_1, \cdots, n_k$ **and** $n_1, \cdots, n_k$ are the leftmost k nodes **then**
6: Merge $n_1, \cdots, n_k$ as n' with $r_{n'} = \sum_{i=1}^{k} r_{n_i}$ (n' is parent of $n_1, \cdots, n_k$) ;
7: Insert n' into N, mark $n_1, \cdots, n_k$ as "processed" and remove them from I;
8: **end if**
9: $n = n - (k - 1)$;
10: **end while**
11: Traverse T, mark each node's level from the root and get max level L;
12: **for** $l = L \rightarrow 2$ **do**
13: Find the leftmost index node p on the $(l-1)^{th}$ level and the leftmost k nodes $n_1, \cdots, n_k$ on the l^{th} level, and then mark them;
14: Record "p" into field *new_parent* of $n_1, \cdots, n_k$ and "$n_1, \cdots, n_k$" into array *new_children* of p without altering their original parent/children;
15: Keep finding new nodes until no unmarked nodes exist in level l;
16: **end for**
17: Replace *parent* and *children* of all nodes with *new_parent* and *new_children*.

alphabetic order, where dynamic programming is employed to simplify the algorithm. In the second stage, we adjust the tree in a bottom-up approach to generate a new tree, which preserves the alphabetic order while keeping the same cost. The construction process is shown in Alg. 1.

Stage 1 (Line 4 to 10) contains several iterations. During each iteration, we select k nodes to merge into one new node and then insert it into the original data sequence to form a new sequence. Recall that k is the number of branches of the tree. The selected k nodes should satisfy three conditions: (1). There are no leaf nodes among them; (2). The sum of their frequencies is the minimum among all k candidate groups; and (3). They should be the leftmost nodes.

We create a new index node as the parent of these k nodes with the frequency as the sum of the k nodes. Then we insert the new node into the data sequence, mark the k nodes as processed leaf nodes, and delete them from the sequence. After that, we start a new iteration and continue the process until there is only one node left in the sequence, which is the root of the tree. At the end of Stage 1, we produce a tree T' without alphabetic order.

Stage 2 (Line 11 to 17) adjusts the tree in a bottom-up, left-right approach such that every k consecutive nodes on the same level have the same parent. Finally, an AH-Tree T is constructed. The correctness proof is provided in [17].

Example 1. Throughout the paper, we use a data set in Table 2. The frequency indicates how many times a datum has been accessed from historical record.

Table 2. Example Data Set

Key	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Fre	16	8	30	4	1	12	27	36	41	2	9	15	19	7	23	1

Based on this data set, we apply Alg. 1 to construct the tree. After two stages, we generate T' and T as shown in Fig. 1, respectively.

4.2 Subroutine Dynamic Programming

It is time-consuming to choose k nodes from the data sequence in stage 1. In this subsection, we design a dynamic programming to solve this problem in $O(ki)$, where i is the number of nodes in the current iteration and k is the number of branches in the tree. We first describe the basic idea and derive the recursive relation, and then verify the correctness in the following part.

Consider the problem of selecting j nodes from sequence $[n_1, n_2, \ldots, n_i]$. There are only two cases to be considered.

Case 1: There is at least one unselected leaf node in $[j+1, \ldots, i]$. Let $f(i, j)$ be the minimal weight sum of the j selected nodes in this case.
Case 2: There is no unselected leaf node in $[j+1, \ldots, i]$. Let $g(i, j)$ be the minimal weight sum of the j selected nodes in this case.

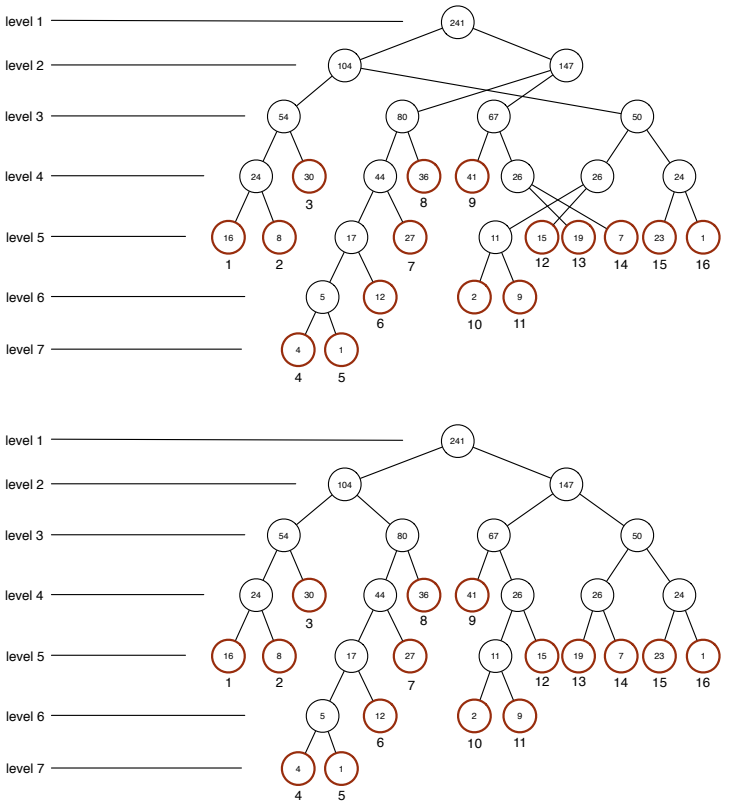


Fig. 1. T' (Tree construction after Stage 1) and T' (Tree construction after Stage 2)

We now derive the recursive relation. In Case 1, the i^{th} node is unselected, otherwise no leaf node exists in $[j+1, \ldots, i-1]$. There are also two subcases:

1. If the i^{th} node is an index node, then there is at least one unselected leaf node in $[j+1, \ldots, i-1]$. We need to solve the subproblem $f(i-1, j)$;
2. If the i^{th} node is a leaf node, then there may be no leaf node in $[j+1, \ldots, i-1]$. We have to consider both $f(i-1, j)$ and $g(i-1, j)$ and choose the minimum.

Then the recursive relation for f is shown below:

$$f(i,j) = \begin{cases} f(i-1,j), & \text{if } i^{th} \text{ node is an index node} \\ \min(f(i-1,j), g(i-1,j)), & \text{if } i^{th} \text{ node is an unmarked leaf} \end{cases}$$

In Case 2, there are also two subcases:

1. If the i^{th} node was an unselected leaf node, it must be selected now, otherwise there will be an unselected leaf node in $[j+1, \ldots, i]$, which violates our assumption of Case 2. So we need to consider the subproblem of $g(i-1, j-1)$.

2. If the i^{th} node is an index node, since whether select it or not will not violate the condition, we have to consider $\min(g(i-1,j-1), g(i-1,j))$.

Hence, the recursive relation for g is as following:

$$g(i,j) = \begin{cases} g(i-1,j-1) + r_i, & \text{if } i^{th} \text{ node is an unmarked leaf} \\ \min(g(i-1,j-1) + r_i, g(i-1,j)), & \text{if } i^{th} \text{ node is an index node} \end{cases}$$

After computing $f(n,k)$ and $g(n,k)$, the final result is $\min(f(n,k), g(n,k))$. We record the choices when generating values in f and g to locate the k nodes.

4.3 Correctness of the Dynamic Programming

We now verify the correctness of our design. The first step is to prove that the problem has *optimal substructure*. In the following, we show that both f and g have optimal substructures. Let's start with g, which is independent to f.

Lemma 1. *(Optimal Substructure of g)* Let $S_i = [d_1, d_2, \ldots, d_i]$ be the data sequence, and $Z_j = [d_{i_1}, \ldots, d_{i_j}]$ be any optimal solution satisfying the condition of Case 2 (we also call it an optimal solution of S_i for convenience). Then:

1. If d_i is an unselected leaf node, then Z_{j-1} is an optimal solution of S_{i-1};
2. If d_i is an index node, then Z_{j-1} is an optimal solution of S_{i-1} if $d_i = d_{i_j}$; otherwise Z_j is an optimal solution of S_{i-1}.

Proof. *Condition 1.* If d_i is a leaf node, then it is surely in Z_j. Suppose $Z'_{j-1} = [d'_{i_1}, \ldots, d'_{i_{j-1}}]$ is an optimal solution of S_{i-1} with cost less than Z_{j-1}, then we replace Z_{j-1} with Z'_{j-1} in Z_j to get Z'_j. Obviously Z'_j is a solution of S_n whose cost is less than Z_j. It contradicts to the condition that Z_j is an optimal solution.

Condition 2. If d_i is an index node, then there are two cases:
(a) if $d_i = d_{i_j}$, that is, we select the i^{th} node. We can use the same method in Condition 1 to prove that Z_{j-1} is an optimal solution of S_{i-1}.
(b) if $d_i \neq d_{i_j}$, that is, the i^{th} node is unselected. Suppose $Z'_j = [d'_{i_1}, d'_{i_2}, \ldots, d'_{i_j}]$, with a cost less than Z_j, is an optimal solution of S_{i-1}. Since the i^{th} node is an index node, it follows that Z'_j is also a solution of S_i. Then we find a solution of S_i with a cost less than Z_j, which contradicts the condition that Z_j is an optimal solution of S_i. Hence, Z_j is an optimal solution of S_{i-1}.

From above, we conclude that g has *optimal substructure*. □

Lemma 2 shows that solving f also contains the *optimal substructure*.

Lemma 2. *(Optimal substructure of f)* Let $S'_i = [d_1, d_2, \ldots, d_i]$ be the sequence, and $Z_j = [d_{i_1}, d_{i_2}, \ldots, d_{i_j}]$ be any optimal solution that satisfies Case 1 (we also refer it an optimal solution of S'_i), then Z_j is an optimal solution of S'_{i-1}.

Proof. The proof is divided into two parts:

Part 1. If d_i is an index node, there is at least one leaf node in $[j+1, \ldots, i-1]$. Thus Z_j is a solution of S'_{i-1}. Let $Z'_j = [d'_{i_1}, d'_{i_2}, \ldots, d'_{i_j}]$ be an optimal solution of S'_{i-1}, which costs less than Z_j. In the case of Z'_j, there must be at least one

unselected leaf node in $[j+1, \ldots, i]$ since d_i is an index node. So Z'_j is a solution of S'_i. That is, there is another solution which costs less than Z_j. Contradiction.

Part 2. d_i is an unselected leaf node. Since d_i is unselected, we prove that Z_j is an optimal solution of both S'_{i-1} and S_{i-1}. Suppose Z'_j is an optimal solution of S_{i-1} and S'_{i-1} with a cost less than Z_j. Since d_i is a leaf node, Z'_j is also a solution of S'_i. Thus we find a solution of S'_i that costs less than Z_j, which contradicts the condition. Thus, Z_j is an optimal solution of S_{j-1} and S'_{j-1}. □

Lemma 1 and Lemma 2 imply that both f and g have optimal substructures. The next two lemmas will complete the final conclusion, in which Lemma 3 can be directly derived from Lemma 1 and Lemma 2.

Lemma 3. Solving $\min(f(n,k), g(n,k))$ problem has optimal substructure.

Lemma 4. Solving $\min(f(n,k), g(n,k))$ problem has overlapping subproblem.

Proof. Solving $\min(f(n,k), g(n,k))$ involves solving $f(n-1,k)$ and $f(n-1,k-1)$, which are two overlapping problems. Thus it has overlapping subproblem. □

Theorem 1. If $Z_j = [d_{n_1}, \ldots, d_{n_k}]$ is the output of our dynamic programming, then it satisfies three conditions:

1. There are no leaf nodes among these k nodes.
2. The frequency sum of Z_j is the minimum among all possible selection.
3. The k nodes should be the leftmost ones among all the candidates.

Proof. It follows directly from Lemma 3 and Lemma 4. □

5 Distributed AH-Tree Construction

We have constructed an AH-Tree using dynamic programming in Sec. 4. To avoid searching from the tree root every time, we employ the distributed index technique. This technique is first introduced in [3] and applied to B^+ tree index. Then it is applied to AH-Tree index in [9]. The tree is split into *replicated* part and *non-replicated* part. The basic idea is to add the dominating range of all ancestors into the replicated-part nodes. Thus, from any replicated-part node, we know which subtree we are looking for, and can directly tune to it.

5.1 Control Table

First we introduce some notations. The index nodes are divided into two parts. Suppose l is the *cut level*. The nodes in replicated part are called *control index*, while the remaining nodes are named *search index*. B_i^j denotes the j^{th} index node of level i, and D_k represents the k^{th} data node in the tree. Note that if the j^{th} node of level i is a data node, then B_i^j does not exist.

Let Δ_i denote the i^{th} subtree below the cut level l, rooted at B_{l+1}^i. To generate the distributed index sequence, we use $\text{PATH}(B_i^j)$ to represent a path from the root to B_i^j excluding B_i^j. For instance, $\text{PATH}(B_4^4)$ in Fig. 1 is $[B_1^1, B_2^2, B_3^3]$. In

order to broadcast data, we linearize the tree by depth-first search. We use $B_i^{j[x]}$ to denote the x^{th} appearance of B_i^j during the process of depth-first search.

For each control index $B_i^{j[x]}$, suppose that PATH$(B_i^{j[x]}) = \{B_1^{1[x_1]}, B_2^{j_2[x_2]}, \ldots, B_{i-1}^{j_{i-1}[x_{i-1}]}\}$, then the control table of $B_i^{j[x]}$ is shown in Table 3.

Table 3. Format of the Control Table

1	MAX(Δ_{g-1})	$B_1^{1[1]}$
2	MAX($B_2^{j_2}$)	$B_1^{1[x_1+1]}$
...	...	...
r	MAX($B_r^{j_r}$)	$B_{r-1}^{j_{r-1}[x_r+1]}$
...	...	...
i	MAX($B_i^{j_i}$)	$B_{i-1}^{j_{i-1}[x_i+1]}$

Each entry of the control table contains two elements: the key value and the control index it hops to. The MAX(Δ_{g-1}) in the first entry gives a lower bound of the dominating range of $B_i^{j[x]}$. If the key k we are looking for is less than MAX(Δ_{g-1}), it means that k has been broadcast, so we have to wait for another round. In this case, we jump to $B_1^{1[1]}$. The key value in the r^{th} entry gives an upper bound of the dominating range of $B_r^{j_r}$. The second element gives a hint of which subtree to hop to in the next step. When the client wants to retrieve a datum with key greater than this element, it should directly hop to the control index in this entry. Note that a control index cannot appear more than k times in one round, but the value of $(x_r + 1)$ in the computing process may be larger than k. In this case the second element of the entry is set to *no*.

Example 2. The control table of the tree with cut level $l = 4$ in Fig. 1 is showed in Table 4.

Table 4. The Control Table of the Example Data Set

$B_1^{1[1]}$	no no	$B_2^{1[1]}$	no no 8 $B_1^{1[2]}$	$B_3^{1[1]}$	no no 8 $B_1^{1[2]}$ 3 $B_2^{1[2]}$	$B_4^{1[1]}$	no no 8 $B_1^{1[2]}$ 3 $B_2^{1[2]}$ 2 $B_3^{1[2]}$	$B_4^{1[2]}$	1 $B_1^{1[1]}$ 8 $B_1^{1[2]}$ 3 $B_2^{1[2]}$ 2 $B_3^{1[2]}$	$B_3^{1[2]}$	2 $B_1^{1[1]}$ 8 $B_1^{1[2]}$ 3 $B_2^{1[2]}$	$B_2^{1[2]}$	3 $B_1^{1[1]}$ 8 $B_1^{1[2]}$	$B_3^{2[1]}$	3 $B_1^{1[1]}$ 8 $B_1^{1[2]}$ 8 $B_1^{1[2]}$
$B_4^{3[1]}$	3 $B_1^{1[1]}$ 8 $B_1^{1[2]}$ 8 $B_1^{1[2]}$ 7 $B_3^{2[2]}$	$B_4^{3[2]}$	6 $B_1^{1[1]}$ 8 $B_1^{1[2]}$ 8 $B_1^{1[2]}$ 7 $B_3^{2[2]}$	$B_5^{2[2]}$	7 $B_1^{1[1]}$ 8 $B_1^{1[2]}$ 8 $B_1^{1[2]}$	$B_1^{1[2]}$	8 $B_1^{1[1]}$	$B_2^{2[1]}$	8 $B_1^{1[1]}$ 16 no	$B_3^{3[1]}$	8 $B_1^{1[1]}$ 16 no 12 $B_2^{2[2]}$	$B_3^{3[2]}$	9 $B_1^{1[1]}$ 16 $B_1^{1[2]}$ 12 $B_2^{2[2]}$	$B_4^{6[1]}$	9 $B_1^{1[1]}$ 16 no 12 $B_2^{2[2]}$ 12 $B_2^{2[2]}$
$B_4^{6[2]}$	11 $B_1^{1[1]}$ 16 no 12 $B_2^{2[2]}$ 12 $B_2^{2[2]}$	$B_2^{2[2]}$	12 $B_1^{1[1]}$ 16 no	$B_3^{4[1]}$	12 $B_1^{1[1]}$ 16 no 16 no	$B_4^{7[1]}$	12 $B_1^{1[1]}$ 16 no 16 no 14 $B_3^{4[2]}$	$B_4^{7[2]}$	16 $B_1^{1[1]}$ 16 no 16 no 14 $B_3^{4[2]}$	$B_3^{4[2]}$	16 $B_1^{1[1]}$ 16 no 16 no	$B_3^{5[1]}$	14 $B_1^{1[1]}$ 16 no 16 no 16 no	$B_4^{8[1]}$	15 $B_1^{1[1]}$ 16 no 16 no 16 no

Look at the control table of $B_4^{6[2]}$ in the lower left corner of the table. Since it is the second appearance of B_4^6, all the keys less than or equal to 11 have been broadcasted. Thus if the searching key is less than 11, we jump to $B_1^{1[1]}$. The second entry means that, if the search key is larger than 16, then there is

no where to go since the largest key is 16. Suppose that we want to search the key value of 14,then the next hop is $B_2^{2[2]}$, which is exactly the nearest ancestor dominating 14.

From the control table in Table 4, we find that some control tables, marked as gray, have redundancy. For example, the control table of $B_3^{4[1]}$, there are two identical entry *16 no*. When the tree becomes large, the redundancy will be as big as half of the whole control index, which will waste lots of resource. As the next section shows, almost *half* of the control tables contains redundancy, so we will save half of the space if we can eliminate those redundancy.

5.2 Redundancy Elimination for Control Table

In this section, we propose a scheme to eliminate redundancy. We claim that our scheme is not only suitable for AH-Tree, but also for any other tree-based indices employing distributed technique. For the balance tree, the scheme can save 50% of the space for storing the control tables.

Having redundancy in the control table means that for two entries A and B in the control table, the key value of them are identical. Formally speaking, suppose the i^{th} and j^{th} entries are redundant in $B_i^{j[x]}$'s control table. It means that the upper bound of the dominating range of i^{th} and j^{th} ancestor of $B_i^{j[x]}$ are the same. Recall that during the process of searching the control table, after we find the key value of some entry less than the searching key, we jump immediately. The following entries with the same key value will never be used. So we can simply discard these following entries. The problem is how to locate the redundant control tables. A simple case is the rightmost path of the tree.

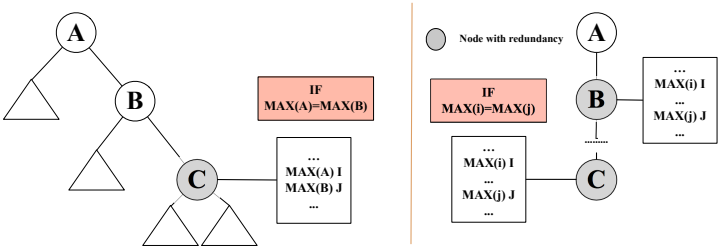


Fig. 2. Example of Redundancy in Control Table

The upper bound of the dominating range of the control index along this path are all the same. Consider the left part of Fig. 2. Since B is the rightmost child of A, we have $\mathrm{MAX}(A) = \mathrm{MAX}(B)$. Thus the control table of C contains redundancy. Another case maybe less obvious. The control table of a control index also contains redundancy if one of its ancestors' control table contains redundancy, as shown in the right part of Fig. 2. However, all those causes can be combined into Theorem 2. Before that, we first give two properties of the control table without proof.

Property 1: If index node A and B are ancestors of C, and A is ancestor of B, then $\mathrm{MAX}(A) \geq \mathrm{MAX}(B) \geq \mathrm{MAX}(C)$, and in the control table of C, the entry of A is in front of that of B.

Property 2: If $\mathrm{MAX}(A) = \mathrm{MAX}(B)$ and B is a child of A, then B must be the rightmost child of A and vice versa.

Theorem 2. Let $\mathrm{PATH}'(B_i^{j[x]}) = [B_2^{j_2}, B_3^{j_3}, \ldots, B_i^{j_i}]$, which is $\mathrm{PATH}(B_i^{j[x]})$ plus $B_i^{j[x]}$ then excludes the root of the tree. The control table of $B_i^{j[x]}$ contains redundancy if and only if there exits consecutive index nodes $B_{a-1}^{j_{a-1}}$ and $B_a^{j_a}$ in $\mathrm{PATH}'(B_i^{j[x]})$, such that $B_a^{j_a}$ is the rightmost child of $B_{a-1}^{j_{a-1}}$.

Proof. $\Rightarrow$. Suppose that $B_{i_1}^{j_{i_1}}$ and $B_{i_2}^{j_{i_2}}$ belong to $\mathrm{PATH}'(B_i^{j[x]})$ and $\mathrm{MAX}(B_{i_1}^{j_{i_1}}) = \mathrm{MAX}(B_{i_2}^{j_{i_2}})$ in the control table of $B_i^{j[x]}$. Without loss of generality, we assume $i_1 < i_2$, so $B_{i_1}^{j_{i_1}}$ is an ancestor of $B_{i_2}^{j_{i_2}}$.
(a) If $i_1 + 1 = i_2$, then $B_{i_2}^{j_{i_2}}$ is a child of $B_{i_1}^{j_{i_1}}$. Since $\mathrm{MAX}(B_{i_1}^{j_{i_1}}) = \mathrm{MAX}(B_{i_2}^{j_{i_2}})$, by Property 2 $B_{i_2}^{j_{i_2}}$ is the rightmost child. Then $B_a^{j_a}$ is $B_{i_2}^{j_{i_2}}$, $B_{a-1}^{j_{a-1}}$ is $B_{i_1}^{j_{i_1}}$.
(b) If $i_1+1 < i_2$, then $i_1 < i_1+1 < i_2$, which implies $\mathrm{MAX}(B_{i_1}^{j_{i_1}}) \geq \mathrm{MAX}(B_{i_1+1}^{j_{i_1+1}}) \geq \mathrm{MAX}(B_{i_2}^{j_{i_2}})$ by Property 1. However, $\mathrm{MAX}(B_{i_1}^{j_{i_1}}) = \mathrm{MAX}(B_{i_2}^{j_{i_2}})$, so $\mathrm{MAX}(B_{i_1}^{j_{i_1}}) = \mathrm{MAX}(B_{i_1+1}^{j_{i_1+1}})$. We conclude that $B_{i_1+1}^{j_{i_1+1}}$ must be the rightmost child of $B_{i_1}^{j_{i_1}}$ by Property 2. Let $B_a^{j_a}$ be $B_{i_1+1}^{j_{i_1+1}}$ and $B_{a-1}^{j_{a-1}}$ be $B_{i_1}^{j_{i_1}}$.
$\Leftarrow$. Since $B_a^{j_a}$ is the rightmost child of $B_{a-1}^{j_{a-1}}$, we have $\mathrm{MAX}(B_a^{j_a})$=$\mathrm{MAX}(B_{a-1}^{j_{a-1}})$, then there are redundancy in the control table. □

By traveling the tree, we can find all the redundant control tables with the help of the above theorem. Then we eliminate all the redundant entries but the one with least level. Finally we get a "clear" control tables.

6 Index and Data Allocation

After adding control tables to the index nodes, the final step is to allocate them onto channels. In this section, we present the scheme of index and data allocation. There are many index allocation method, such as [4,16,18], but they didn't employ distributed technique. In this paper, we use the simple pyramid scheduling scheme to allocate data and indices onto channels.

We define $W(B_i^{j[x]})$ as the weight of the index node $B_i^{j[x]}$, which denotes the sum of the probability of all its data descendants, while $W(D_i)$ is the access probability of data D_i. Since we apply the same method to allocate index and data, we only describe data allocation in the paper, which is identical to index allocation method. The algorithm of data allocation is shown in Alg. 2.

There is a dynamic threshold for each index channel. In Alg. 2, recall that n is the number of index channels. Suppose that the sum of all the weight is SUM, then the threshold of the first channel is SUM/n. Thus, we assign indices to the first index channel one by one until the total weight of all these assigned

Algorithm 2. Data Allocation on Multiple Channels

Input: W, the weight set of the data; n, the number of the data channels;
Output: $C = \{C_1, C_2, \ldots, C_n\}$, the generated data channel set.

Sort the data set by frequency in ascending order, results in $I = \{D_1, D_2, \ldots, D_t\}$;
$SUM = \sum_{i=1}^{t} W(D_i)$;
Set $ave = 0$; $p = SUM$; $thre = \frac{SUM}{n}$; $C_i = \emptyset$; $j = 1$;
for $i = 1$ to t **do**
  if $ave \leq thre$ **then**
    $ave = ave + W(D_i)$;
    $C_j = C_j \cup \{D_i\}$;
  else
    $p = p - ave$; $ave = 0$; $thre = \frac{p}{n-j}$; $j{+}{+}$; $i{-}{-}$;
  end if
end for

indices exceeds the threshold of the first channel. Next, we begin to assign the remaining indices to the next channel. Instead of having the same threshold for all the channels, we set the threshold of next channel as

$$\frac{\text{total weight of the remaining indices}}{\text{the number of remaining index channels}}$$

for fairness. Then we repeat the above process until no index is left.

Example 3. We use the data set in sec.4 and apply the allocation scheme to it. Suppose there are 4 data channels.

(1) In the first iteration, the threshold $thre = (16 + 8 + 30 + 4 + 1 + 12 \ldots + 23 + 1)/4 = 62$. We assign the data one by one to the first channel until the sum of frequencies of assigned data exceeds 62.This ends the assignment of the first channel and the first 6 data are assigned to it.
(2) We calculate the threshold of the second channel, $thre = (27 + 36 + 41 + \ldots + 23 + 1)/3 = 60$, which the assigned data and channel are not considered any more. We assign the data one by one to the second channel until the sum of frequencies of assigned data exceeds 60.

We apply this process until all the data have been assigned. The final allocation is: $1 : \{1, 2, \ldots, 5, 6\}, 2 : \{7, 8, \}, 3 : \{9, 10, 11, 12\}, 4 : \{13, 14, 15, 16\}$.

7 Simulation

In this section, we evaluate the performance of AH-Tree based system in different conditions. We conduct the performance evaluation on k, l, and m.

Firstly, we model different system conditions by setting all parameters in Table 5. We let the frequency of each data follow Zipf distribution. The number data items are set as 10,000 and we generate 20,000 clients requests in

our experiments. Performance of our system is mainly measured by two metrics: *Average Access Latency* (AAT) and *Average Tuning Time* (ATT), both of which are counted in *logical time units*. As proposed in [12], each logical time unit represents the time required to broadcast 1KB data. For index bucket with 1 head segment, k children pointers and 1 default pointer, the size equals to $(k+2)*0.1$KB, that is to say, it requires $(k+2)*0.1$ time units to visit. Our experiments are conducted on a computer with Intel(R) Core(TM) i7-3610QM (2.30GHz) CPU and 4.00 GB memory under Windows 7 version 6.1. The simulator is implemented in Java 1.7.005.

Table 5. Parameters used in our experiments

Parameter	Default value	Range	Meaning
t	10,000		Number of data items
r	20,000		Number of requests
A	10		Number of available channels
k	3	2 to 20	k-ary AH-Tree
l	3	1 to (L-1)	cut level, L is the height of tree
m	3	1 to 9	Number of index channels

Secondly, we verify the effectiveness of our index-allocation algorithm, namely, the pyramid scheduling scheme. We use two different allocation algorithms: the *McNaughton's Wrap-Around* algorithm which simply allocates indices evenly onto index channels, and our pyramid scheduling algorithm, then compare their respective performances. From Fig. 3 to Fig. 8, we can conclude that (1). Pyramid index allocation achieves better performance than simple wrap-around algorithm, and (2). the AAL and ATT of our system have closer relationship with parameter settings of k, l, m than the adopted allocation scheme, because the shapes of the lines look similar with different allocation schemes.

Thirdly, we compare the performance of AH-Tree based system with B^+-Tree based one. Both of the two systems adopt pyramid index allocation algorithm. Generally, Fig. 9 to Fig. 14 reveal that AH-Tree performs better than B+-Tree on both access latency and tuning time for data following Zipf distribution. In Fig. 10, the ATT declines with the increase of l since more control indices contribute to faster hopping to targeted data. In the meanwhile, the AAL in Fig. 9 firstly drops then rises again since too much control indices make the index sequences in index channels too long, thus lengthen the access latency. Fig. 11 clearly shows that large k has negative effects on AH-Tree's performance and the AAL of AH-Tree based system fluctuates severely with the variation of k. Therefore, choosing a proper k for AH-Tree based system is crucial for its user experience. For Zipf distribution, the access latency is mainly determined by the waiting time for small number of frequently visited indices, so we find that larger m leads to the decrease of AAL in Fig. 7. On the other hand, the change of m does not change the structure of index sequence, hence it has no effects on ATT as shown in Fig. 8.

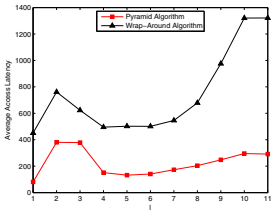


Fig. 3. Change of AAL under Zipf distribution w.r.t. l ($k = 3$, $m = 3$)

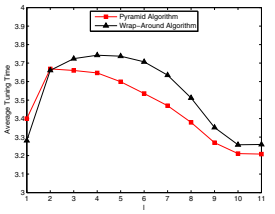


Fig. 4. Change of ATT under Zipf distribution w.r.t. l ($k = 3$, $m = 3$)

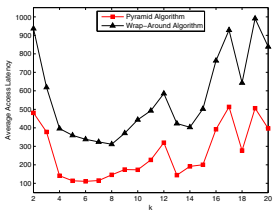


Fig. 5. Change of AAL under Zipf distribution w.r.t. k ($l = 3$, $m = 3$)

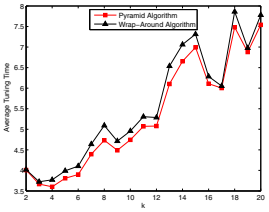


Fig. 6. Change of ATT under Zipf distribution w.r.t. k ($l = 3$,$m = 3$)

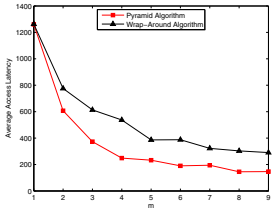


Fig. 7. Change of AAL under Zipf distribution w.r.t.m ($k = 3$, $l = 3$)

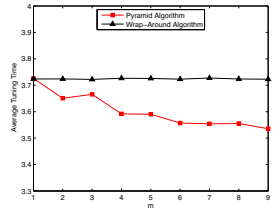


Fig. 8. Change of ATT under Zipf distribution w.r.t.m ($k = 3$, $l = 3$)

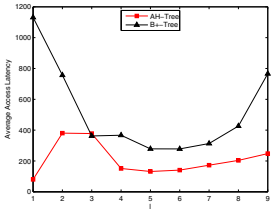


Fig. 9. Change of AAL under Zipf distribution w.r.t. l ($k = 3$,$m = 3$)

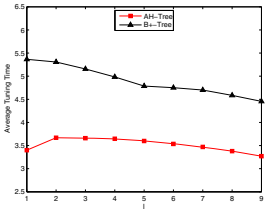


Fig. 10. Change of ATT under Zipf distribution w.r.t. l ($k = 3$,$m = 3$)

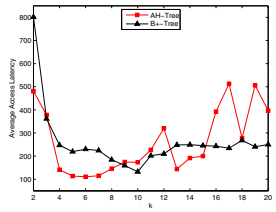


Fig. 11. Change of AAL under Zipf distribution w.r.t. k ($l = 3$,$m = 3$)

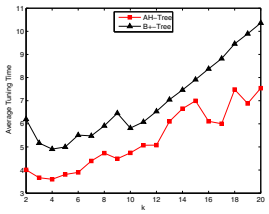


Fig. 12. Change of ATT under Zipf distribution w.r.t. k ($l = 3$,$m = 3$)

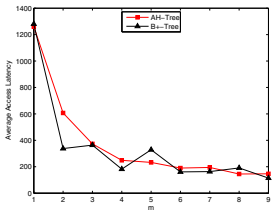


Fig. 13. Change of AAL under Zipf distribution w.r.t.m ($k = 3$, $l = 3$)

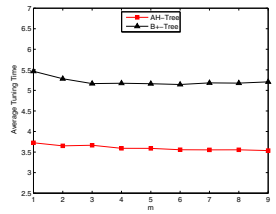


Fig. 14. Change of ATT under Zipf distribution w.r.t.m ($k = 3$, $l = 3$)

8 Conclusion

In this paper, we propose an efficient AH-Tree construction with the help of dynamic programming. Our algorithm can build arbitrary k-ary AH-Tree index with bounded tree height in $O(kn^2)$ time, where n is the number of data items to be broadcast. We then modify this tree into a distributed index sequence with a general and effective control-table shrinking technique to further reduce the index length and the client searching time. We prove the correctness and complexity of our design theoretically and illustrate the performance of the broadcast system by numerical experiments. Both theoretical proofs and simulation results validate the efficiency of our design. To the best of our knowledge, we are the first to propose the detailed design for k-ary Hu-Tucker AH-Tree construction with time complexity analysis, which solves a long-time open question since the beginning of twenty-first's century.

References

1. Hu, T., Tucker, A.: Optimal computer search trees and variable-length alphabetical codes. SIAM Journal on Applied Mathematics 21(4), 514–532 (1971)
2. Yao, Y., Tang, X., Lim, E., Sun, A.: An energy-efficient and access latency optimized indexing scheme for wireless data broadcast. IEEE Trans. on Knowledge & Data Engineering 18(8), 1111–1124 (2006)
3. Imielinski, T., Viswanathan, S., Badrinath, B.: Data on air: Organization and access. IEEE Trans. on Knowledge & Data Engineering 9(3), 353–372 (1997)
4. Jung, S., Lee, B., Pramanik, S.: A tree-structured index allocation method with replication over multiple broadcast channels in wireless environments. IEEE Trans. on Knowledge & Data Engineering 17(3), 311–325 (2005)
5. Xu, J., Lee, W., Tang, X., Gao, Q., Li, S.: An error-resilient and tunable distributed indexing scheme for wireless data broadcast. IEEE Trans. on Knowledge & Data Engineering 18(3), 392–404 (2006)
6. Adamic, L., Huberman, B.: Zipf's law and the internet. Glottometrics 3(1), 143–150 (2002)
7. Zhong, J., Wu, W., Gao, X., Shi, Y., Yue, X.: Efficient redesign and comparison of various indexing schemes for wireless data broadcasting. Submitted to Knowledge and Information Systems (2012)
8. Shivakumar, N., Venkatasubramanian, S.: Efficient indexing for broadcast based wireless systems. ACM Journal of Mobile Networks and Applications 1(4), 433–446 (1996)
9. Zhong, J., Wu, W., Shi, Y., Gao, X.: Energy-Efficient Tree-Based Indexing Schemes for Information Retrieval in Wireless Data Broadcast. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part II. LNCS, vol. 6588, pp. 335–351. Springer, Heidelberg (2011)
10. Zhong, J., Gao, Z., Wu, W., Chen, W., Wang, L.: Multi-channel energy-efficient hash scheme broadcasting. SEDE, Los Angeles (2012)
11. Gao, X., Shi, Y., Zhong, J., Zhang, X., Wu, W.: Sambox: A smart asynchronous multi-channel blackbox for b+-tree based data broadcast system under wireless communication environment. SEDE, Los Angeles (2012)

12. Lu, X., Gao, X., Yang, Y.: Setmes:a scalable and efficient tree-based mechanical scheme for multi-channel wireless data broadcast. Submitted to ACM ICUIMC (2013)
13. Sun, W., Liu, P., Wu, J., Qin, Y., Zheng, B.: An automaton-based index scheme for on-demand XML data broadcast. In: Lee, S.-g., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part II. LNCS, vol. 7239, pp. 96–110. Springer, Heidelberg (2012)
14. Vlajic, N., Charalambous, C., Makrakis, D.: Wireless data broadcast in systems of hierarchical cellular organization. In: IEEE International Conference on Communications, ICC 2003, vol. 3, pp. 1863–1869 (2003)
15. Yee, W., Navathe, S.: Efficient data access to multi-channel broadcast programs. In: Proceedings of the 12th International Conference on Information and Knowledge Management, pp. 153–160 (2003)
16. Wang, S., Chen, H.: Tmbt: An efficient index allocation method for multi-channel data broadcast. In: Advanced Information Networking and Applications Workshops, AINAW 2007, vol. 2, pp. 236–242 (2007)
17. Hu, T.: A new proof of the tc algorithm. SIAM Journal on Applied Mathematics 25(1), 83–94 (1973)
18. Lo, S., Chen, A.: Optimal index and data allocation in multiple broadcast channels. In: 16th International Conference on Data Engineering, ICDE 2000, pp. 293–302 (2000)

MVP Index: Towards Efficient Known-Item Search on Large Graphs

Ming Zhong[1], Mengchi Liu[2], Zhifeng Bao[3], Xuhui Li[1], and Tieyun Qian[1]

[1] State Key Laboratory of Software Engineering, Computer School, Wuhan University, Wuhan 430072, China
{clock,lixuhui,qty}@whu.edu.cn
[2] School of Computer Science, Carleton University, Ottawa K1S5B6, Canada
mengchi@scs.carleton.ca
[3] School of Computing, National University of Singapore, Singapore 117417
baozhife@comp.nus.edu.sg

Abstract. This paper is motivated by the lack of study on the diversity of user information needs in the scenario of graph search, which offers the prospect of significant improvements on search. We report our investigation on this issue, and then exploit the knowledge to optimize a commonly-used type of graph search: *known-item search* which only wants the answer trees of a familiar and compact pattern. To address the problem, we propose a novel MVP (Matched Vertex Pruning) index, which captures the query-independent local connectivity information in the graph, to reduce the search space with heuristics by pruning matched vertices that will not participate in the answer trees with heights less than a threshold. Moreover, our optimization approach is independent of search algorithm, and requires the minimal index access times. Our experimental results show that our approach can generally reduce the number of matched vertices to 1%-10%, thereby effectively improving the efficiency of the known-item search.

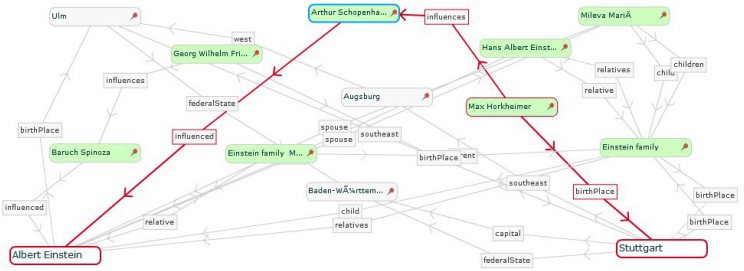
1 Introduction

The importance of understanding why users are performing their searches has already been realized by the traditional search engine community for years [2, 12]. Researchers propose a "taxonomy of web search" that classifies daily user queries into some common search types, such as navigational search and informational search. Now, all of the major web search engines try to detect which type of search the user is performing, to better accommodate common user behaviors in their search results pages.

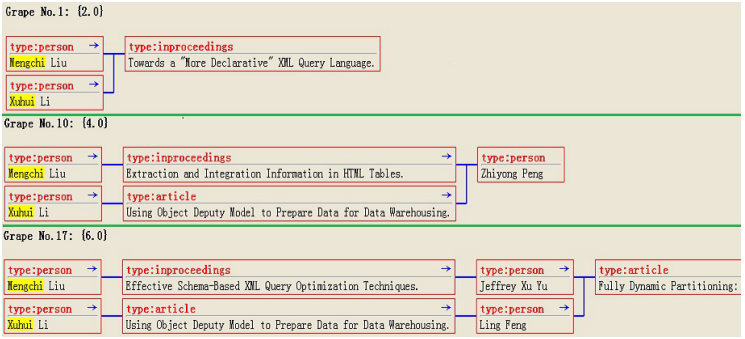
However, to the best knowledge we have, there is still a lack of study on search types in the scenario of graph search. By our observation, graph search reflects an even more diverse set of underlying information needs, and the knowledge of those needs offers the prospect of significant improvements. More importantly, for graph search, the potential improvements are not only on the presentation of the search results, but also on the optimization of the search process. Let us consider the following two types of graph search, which we call *exploratory search* and *known-item search* respectively.

Example 1 (Exploratory Search). Consider a keyword query "albert einstein stuttgart". It intends to discover the associations between the scientist Albert Einstern and the German city Stuttgart. Obviously, these two entities have no direct connections in common

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 193–200, 2013.



(a) Exploratory search on DBPedia.



(b) Known-item search on DBLP.

Fig. 1. Two typical search types on graph

sense. For that, the user likely wants to explore the whole graph to find as many relevant results as possible and manually browse the results. Figure 1(a) illustrates a variety of paths connecting Albert Einstern and Stuttgart in the DBPedia[1] graph. For example, the highlighted path indicates: Stuttgart is the birthplace of Max, Max influences Arthur, who is influenced by Albert Einstern.

In sharp contrast to the exploratory search, the known-item search has already some answers of a particular meaningful and frequently-used pattern in mind.

Example 2 (Known-Item Search). Consider a keyword query "mengchi xuhui". It intends to locate the paper coauthored by Mengchi and Xuhui, or the person who is a coauthor of both of them. Figure 1(b) illustrates several answer trees (where the vertices on the left are the leaves) in the DBLP[2] graph, where "person" and "paper" entities are represented as vertices and "write" and "cite" relationships between them are represented as edges. The answers with label Grape No.1 and No.10 are exactly of the target pattern. The Grape No.1 shows a paper coauthored by Mengchi Liu and Xuhui Li, and the Grape No.10 indicates that Zhiyong Peng is a coauthor of both of them.

These two search types will lead to quite different processing and optimization strategies (see details in Section 2). The known-item search does not need to explore the

[1] http://dbpedia.org/
[2] http://www.informatik.uni-trier.de/~ley/db/

whole graph for unfamiliar large and complex answer patterns. It is promised that the efficiency of the known-item search can be remarkably improved, especially on large graphs, compared with the general graph search (i.e., naive exploratory search). On the other hand, a large proportion of real world queries are known-item search, at least 20%-30% by statistics, according to the experiences gained from traditional search engines [2]. However, existing approaches (e.g., [1, 3–7, 11]) has not yet taken advantage of this valuable feature. As a result, optimizing the known-item search is a critical task.

In this paper, we propose a novel indexing technique for optimizing the known-item search. We summarize the contributions of this paper as follows.

- We initiate the problem of studying the user information needs for graph search.
- We propose a practical matched vertex pruning strategy for optimizing the known-item search on graph, which is independent of search algorithm.
- We present an indexing technique that captures the query-independent local connectivity information in the graph for supporting fast online pruning, which requires the minimal index access times for processing a query.
- We run experiments to evaluate the pruning effect. By observation, our approach can significantly reduce the search space on large graphs.

2 Diversity of Graph Search

Keyword search on graph reflects a diverse set of underlying information needs. The knowledge of the needs offers the prospect of significant improvements. Achieving the improvements involves three primary tasks. First, we need to know different information needs. Second, we need a way to associate information needs with queries. Third, we need to extend the search engines in order to better accommodate users' intents.

To investigate users' real information needs, we conducted a survey in the 29th National Database Conference of China held in 2012. Attendees could use a state-of-the-art graph search engine to search on the DBLP graph. By interviewing the attendees, we summarized the following two typical search types, which could represent most of the needs.

- **Exploratory Search**. By issuing an exploratory search, the user is either unfamiliar with the underlying graph structures, unsure about what patterns of answer trees they need, or both. So, the user's goal is to find as various answer trees as possible, so that they can learn something by reading and comprehending the search results.
- **Known-Item Search**. By issuing a known-item search, the user's target is a or a set of answer trees of a particular pattern, either because the user has seen them before or because the user assumes them exist by prior experience.

We do not address the automatic identification of search types in this paper. There are some similar studies on web search, such as [8]. Our focus is how to exploit the knowledge of information needs to improve graph search. Exploratory search aims to find rich results of various patterns, which means a complete search over the graph. This is nearly unfeasible on very large graphs, since the computational complexity and memory usage of graph search is extremely high. Therefore, our opinion is that exploratory search

should take into consideration the parallel computation paradigms like Pregel [10] and Trinity [13], or the novel query mechanism that operates in a "semantic pattern first, entity subgraph later" way [14].

Unlike the exploratory search, the known-item search usually focuses on meaningful and familiar answer patterns with limited size. When an answer tree has a relatively large size (which depends on the number of keywords in the query), it is not likely to be a so-called known-item. Because the relationships between the entities in the answer are loose, so that the pattern cannot be very meaningful to and frequently-used by the user. For example, consider the previous example illustrated in Figure 1(b). Besides the Grape No.1 and No.10, there are also other larger answers like the Grape No.17 can be found by keeping on searching on the graph. The Grape No.17 shows a more complicated pattern, which may be interested by some users but is not possible to be frequently queried by a known-item search.

Therefore, the known-item search does not need to explore the whole graph for unfamiliar large and complex answer trees. It means that we could avoid producing an answer tree if we can predicate that it will grow too large. This is different from the top-k search, which can only limit the number of returned answers but can not limit the size of them directly. Inspired by this idea, we propose an effective optimization approach for the known-item search in the next section.

3 Matched Vertex Pruning

3.1 Preliminary

Let $\mathcal{G}$ be a vertex-labeled undirected weighted graph and $\mathcal{Q}$ be a query composed of a set of keywords. For a vertex v, if its label contains the keyword $t \in \mathcal{Q}$, v is called a *matched vertex* of keyword t. Let $mv(t)$ denote the set of matched vertices of t in $\mathcal{G}$. The answers to $\mathcal{Q}$ are *minimum connected trees*, which are subtrees of $\mathcal{G}$ containing at least one matched vertex of each keyword in $\mathcal{Q}$. Moreover, any proper subtrees of them do not contain (the matched vertices of) all keywords. For an answer tree a, let $root(a)$ be its root, $rkpath(a, t)$ be the root-keyword path between $root(a)$ and the nearest matched vertex of keyword t in a, and $|rkpath(a, t)|$ be the length of the path.

Generally, compacter answer trees include more relevant information. To support efficient top-k search, we follow [5] to score (the compactness of) an answer tree a as the sum of the lengths of all its root-keyword paths.

Consider the running example illustrated in Figure 2. Figure 2(a) illustrates an example graph, where the vertices containing the terms a, b, c are marked. Figure 2(c) illustrates the best (most compact) answer tree to the query in Figure 2(b). The score of this answer tree is 0.8 + 0.8 + 1.5 = 3.1. Note that, the lower score the better answer.

3.2 Our Approach

Since the known-item search only wants the compact enough answer trees, the basic idea of our optimization approach is to prune the matched vertices that will not be

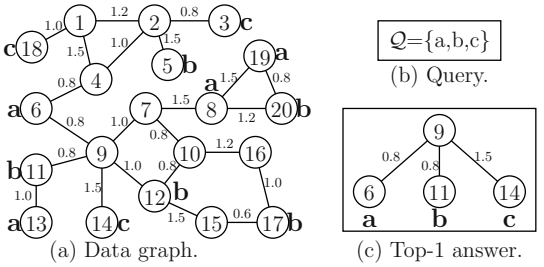


Fig. 2. A running example

contained by such answer trees to a given query before search, so that the unproductive graph traversals started from the pruned matched vertices can be avoided, thereby decreasing response time.

While, whether an answer tree is compact enough is a relative concept, which depends on the underlying graph and the user. Thus, the measurement should be tunable with regards to different circumstances. Moreover, we need to be able to know before search if a matched vertex is contained by an answer tree satisfying the measurement demand. Or, it is unfeasible to track and prune those matched vertices. It is the reason why we cannot just choose the score of answer tree as measurement, because finding the answer trees under a specific score is an agnostic problem before search. For that, we heuristically choose the height of answer tree as the measurement and give the formal definition of δ-*ceiling answer tree* below.

Definition 1 (**δ-Ceiling Answer Tree).** *For an answer tree a to a query $\mathcal{Q}$, its height is the length of its longest root-keyword path. The answer tree is called a δ-ceiling answer tree if and only if its height is at most δ, namely,* $\max\{|rkpath(a,t)|| \ t \in \mathcal{Q}\} \leqslant \delta$.

Given a graph $\mathcal{G}$, a keyword query $\mathcal{Q}$ and the expected maximum answer tree height δ, we call the matched vertices contained by the δ-ceiling answer trees to $\mathcal{Q}$ in $\mathcal{G}$ as *seeds*. So, our pruning target is the matched vertices other than seeds.

To identify the seeds, we do not have to really find the δ-ceiling answer trees. Instead, we fulfill that by locating their roots. Let us see the following concepts.

Definition 2 (**δ-Neighborhood).** *For each vertex $v \in \mathcal{V}$, the δ-neighborhood of v, denoted as $\Delta(v)$, is a set of vertices in $\mathcal{G}$ that can be connected to v by a path no longer than δ. In particular, the δ-neighborhood of a vertex includes this vertex itself.*

Definition 3 (Potential Root). *For each vertex $v \in \mathcal{V}$, v is a potential root of a keyword $t \in \mathcal{Q}$, if and only if v belongs to the δ-neighborhood of a matched vertex of t, namely, $\exists u \in mv(t)$ s.t. $v \in \Delta(u)$. We use $pr(t)$ to denote the set of potential roots of t.*

With above definitions, the following theorem holds. Its proof is omitted.

Theorem 1. *For a vertex $v \in \mathcal{V}$, if v is a potential root of each keyword in $\mathcal{Q}$, namely, $v \in \bigcap_{t \in \mathcal{Q}} pr(t)$, then v is the root of a δ-ceiling answer tree to $\mathcal{Q}$, and vice versa.*

Intuitively, Theorem 1 indicates how to locate the roots of the δ-ceiling answer trees to a specific query by using the δ-neighborhoods and the potential roots. Consider the

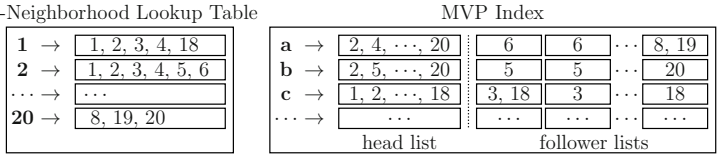


Fig. 3. Index structures

running example illustrated in Figure 2 again. Given $\delta = 2.0$, the potential roots of each keyword in the query are listed below.

pr(a): 2 4 6 7 8 9 11 12 13 19 20
pr(b): 2 5 6 7 8 9 10 11 12 13 15 16 17 19 20
pr(c): 1 2 3 4 9 14 18

Thus, the roots of the δ-ceiling answer trees to the query are 2 and 9. Then, we can identify the seeds of the query: the matched vertices whose δ-neighborhoods contain the roots. We compare the original matched vertices and the seeds below.

All matched vertices: 3 5 6 8 11 12 13 14 17 18 19 20
Seeds: 3 5 6 11 12 13 14

We successfully reduce the number of matched vertices from 12 to 7. In real graphs, seeds are usually a much smaller subset of original matched vertices (see our experiments in Section 5), conforming to our expectation of the known-item search.

A major advantage of our pruning strategy is its search-independence. All of the task can be fulfilled before search. Thus, current graph search algorithms and index techniques can still be used after the matched vertex pruning. It guarantees that the performance of the known-item search can be improved.

4 MVP Index

In this section, we present an MVP (Matched Vertex Pruning) index for implementing our pruning strategy quickly in query time. As illustrated in Figure 3, our index system has the following two index structures.

δ-Neighborhood lookup table is used for looking up the δ-neighborhood of a specific vertex. A δ-neighborhood is physically represented as a sorted list of vertex ids (vids). The lookup table is constructed in advance of the MVP index.

MVP index is like an inverted index. The index entry of a term t is composed of a *head list* and a number of *follower lists* organized in a particular order for efficient process. The head list consists of the vids of the potential roots of t, namely, is the sorted $pr(t)$. Following the head list, there are $|pr(t)|$ corresponding follower lists. The i-th follower list consists of the vids of the matched vertices of t in the δ-neighborhood of the i-th vertex in the head list.

For example, given δ = 2.0, the index entry of term c in the MVP index of the running example graph is $\{1, 2, 3, 4, 9, 14, 18\}$, $\{3, 18\}$, $\{3\}$, $\{3\}$, $\{3\}$, $\{14\}$, $\{14\}$, $\{18\}$. The follower list corresponding to the first vertex 1 in the head list is $\{3, 18\}$, which means the vertices 3 and 18 contain term c and belong to the δ-neighborhood of potential root 1.

Given a query $\mathcal{Q}$, we will get the index entries of each keyword in $\mathcal{Q}$ from the MVP index. Then, we can get the roots of δ-ceiling answer trees to $\mathcal{Q}$ by intersecting the head lists of all keywords in $\mathcal{Q}$ according to Theorem 1. The seeds of $\mathcal{Q}$ can be fetched from the follower lists corresponding to such roots in the head lists of each keyword. In the running example, the roots of δ-ceiling answer trees are 2 and 9, whose follower lists in the index entry of keyword c are $\{3\}$ and $\{14\}$ respectively. So, the set of seeds matched by keyword c is $\{3\} \cup \{14\} = \{3, 14\}$.

A very important advantage of our index is that it requires the minimal access times for query processing. By simply using a normal inverted index, we still need to access it once for each keyword in the query to initial the search, as many as our index. In contrast, current graph indexing techniques [5,9,11] will incur unlimited times of index access during search, and thus have to reside in memory for avoiding expensive disk IO, which is unfeasible on large graphs due to tremendous memory usage.

5 Experiment

Our experiments were run on the graphs generated from DBLP and IMDB[3]. The DBLP graph has about 0.83M vertices, 1.25M edges, and 4.30M terms in the labels. The IMDB graph has about 3.42M vertices, 13.06M edges, and 6.81M terms in the labels. The edges are assigned a weight in the range of [0.39, 0.99].

For both of the test graphs, we constructed their MVP indexes with $\delta = 2.0$. The time cost of indexing a large graph like IMDB is about 5 hours by using a basic indexing algorithm. If using a MapReduce environment, the time cost of indexing can be effectively reduced, according to our observation in [15].

Table 1 shows the matched vertex pruning results of several sample keyword queries we tested by using the MVP indexes. The 3rd column shows the numbers of matched vertices before pruning. We can see that most keywords have a large number of matched vertices. The 4th column shows the numbers of seeds (i.e., matched vertices remained after pruning), most of which are only about 1%-10% of the previous numbers. It is proved that our approach can reduce the search space significantly. In particular, the pruning effect of q1 is not as good as the others. That is because the keywords in q1 are

Table 1. The statistics of the matched vertex numbers before and after pruning

id	query	matched vertex #	seed #
DBLP			
q1	sensor network	2842, 12509	2342, 5714
q2	banks keyword search	199, 175, 4986	15, 26, 186
q3	michael daniel david eric	3758, 1699, 4011, 984	234, 151, 265, 136
q4	wang chang graph index update	2391, 1054, 3911, 833, 612	21, 20, 47, 82, 49
IMDB			
q5	marion batman nolan	978, 539, 374	54, 117, 35
q6	nicolas cage john travolta	1094, 255, 22330, 20	5, 9, 196, 11

[3] http://www.imdb.com

actually a phrase and they usually appear together in the labels, so that their matched vertices are always contained by δ-ceiling answer trees. In that case, the search is still fast without a lot of prunes, since not many traversals are needed to find the top-k.

6 Conclusion

In this paper, we reveal the diversity of user information needs in the scenario of keyword search on graph, and propose an MVP index to improve the search efficiency for queries driven by a particular type of needs called known-item search. The experimental results show that our approach can effectively reduce the search space.

Acknowledgments. This work was funded by National Natural Science Foundation of China (NSFC) under Grant No. 61202036 and Research Fund for the Doctoral Program of Higher Education from Chinese Ministry of Education under Grant No. 20120141120013, and has been supported partially by NSFC under Grant No. 61272110, 61272275 and 61202034.

References

1. Bhalotia, G., Hulgeriy, A., Nakhez, C., Chakrabarti, S., Sudarshan, S.: Keyword searching and browsing in databases using banks. In: ICDE, pp. 431–440 (2002)
2. Broder, A.: A taxonomy of web search. SIGIR Forum 36(2), 3–10 (2002)
3. Ding, B., Yu, J.X., Wang, S., Qin, L., Zhang, X., Lin, X.: Finding top-k min-cost connected trees in databases. In: ICDE, pp. 836–845 (2007)
4. Golenberg, K., Kimelfeld, B., Sagiv, Y.: Keyword proximity search in complex data graphs. In: SIGMOD, pp. 927–940 (2008)
5. He, H., Wang, H., Yang, J., Yu, P.S.: Blinks: Ranked keyword searches on graphs. In: SIGMOD, pp. 305–316 (2007)
6. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: Bidirectional expansion for keyword search on graph databases. In: SIGMOD, pp. 505–516 (2005)
7. Kimelfeld, B., Sagiv, Y.: Finding and approximating top-k answers in keyword proximity search. In: PODS, pp. 173–182 (2006)
8. Lee, U., Liu, Z., Cho, J.: Automatic identification of user goals in web search. In: WWW, pp. 391–400 (2005)
9. Li, G., Ooi, B.C., Feng, J., Wang, J., Zhou, L.: Ease: An effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In: SIGMOD, pp. 903–914 (2008)
10. Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: A system for large-scale graph processing. In: SIGMOD, pp. 135–146 (2010)
11. Markowetz, A., Yang, Y., Papadias, D.: Reachability indexes for relational keyword search. In: ICDE, pp. 1163–1166 (2009)
12. Rose, D.E., Levinson, D.: Understanding user goals in web search. In: WWW, pp. 13–19 (2004)
13. Sun, Z., Wang, H., Wang, H., Shao, B., Li, J.: Efficient subgraph matching on billion node graphs. PVLDB 5(9), 788–799 (2012)
14. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. In: ICDE, pp. 405–416 (2009)
15. Zhong, M., Liu, M.: A Distributed Index for Efficient Parallel Top-k Keyword Search on Massive Graphs. In: WIDM, pp. 27–32 (2012)

Indexing Reverse Top-k Queries in Two Dimensions

Sean Chester, Alex Thomo, S. Venkatesh, and Sue Whitesides

University of Victoria, Victoria, Canada
{schester,sue}@uvic.ca, {thomo,venkat}@cs.uvic.ca

Abstract. We consider the recently introduced monochromatic reverse top-k query which asks for, given a (possibly new) tuple q and a dataset $\mathcal{D}$, all possible top-k queries on $\mathcal{D} \cup \{q\}$ for which q is in the result. Towards this problem, we introduce the first *query-agnostic* approach, which leads to an efficient index. We present the novel insight that by representing the dataset as an arrangement of lines, a critical k-polygon can be identified and can singularly answer reverse top-k queries.

1 Introduction

In this age of arbitrarily large datasets, personalizing query results has become ubiquitous. A common approach is the traditional (linear) top-k query, which has been a staple of database systems for over a decade. (See the survey by Ilyas et al. [5].) In querying a dataset $\mathcal{D}$ of n numeric tuples $(a_1 \in \mathbb{R}, \ldots, a_d \in \mathbb{R})$, a top-$k$ query models the user with an ordered list of weights $(w_1, \ldots, w_d)$, representing his degree of "personal preference" for each of the d attributes of $\mathcal{D}$. In executing the query, each tuple $t \in \mathcal{D}$ is assigned a score, $\text{score}(t) = w_1 a_1 + \ldots + w_d a_d$, and the k tuples with highest score are presented.

In this paper, we consider these top-k queries from the perspective of the tuple rather than the user. A tuple $t \in \mathcal{D}$ is only relevant if it is the response to some top-k query. Its relevance is proportional to the breadth of queries for which it is returned. A *reverse* top-k (mRTOP) query [10] computes that breadth. Given a (possibly new) tuple of interest, q, a reverse top-k query reports the set of (traditional linear) top-k queries on $\mathcal{D} \cup \{q\}$ for which q is in the result set.

Table 1. *Top-k query example.* Shown is a dataset $\mathcal{D}$ of two fictitious basketball player tuples, $p1$ and $p2$, with two normalized attributes, points (pts_norm) and blocks (blks_norm). Also shown is an additional query tuple, $q = (0.725, 0.400)$. Two top-k queries are given in the rightmost columns, along with each tuple's score and rank.

			query a: $(0.75, 0.25)$		query b: $(0.25, 0.75)$	
pid	pts_norm	blks_norm	score	rank	score	rank
p1	0.333	1.000	0.500	3	0.833	1
p2	0.667	0.167	0.542	2	0.292	3
q	0.725	0.400	0.644	1	0.481	2

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 201–208, 2013.

Table 1 gives a small example of two traditional top-k queries, namely $a = (0.75, 0.25)$ and $b = (0.25, 0.75)$. Of these, only query a would be in the response to a *reverse top*-1 query on tuple q, because tuple q is only ranked among the best 1 tuples for query a, not for query b. Both queries would be in the response to a *reverse top*-2 (or top-3) query on q, because tuple q is ranked among the best 2 tuples for both queries.

Users can specify top-k queries from all of $\mathbb{R}^d$, so reverse top-k query solutions are infinite sets.[1] For the dataset in Table 1, the reverse top-1 query reports all traditional queries within $[(1.00, 0.00), (0.605, 0.395)]$, a range within which query a, but not query b, falls. We focus on the two dimensions and the positive quadrant. We do note, however, that the ideas we present generalize cleanly to all quadrants, an extension shown to be of significant interest by Ranu and Singh [7].

State of the Art. Reverse top-k queries are quite new and an example of the growing field of *reverse data management* [6]. As yet, there are two algorithms to answer mRTOP queries, the one originally proposed [10] and recently refined [11] by Vlachou et al., and a subsequent algorithm proposed by Wang et al. [12]. Both are linear-cost, two-dimensional algorithms. Both also have a common limitation, that their computation is heavily centred on and sensitive to the particular query tuple. The former compares every data tuple to the query tuple in terms of both pareto-dominance and a radial plane sweep. The latter achieves an experimental order-of-magnitude improvement by employing the geometric duality of Das et al. [3] to segment $\mathbb{R}^2$ and simultaneously computing the rank of the query tuple for each induced region.

Our Query-Agnostic, Index Approach. We term this common limitation *query-dependence*. In either algorithm, any computation done towards resolving one query is unusable for subsequent queries. The computation, involving in both algorithms a full table-scan, must be restarted from scratch. Furthermore, it cannot begin until the query is known. Our approach to the mRTOP problem, illustrated in Fig. 1, is to create an index on $\mathcal{D}$. We do this by employing *four* key geometric techniques: duality (using the transform of Chester et al. [2]), arrangements of lines (surveyed nicely by Sharir [9] in 1995), plane sweep (detailed well in the introductory text of de Berg et al. [1]), and data depth contours [4,8,13].

Our index, constructed *without knowledge of* q, can respond to many queries, each with only logarithmic cost. The general idea, shown in Fig. 1 (left), is to convert $\mathcal{D}$ into an arrangement of lines and identify the *critical* k*-contour*, a unique polygon, with a plane sweep algorithm. The index is a succinct representation of the polygon and each query is then equivalent to identifying the intersection of a query line l_q with that polygon, as illustrated in Fig. 1 (right).

Summary of Our Contributions. This paper makes several significant advances on the state of the art for reverse top-k queries. We:

[1] Vlachou et al. [10] define this problem as the *monochromatic* reverse top-k query, in contrast to the *bichromatic* reverse top-k query, but we will simply refer to it as a reverse top-k (mRTOP).

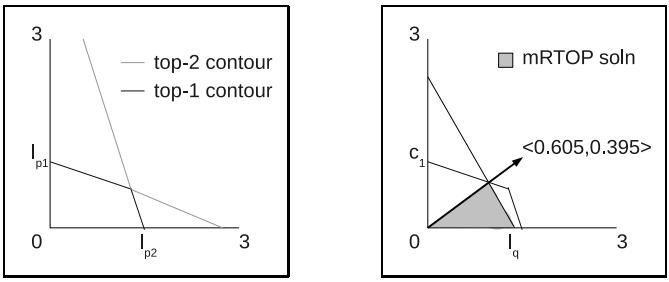


Fig. 1. *Illustration of our algorithm.* To the left, $\mathcal{D}$ is converted to an arrangement of lines by transforming each tuple $t = (a_1, a_2)$ into a line $a_1x + a_2y = \tau$, for an arbitrary, constant, positive real, τ (in this case, $\tau = 1$). The two contours are visible in different shades. To the right, the first contour is used to answer the reverse top-1 query on q.

- introduce the first query-agnostic approach. This allows computation to be reused or even done offline in advance. It also implies that the performance is consistent regardless of the query point. Previous work is sensitive both to the dataset *and* the choice of query point;
- introduce new geometric techniques that are themselves of high interest: our novel depth contours and k-polygons provide tools that are useful to researchers investigating related top-k and reverse top-k problems;
- and demonstrate consistently better empirical performance than existing algorithms by orders of magnitude.

2 Polygons, Contours, and Our Indexing Algorithm

Throughout all this work, we assume queries are executed on a two-dimensional, numeric relation $\mathcal{D}$ which is a set of tuples ($v_1 \in \mathbb{R}^+, v_2 \in \mathbb{R}^+$). Tuples can also alternatively be viewed as points (v_1, v_2) in the Euclidean plane or as two-dimensional vectors $\boldsymbol{v} = \langle v_1, v_2 \rangle$. We assume $|\mathcal{D}|$ is "large" and that k is a small constant, $k \in \mathbb{Z}^+ \ll |\mathcal{D}|$.

The *monochromatic reverse* top-k (mRTOP) query is a query tuple $q = (q_1, q_2)$ not necessarily in $\mathcal{D}$. The response is the set of traditional, linear top-k queries on $\mathcal{D} \cup \{q\}$ for which q is in the result set. Formally:

Definition 1. *The response to a* reverse top-k query, $q = (q_1, q_2)$, *is the set of angles*

$$mRTOP(q) = \{\theta \in [0, \pi/2] : |\{v \in \mathcal{D} : v_1 + v_2 \tan\theta > q_1 + q_2 \tan\theta\}| < k\}.$$

Our objective is to produce a succinct representation of $\mathcal{D}$ that encodes everything needed to respond efficiently to mRTOP queries in two dimensions.

2.1 The Critical k-Polygon, $\mathcal{P}_k$

We use the duality transform introduced by Chester et al. [2] to convert a set of tuples, $\mathcal{D}$, into a set of lines, $\mathcal{L}$. A tuple $v = (v_1, v_2)$ is interpreted as a vector $\boldsymbol{v} = \langle v_1, v_2 \rangle$ and then transformed to the line $l_{\boldsymbol{v}}$ with equation $y = \frac{\tau}{v_2} - \frac{v_1}{v_2}x$. The constant τ is an arbitrary but positive real. The line $l_{\boldsymbol{v}}$ is exactly the vector nullspace of $\boldsymbol{v}$, but translated into the positive quadrant. The score for tuple v for any weight vector lying on $l_{\boldsymbol{v}}$ is exactly τ. The transformation is illustrated in Fig. 1 for the two tuples given in Table 1, using $\tau = 1$.

Having used this duality transform, one can immediately infer the highest-ranked tuples for any user weight vector $\boldsymbol{w}$, because the rank of any line in the direction of $\boldsymbol{w}$ is precisely given by its position in the resultant arrangement of lines. In fact, for a ray emanating from O, the order in which lines are encountered exactly gives their ranks in the dataset. If one took together all line segments with equal rank, a star-shaped polygon would be formed, as indicated in Lemma 1:

Lemma 1. *A k-contour is a star-shaped polygon.*

Lemma 1 establishes that we can represent $\mathcal{D}$ as a set of polygons with a unique depth i, each of which itself encodes the i'th ranked tuple for any possible traditional, linear top-k query. If there is only one value k of interest, then the entire dataset can be represented just by one polygon, which we alternately call the k-contour or $\mathcal{P}_k$ of $\mathcal{D}$.

2.2 Properties of $\mathcal{P}_k$ and Its Convex Hull Approximation

There are some important properties of the k-polygon, including bounds on its size, that translate into bounds on asymptotic complexity. In particular, we show in Lemma 2 that if one created a new line from a new tuple, it would have a small number of intersection points with the polygon.

Lemma 2. *Any line l can intersect $\mathcal{P}_k$ at most k times.*

Furthermore, we can approximate $\mathcal{P}_k$ with its convex hull while maintaining bounds on the closeness of that approximation. Specifically, we have Lemma 3:

Lemma 3. *A concave region between vertices of the convex hull of the k-contour's upper boundary can have at most $2k - 1$ vertices.*

Consequently, we can transform the problem under study into one simpler:

Theorem 1. *The response to a mRTOP query, given query vector $\boldsymbol{q} = \langle q_1, q_2 \rangle$, is the component of $l_{\boldsymbol{q}}$ ($y = \frac{\tau}{q_2} - \frac{q_1}{q_2}x$) that intersects the interior of $\mathcal{P}_k$.*

2.3 A Reverse Top-k Index

So, expectedly, our index is a convex hull approximation of the k-polygon that is annotated with the $\mathcal{O}(k)$ edges suppressed by each face of the approximation. The query operation is, as indicated in Theorem 1, to find the intersection with $\mathcal{P}_k$ of l_q, the dual of the query tuple q. Here we briefly describe the representation and construction of that index.

The k-Polygon Index Structure. For the data structure, we exploit Lemma 3. Let $\mathcal{H}$ denote the set of vertices of the convex hull of a k-polygon, $\mathcal{P}_k$. We maintain two arrays, which we collectively refer to as the *dual-array representation* of $\mathcal{P}_k$. The first, which we call the *convex hull array*, contains the $|\mathcal{H}|$ vertices of $\mathcal{H}$, ordered anti-clockwise from the x-axis. The second array, which we call the *concavity array*, is of size $|\mathcal{H}| - 1$. The i'th entry is a sequential list of the up to $2k - 1$ vertices of the k-polygon between the i'th and $(i + 1)$'st vertices of $\mathcal{H}$.

Constructing the k-Contour. To construct $\mathcal{P}_k$, we use a radial plane sweep. First, all lines are sorted by their y-intercept and the intersection points of adjacent lines are inserted into a priority queue, sorted by angle from the y-axis. The algorithm proceeds like a classic plane sweep algorithm by popping "events" (i.e., intersection points) from the queue; swapping the order of the intersecting lines to maintain a proper sort; and inserting into the priority queue the (up to) two new intersection points of lines made newly adjacent, provided that the angle is greater than that of the intersection point currently being processed. This proceeds until the queue is empty.

Determining the k-contour in this manner is efficient. Whenever a popped event involves the line in the k'th index of the sorted array of lines, it represents a new vertex of the k-contour.

Querying the k-Contour Index. To query the dual-array representation, we conduct a binary search on the convex hull to find the (at most two) intersection points with l_q. Then, the intersection points of l_q with $\mathcal{P}_k$ must occur within the concavity lists for those two convex hull edges. We proceed with a sequential scan of the $\mathcal{O}(k)$ edges of $\mathcal{P}_k$ in each list and report all intersection points.

Asymptotic Analysis. From the algorithm we describe, one can derive the following bounds on our algorithm:

Theorem 2. *The two dimensional mRTOP problem can be solved using $\mathcal{O}(\log n + k)$ query time with an index that requires $\mathcal{O}(n)$ disk space.*

3 Experimental Evaluation

Here, we examine the performance of our index through experimentation. We answer two questions. Since the size of the k-polygon is $\mathcal{O}(|\mathcal{D}|)$ and the size of its convex hull is $|\mathcal{H}|$, then the query cost of our index is $\mathcal{O}(k + \log\ |\mathcal{H}|)$. So, a natural question is, "What values do $|DS|$ and $|\mathcal{H}|$ typically have?" This indicates how much disk space the data structure requires and whether it will fit in memory. It also indicates the number of IOs required, if not.

The second question is of raw performance, the wall time to build and later query the index. We compare our index to that of repeatedly executing the algorithms of Vlachou et al. [10] and of Wang et al. [12].

3.1 Experimental Setup

For the experiments, we implemented and optimised the algorithms of Vlachou et al., of Wang et al., and of this paper (Chester et al.) in C and compiled our implementations with the GNU C compiler 4.4.5 using the *-O6* flag. We ignore the cost of outputting the response, because this is moreorless the same for each algorithm. On the other hand, each algorithm interprets the tuples differently, so we do include in the measurements the cost of reading the input files.

We ran the experiments on a machine with an AMD Athlon processor with four 3GHz overclocked cores and 8GB RAM, running Ubuntu. The timings were calculated using the linux *time* command. The data structure sizes were measured in separate trials by modifying the code to count vertices.

Table 2. *The five basketball datasets.* Each of the five datasets under study and the attributes used in the $2d$ projection to create the datasets.

Dataset #	x-Attribute	y-Attribute	Correlation
1	Points,	Field goals made	Correlated
2	Defensive rebounds,	Blocks	Anticorrelated
3	Personal fouls,	Free throw attempts	Independent
4	Defensive rebounds,	Assists	Correlated
5	Blocks,	Three pointers made	Anticorrelated

Datasets. We use the regular season statistics from *databasebasketball.com* and create five datasets with which to perform experiments by projecting combinations of two attributes. The attribute combinations are chosen to diversify the degree of (anti-)correlation based on intuitive reasoning about the attributes and are described in Table 2. A traditional top-k query on each pair is equivalent to asking for the k best player-seasons according to a given blend of the skills.

We reserve the most recent season, 2009, as a set of 578 query points and use the other seasons, 1946-2008, as the dataset of 21383 tuples. Each mRTOP query is equivalent to asking, "for which blends, if any, of the given two skills was this particular player's performance this season ranked in the top-k all-time?" This contrasts to traditional analysis of basketball data which would struggle to define rankings on more than one attribute, to the detriment of rounded players.

3.2 Experimental Results

Our first experiment evaluated the size of our data structure, since it strongly affects query time. Fig. 2 shows how the size varies with k on datasets 3 and 1, which produced the largest and smallest data structures, respectively. Other datasets exhibited very similar behaviour, with the convex hull size remaining relatively constant and the total contour size growing linearly with k.

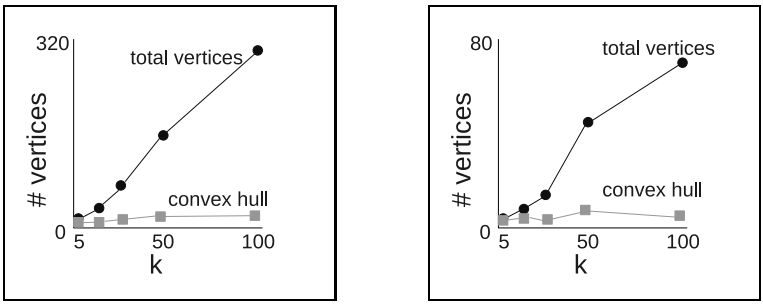


Fig. 2. *Contour size as a function of k.* To the left is shown the number of data structure vertices for the contours derived on dataset 5; to the right, on dataset 1.

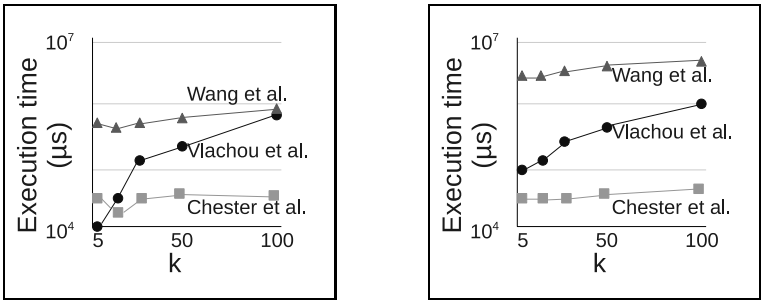


Fig. 3. *Execution time for the implemented algorithms.* For the batch of 578 queries comprised of 2009 basketball statistics, using the statistics from 1946-2008 as a dataset. Vlachou et al. is measured using pre-sorted data; Wang et al. is not. To the left are results for dataset 4 and, the right, 5. The y-axis is on a logarithmic scale.

The second experiment investigated how construction and query time varied with respect to k and attribute correlation. The execution time of our algorithm was nigh constant: across all experiments the construction time had an average duration of 34ms with a standard deviation of 9.1ms; the query time, 480μs with a standard deviation of 22μs; and the total time for construction and querying, 35ms with a standard deviation of 8.8ms. Fig. 3 illustrates the total execution times for the three algorithms.[2] Although not previously reported in literaure, we observed the algorithms of Vlachou et al. and of Wang et al. are sensitive to the sortedness of the input; so, we report their performances for the quicker of when the data is presorted by y value and when that presorted file is randomized with the linux command *sort -R*. The algorithm of Vlachou et al. was up to an order of magnitude *faster* on presorted data; that of Wang et al. was up to an order of magnitude *slower* on presorted data.

[2] We omitted results for datasets 1, 2, and 3 because they were very similar to those of dataset 4. We show the results in which the algorithms perform closest.

3.3 Discussion

Overall, our indexing does superbly, with a query cost slightly less than 1μs per query, independent of k, typically three to four orders of magnitude faster than the other two algorithms. In fact, our algorithm in most cases runs one or two orders of magnitude faster, *even with the construction cost included*. This is remarkable, because the construction cost we assume will be done offline in advance of the query's arrival, but implies that the index construction is sufficiently fast to render it feasible in non-indexing scenarios, too. That the query time does not vary much is not surprising in light of our analysis of data structure size. From Fig. 2, the convex hull is consistently under forty vertices.

It is worth noting that there are a few instances in which the algorithm of Vlachou et al. outperforms our index for low values of k on presorted data. As soon as k tuples are seen that dominate the query tuple, a null result can be reported and the Vlachou et al. algorithm can be halted. This is substantially more likely for lower k. With sorted data, these points will be among the first seen. So, an important question is whether expecting presorted data is realistic.

With such encouraging experimental results, we intend next to generalize the algorithm and techniques for higher dimensions.

References

1. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications, 3rd edn. Springer TELOS, Santa Clara (2008)
2. Chester, S., Thomo, A., Venkatesh, S., Whitesides, S.: Indexing for vector projections. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part II. LNCS, vol. 6588, pp. 367–376. Springer, Heidelberg (2011)
3. Das, G., Gunopulos, D., Koudas, N., Sarkas, N.: Ad-hoc top-k query answering for data streams. In: Proc. VLDB, pp. 183–194. ACM, New York (2007)
4. Hugg, J., Rafalin, E., Seyboth, K., Souvaine, D.: An experimental study of old and new depth measures. In: Proc. ALENEX, pp. 51–64. Springer (2006)
5. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing techniques in relational database systems. ACM Computing Surveys 40 (October 2008)
6. Meliou, A., Gatterbauer, W., Suciu, D.: Reverse data management. In: Proc. VLDB, vol. 4(12) (August 2011)
7. Ranu, S., Singh, A.K.: Answering top-k queries over a mixture of attractive and repulsive dimensions. In: Proc. VLDB, vol. 5(3) (2011)
8. Rousseeuw, P.J., Hubert, M.: Depth in an arrangement of hyperplanes. Discrete & Computational Geometry 22(1), 167–176 (1999)
9. Sharir, M.: Arrangements in higher dimensions: Voronoi diagrams, motion planning, and other applications. In: Sack, J.-R., Akl, S.G., Dehne, F., Santoro, N. (eds.) WADS 1995. LNCS, vol. 955, pp. 109–121. Springer, Heidelberg (1995)
10. Vlachou, A., Doulkeridis, C., Kotidis, Y., Norvag, K.: Reverse top-k queries. In: Proc. ICDE, pp. 365–376. IEEE (March 2010)
11. Vlachou, A., Doulkeridis, C., Kotidis, Y., Norvag, K.: Monochromatic and bichromatic reverse top-k queries. TKDE 23, 1215–1229 (2011)
12. Wang, B., Dai, Z., Li, C., Chen, H.: Efficient computation of monochromatic reverse top-k queries. In: Proc. FSKD, vol. 4, pp. 1788–1792. IEEE (August 2010)
13. Zuo, Y., Serfling, R.: Structural properties and convergence results for contours of sample statistical depth functions. Annals of Statistics 28, 483–499 (2000)

Beyond Click Graph: Topic Modeling for Search Engine Query Log Analysis

Di Jiang, Kenneth Wai-Ting Leung, Wilfred Ng, and Hao Li

Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong, China
{dijiang,kwtleung,wilfred,hliaj}@cse.ust.hk

Abstract. Search engine query log is a valuable information source to analyze the users' interests and preferences. In existing work, click graph is intensively utilized to analyze the information in query log. However, click graph is usually plagued by low information coverage, failure of capturing the diverse types of co-occurrence and the incapability of discovering the latent semantics in data. In this paper, we go beyond click graph and analyze query log through the new perspective of probabilistic topic modeling. In order to systematically explore the potential assumptions of the latent structure of the log data, we propose three different topic models. The first model, the *Meta-word Model* (MWM), unifies the co-occurrence of query terms and URLs by the meta-word occurrence. The second model, the *Term-URL Model* (TUM), captures the characteristics of query terms and URLs separately. The third model, the *Clickthrough Model* (CTM), captures the clicking behavior explicitly and models the ternary relation between search queries, query terms and URLs. We evaluate the three proposed models against several strong baselines on a real-life query log. The experimental results show that the proposed models demonstrate significantly improved performance with respect to different quantitative metrics and also in applications such as date prediction, community discovery and URL annotation.

1 Introduction

Search engine query log provides a good window for understanding the users' underlying interests and preferences. Therefore, query log has also been serving as the basis of many functionalities of search engines, such as spelling correction [1], query suggestion [3] and search personalization [16][9]. The majority of existing work on query log analysis is conducted by analyzing click graph, which is essentially a bipartite graph built upon search queries and clicked URLs. While reasonably good performance has been achieved for some tasks, some inherent drawbacks of click graph have not been satisfactorily addressed so far. The following example illustrates the limitations of click graph.

Consider the query log sample in Table 1 and its corresponding click graph in Fig. 1. The following three limitations of click graph can be observed:

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 209–223, 2013.

Table 1. Search Engine Query Log Sample

ID	User	Query	Clicked URL	Timestamp
q_1	u_1	flash player	get.adobe.com/.../...	2011-04-11 15:12:41
q_2	u_1	adobe		2011-04-12 11:13:44
q_3	u_1	flash	www.adobe.com/.../...	2011-04-12 11:14:21
q_4	u_1	photoshop	www.brothersoft.com/...	2011-04-13 07:13:01
q_5	u_1	flash		2011-04-15 19:13:01
q_6	u_2	blade series	en.wikipedia.org/.../...	2011-04-10 09:44:26
q_7	u_2	superhero	www.superherodb.com/	2011-04-14 14:35:14
q_8	u_2	flash	www.imdb.com/.../...	2011-04-14 14:36:26

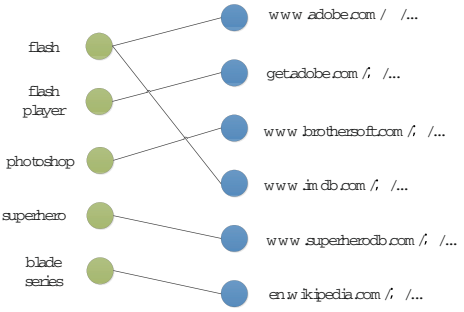


Fig. 1. An Example Click Graph of Table 1

1. *The click graph combines q_3, q_5 and q_8 as a single node "flash". However, these queries are submitted to satisfy different information needs. By manually analyzing the corresponding URLs, we find that u_1 submits q_3 to look for a software product of Adobe while u_2 submits q_8 to search a popular TV series. Due to the polysemy of the query terms, information confusion clearly exists in the example click graph.*
2. *The click graph ignores the user information, which is effective in disambiguating the meaning of queries. If the user information is taken into consideration, we can see that u_1 is interested in IT technology and u_2 is interested in superheroes. Thus, q_3 is more likely to be related to the software product and q_8 is more likely to be about the TV series of the superhero.*
3. *The click graph ignores the information of timestamps and abandoned queries, which are critical for inferring a query's real meaning. Before submitting q_3, u_1 searched q_2 ("adobe") within a minute. Thus, q_3 is likely to be related to "adobe" and to be interpreted as Adobe Flash. Right before q_8, u_2 searched q_7 ("superhero"). Therefore, q_8 is more likely to be related to the TV series of the superhero Flash. Another drawback of ignoring the timestamp is the lack of capturing the web dynamics. For example, if the TV series Flash is a hot topic during the period, then q_5 is also likely to be submitted by u_1 to satisfy the information need about the TV series.*

Due to the advantage of capturing complicated relations in a principled manner, we explore the paradigm of probabilistic topic modeling to tackle with the unsolved problems of click graph. However, this task is not trivial and the challenges are primarily twofold. First, as is shown in Table 1, each log entry contains different types of information. How to integrally utilize all the information to tackle the limitations of click graph is a challenging issue. Second, different from the scenario of document modeling which faces homogeneous *words*, query log is composed of two kinds of heterogeneous items, the query terms and the URLs. Thus, topic modeling on query log needs to handle the heterogeneous items and capture the complicated co-occurrence between them. The two challenges render conventional topic models inapplicable or they can only work suboptimally in the scenario of query log analysis.

To better handle with the aforementioned challenges, we first pre-process the raw query log, making it suitable for topic modeling. Then we propose three probabilistic topic models: the *Meta-word Model* (MWM), the *Term-URL Model* (TUM) and the *Clickthrough Model* (CTM), in order to systematically explore the potential assumptions of the relations between the query terms and URLs. The *Meta-word Model* (MWM) unifies the co-occurrence relations between query terms and URLs as the meta-word co-occurrence, and assumes these meta-words follow the same distribution given a topic. The *Term-URL Model* (TUM) models the clickthrough behavior explicitly and assumes that query terms and URLs follow different distributions given a topic. The *Clickthrough Model* (CTM) introduces the variable of search query and utilizes the ternary relationship between search queries, query terms and URLs. We quantitatively evaluate the proposed models with conventional topic models such as Latent Dirichlet Allocation (LDA) [2] and Topics-Over-Time (TOT) [21]. The proposed models demonstrate significantly better performance in the scenario of query log analysis. Furthermore, we also compare the proposed models with some click graph based approaches in the applications of community discovery and URL annotation. The three models also demonstrate superior performance. The contributions of this paper are summarized as follows:

1. *First, we identify the limitations of click graph and view search engine query log from a new perspective of probabilistic topic modeling.*
2. *Second, we formulate three probabilistic topic models to analyze query log. The three models can effectively integrate multiple types of information in query log and systematically explore different assumptions of the relations between query terms and URLs.*
3. *Third, we carry out extensive evaluations on the three proposed models with a real-life query log. The proposed models demonstrate significantly improved performance compared to several strong baselines with regard to different quantitative metrics. We also validate the usefulness of the models with applications such as date prediction, community discovery and URL annotation.*

The remainder of the paper is organized as follows. In Section 2, we review the related work. In Section 3, we discuss the pre-processing procedure. In Section 4,

we formulate three probability topic models, the *Meta-word Model* (MWM), the *Term-URL Model* (TUM) and the *Clickthough Model* (CTM), in order to discover search topics from query log. The experimental results are presented in Section 5. Some discussion is presented in Section 6. Finally, the paper is concluded in Section 7.

2 Related Work

In recent years, probabilistic data analysis is gaining momentum in data mining [2] [19] [21]. Among them, the topic modeling approach demonstrates superior performance in exploring the latent knowledge of electronic archives. Griffiths *et al.* [4] applied Latent Dirichlet Allocation (LDA) to scientific articles and studied its effectiveness in finding scientific topics. As an extension of LDA, Wang *et al.* [21] presented a topic model that captures both the latent structure of data and how the structure changes over time. There follow more topic models that are proposed to handle the problems of document analysis that exist in specific domains, such as sentiment analysis [11] and geographical analysis [10]. Furthermore, Kang *et al.* [12] proposed a topic-concept cube which supports online multidimensional mining of query log. Mei *et al.* [17] proposed a novel probabilistic approach to model the subtopic themes and spatiotemporal theme patterns simultaneously. Some recent work on query log analysis also studied the impact of temporal issues. Ha-Thuc *et al.* [5] proposed an approach for event tracking with emphasis on scalability and selectivity, and their experiments showed that the approach can extract important temporal patterns about the news events. To the best of our knowledge, our work is the first one to systemically explore different assumptions about the relations between query terms and URLs via probabilistic topic modeling. The experimental results show that topic modeling is an effective approach to discover the latent semantics in query log and outperforms several strong baselines with regard to both quantitative metrics and real applications.

3 Pre-processing

In the pre-processing procedure, we first organize the log entries of the ith search engine user as the document d_i and then group the consecutive queries that have semantic relations as *search sessions*. A search session refers to a series of queries which are submitted within a short time period to satisfy the same information need. In order to avoid the performance degradation that is caused by including irrelevant queries in the same session, we prioritize the semantic coherency across queries within the same session. The *query reformulation taxonomy* proposed in [8] consists of a series of rules that evaluate the lexical similarity between queries and demonstrates high precision in detecting semantically relevant search queries. Thus, we utilize it to evaluate the relevancy between two consecutive queries in the log. Finally, we use the stopword list provided in [15] to filter

out the non-informative terms from each search query. The timestamps are normalized to a real number between 0 and 1 based on the earliest and the latest timestamps in query log.

4 Topic Models

4.1 Meta-word Model (MWM)

The *Meta-word Model* (MWM) assumes that each user's query log (i.e., each document) has a unique distribution over a set of K search topics and each of which is represented as a multinomial distribution over all the meta-words in the vocabulary drawn from a symmetric Dirichlet prior β. The meta-words have two potential interpretations:

- The first interpretation only utilizes the query terms. This interpretation simply ignores the URLs and is denoted as MWM-T in the rest of the paper.
- The second interpretation considers both the query terms and the URLs as meta-words. This interpretation is denoted as MWM-TU in the rest of the paper. Note that this interpretation does not explicitly capture the click-through behavior, since the query terms and URLs are utilized without differentiation.

Although we may only use the URLs as the meta-words to derive topics that solely consist of URLs, this option is not included as an interpretation of MWM due to its lack of topic interpretability and the incapability of supporting downstream applications. The generative process of MWM is depicted in Algorithm 1. Each document is generated by first drawing a document-specific mix θ over topic 1 to topic K that is drawn from a symmetric Dirichlet prior α. Since the information within the same session serves the same information need, we assume that a session is relevant to the same search topic. This observation inspires us to use sessions rather than meta-words as the basic unit of topic assignment. Since we assign search topics on a session basis, a session-specific topic z is drawn from θ. Then within the session, some meta-words are drawn from a multinomial distribution based on the topic z. In MWM, the topic assignment of a session is not only subject to the co-occurrence of meta-words but also subject to the timestamps within the session. We utilize the continuous Beta distribution to capture the temporal prominence of each topic. The timestamps within a session are drawn from a Beta distribution ψ_z which is specific to the session topic z. Ultimately, each meta-word w is picked in proportion to how much the enclosing document prefers the topic z and how much the topic prefers the meta-word w. The timestamp is picked in proportion to how much the enclosing document prefers the topic z and how much the topic prefers the timestamp t.

We aim to find an efficient way to compute the joint likelihood of the observed meta-words and timestamps with the hyperparameters:

$$P(\mathbf{w}, \mathbf{t}, \mathbf{z}|\alpha, \beta, \Psi) = P(\mathbf{w}|\mathbf{z}, \beta)P(\mathbf{t}|\Psi, \mathbf{z})P(\mathbf{z}|\alpha). \quad (1)$$

Algorithm 1. Generative Process of MWM

for topic $k \in 1, ..., K$ **do**
  draw a meta-word distribution $\phi_k \sim \text{Dirichlet}(\beta)$
end for
for each document $d \in 1, ..., D$ **do**
  draw d's topic distribution $\theta_d \sim \text{Dirichlet}(\alpha)$
  for each session s in d **do**
    choose a topic $z \sim \text{Multinomial}(\theta_d)$
    generate meta-words $w \sim \text{Multinomial}(\phi_z)$
    draw timestamps $t \sim$ from $\text{Beta}(\Psi_z)$
  end for
end for

We will use this joint likelihood to derive efficient updates for the parameters Θ, Φ and Ψ. The right term $P(\mathbf{z}|\alpha) = \int P(\mathbf{z}|\Theta)P(\Theta|\alpha)d\Theta$ is the same as for the standard LDA and this term ultimately contributes the same terms to the full conditional as well as the sampling formula for updating individual topic assignments z_i. Thus, we use the same derivation as in [4]. Using the independence assumptions of the model, we consider the probability of the meta-words and the timestamps. The probability of the meta-words is given as follows:

$$P(\mathbf{w}|\mathbf{z}, \beta) = \int \prod_{d=1}^{D} \prod_{s=1}^{S_d} \prod_{i=1}^{W_{ds}} P(w_{dsi}|\phi_{z_{ds}})^{N_{dsw_{dsi}}} \prod_{z=1}^{K} P(\phi_{z_{ds}}|\beta) d\Phi. \tag{2}$$

The probability of the timestamps is listed as follows:

$$P(\mathbf{t}|\Psi, \mathbf{z}) = \prod_{d=1}^{D} \prod_{s=1}^{S_d} \prod_{j=1}^{T_{ds}} P(t_{dsj}|\psi_{z_{ds}})^{N_{dst_{dsj}}}. \tag{3}$$

After combining terms, applying Bayes rule and folding terms into the proportionality constant, the conditional probability of the kth topic for the ith session is defined as follows:

$$\begin{aligned} &P(z_i = k|\mathbf{z}_{-\mathbf{i}}, \mathbf{w}, \mathbf{t}, \alpha, \beta, \Psi) \propto \\ &\frac{C_{dk}^{DK} + \alpha_k}{\sum_{k'=1}^{K}(C_{dk'}^{DK} + \alpha_{k'})} \frac{\Gamma(\sum_{w=1}^{W}(C_{kw}^{KW} + \beta_w))}{\Gamma(\sum_{w=1}^{W}(C_{kw}^{KW} + \beta_w + N_{iw}))} \\ &\prod_{w=1}^{W} \frac{\Gamma(C_{kw}^{KW} + \beta_w + N_{iw})}{\Gamma(C_{kw}^{KW} + \beta_w)} \prod_{j=1}^{T} \frac{(1-t_j)^{\psi_{k1}-1} t_j^{\psi_{k2}-1}}{B(\psi_{k1}, \psi_{k2})}. \end{aligned} \tag{4}$$

Gibbs sampling [20] is used to estimate the probability that a query belongs to a certain topic. For simplicity and efficiency, we estimate these Beta distribution ψ_z by the method of moments, once per iteration of Gibbs sampling. After each iteration, we update ψ_{k1} and ψ_{k2} for each topic as follows:

$$\psi_{k1} = \bar{t_k}(\frac{\bar{t_k}(1-\bar{t_k})}{s_k^2} - 1), \tag{5}$$

$$\psi_{k2} = (1 - \bar{t_k})(\frac{\bar{t_k}(1 - \bar{t_k})}{s_k^2} - 1), \tag{6}$$

where $\bar{t_k}$ and s_k^2 denote the sample mean and biased sample variance of topic k's timestamps.

4.2 Term-URL Model (TUM)

Since the semantics of the URLs is less ambiguous than query terms [13], the URLs are stronger endorsements than query terms in terms of the topical commonality. Considering them separately can better capture the importance of URL clicking, since the amount of query terms significantly outnumbers that of the URLs. Therefore, we further propose the *Term-URL Model* (TUM) to capture the topical distribution of query terms and URLs separately.

The generative process of TUM is presented in Algorithm 2. Similar to MWM, we constrain that the query terms and URLs in the same session share the same topic. The query term selection process is the same as the meta-word selection process in MWM and the timestamp selection process is the same as that of the MWM. Additionally, each URL is picked in proportion to how much the enclosing document prefers the topic z and how much the topic prefers the URL u. The joint probability of terms, URLs and timestamps is given as follows:

Algorithm 2. Generative Process of TUM

for topic $k \in 1, ..., K$ **do**
  draw a term distribution $\phi_k \sim$ Dirichlet(β)
  draw a URL distribution $\Omega_k \sim$ Dirichlet(δ)
end for
for each document $d \in 1, ..., D$ **do**
  draw d's topic distribution $\theta_d \sim$ Dirichlet(α)
  for each session s in d **do**
    choose a topic $z \sim$ Multinomial(θ_d)
    generate terms $t \sim$ Multinomial(ϕ_z)
    if $X_s = 1$ **then**
      generate URLs $u \sim$ Multinomial(Ω_z)
    end if
    draw timestamps t from Beta(Ψ_z)
  end for
end for

$$P(\mathbf{w}, \mathbf{t}, \mathbf{u}, \mathbf{z}|\alpha, \beta, \delta, \Psi, \mathbf{X}) = P(\mathbf{w}|\mathbf{z}, \beta)P(\mathbf{u}|\mathbf{z}, \delta, \mathbf{X})P(\mathbf{t}|\Psi, \mathbf{z})P(\mathbf{z}|\alpha). \tag{7}$$

In Equation (7), the terms $P(\mathbf{z}|\alpha)$, $P(\mathbf{w}|\mathbf{z}, \beta)$ and $P(\mathbf{t}|\Psi, \mathbf{z})$ are the same as those in MWM. However, one issue that results from separate modeling the topical distributions of queries and URLs is that sometimes the session is *abandoned* and no clickthrough is raised. Since these queries are also complementary with

respect to the user's search interests [14], we introduce a variable X to indicate whether there exists clickthrough in the session.

$$P(\mathbf{u}|\mathbf{z},\delta,\mathbf{X}) = \int \prod_{d=1}^{D} \prod_{s=1}^{S_d} \{ \prod_{i=1}^{U_{ds}} P(u_{dsi}|\Omega_{z_{ds}})^{N_{dsu_{dsi}}} \}^{I(X_{ds}=1)} \prod_{z=1}^{K} P(\Omega_{z_{ds}}|\delta) d\Omega. \quad (8)$$

The generative process of TUM is further updated as follows. The user first decides the topic and then selects some query terms according to the chosen topic. For each session, the user needs to decide whether to click on some URLs. If $X = 1$, the user clicks on one or more URLs according to the chosen topic. Again, Gibbs sampling is used to estimate the probability that a query belongs to a certain topic. At each transition step of the Markov chain, the conditional probability of the topic of the queries in the ith session should be differentiated. Using a deduction process similar to MWM, we can obtain the update formulas for TUM. The topic of the queries in the ith session, z_i is drawn according to Equation (9) and the temporal parameters ψ_z are updated after each iteration according to Equations (5) and (6).

$$\begin{aligned} P(z_i = k|\mathbf{z_{-i}},\mathbf{w},\mathbf{t},\mathbf{u},\mathbf{X},\alpha,\beta,\delta,\Psi) \propto \\ \frac{C_{dk}^{DK} + \alpha_k}{\sum_{k'=1}^{K}(C_{dk'}^{DK} + \alpha_{k'})} \prod_{j=1}^{T} \frac{(1-t_j)^{\psi_{k1}-1} t_j^{\psi_{k2}-1}}{B(\psi_{k1},\psi_{k2})} \\ \frac{\Gamma(\sum_{w=1}^{W}(C_{kw}^{KW} + \beta_w))}{\Gamma(\sum_{w=1}^{W}(C_{kw}^{KW} + \beta_w + N_{iw}))} \prod_{w=1}^{W} \frac{\Gamma(C_{kw}^{KW} + \beta_w + N_{iw})}{\Gamma(C_{kw}^{KW} + \beta_w)} \\ \{ \frac{\Gamma(\sum_{u=1}^{U}(C_{ku}^{KU} + \delta_u))}{\Gamma(\sum_{u=1}^{U}(C_{ku}^{KU} + \delta_u + N_{iu}))} \prod_{u=1}^{U} \frac{\Gamma(C_{ku}^{KU} + \delta_u + N_{iu})}{\Gamma(C_{ku}^{KU} + \delta_u)} \}^{I(X_i=1)}. \end{aligned} \quad (9)$$

4.3 Clickthrough Model (CTM)

TUM assumes that query terms and URLs have the topical independence, i.e., their generation processes are independent given the topic. A more sophisticated strategy is to assume that the two items are not independent given the topic. We propose CTM to model the dependence between query terms and the URLs through search queries. The generative process of CTM is presented in Algorithm 3. As we assume that search queries, query terms and URLs within a session share the same search topic, we also use search session as the basic unit for topic assignment. Similar to MWM and TUM, a session-specific topic z is drawn from θ. Within the session, some query terms are drawn from a multinomial distribution based on the topic z. These query terms are then composed as search queries. We also use an indicator X to indicate whether there exists clickthrough in a search session. If there exists clickthrough ($X = 1$), the URLs are drawn from a multinomial distribution, which is identified by the selected topic z and the corresponding search query q. The joint likelihood of generating the corpus is as follows:

$$P(\mathbf{w},\mathbf{u},\mathbf{t},\mathbf{q},\mathbf{z}|\alpha,\beta,\delta,\Psi,\mathbf{X}) = P(\mathbf{w}|\mathbf{z},\beta)P(\mathbf{u}|\delta,\mathbf{z},\mathbf{q},\mathbf{X})P(\mathbf{q}|\mathbf{w})P(\mathbf{t}|\Psi,\mathbf{z})P(\mathbf{z}|\alpha), \quad (10)$$

Algorithm 3. Generative Process of CTM

for topic $k \in 1, ..., K$ **do**
  draw a term distribution $\phi_k \sim \text{Dirichlet}(\beta)$
  for query $q \in 1, ..., Q$ **do**
    draw a URL distribution $\Omega_{qk} \sim \text{Dirichlet}(\delta)$
  end for
end for
for each meta document $d \in 1, ..., D$ **do**
  draw d's topic distribution $\theta_d \sim \text{Dirichlet}(\alpha)$
  for each session s in d **do**
    choose a topic $z \sim \text{Multinomial}(\theta_d)$
    generate terms $t \sim \text{Multinomial}(\phi_z)$
    for each query q in s **do**
      if $X_q = 1$ **then**
        generate URLs $u \sim \text{Multinomial}(\Omega_{qz})$
      end if
    end for
    draw timestamps t from Beta(Ψ_z)
  end for
end for

In CTM, the formula terms $P(\mathbf{z}|\alpha)$, $P(\mathbf{w}|\mathbf{z}, \beta)$ and $P(\mathbf{t}|\Psi, \mathbf{z})$ are the same as those in TUM. $P(\mathbf{q}|\mathbf{w})$ is constant and independent of the search topic. The major difference is that w and u are not independent anymore given the topic. The generation of u subjects to both the topic z and the corresponding search query q.

$$P(\mathbf{u}|\delta, \mathbf{z}, \mathbf{q}, \mathbf{X}) = \int \prod_{d=1}^{D} \prod_{s=1}^{S_d} \{ \prod_{i=1}^{U_{ds}} P(u_{dsi}|\Omega_{q_{u_{dsi}} z_{ds}})^{N_{dsu_{dsi}}} \}^{I(X_{ds}=1)} \prod_{q=1}^{Q} \prod_{z=1}^{K} P(\Omega_{qz_{ds}}|\delta) d\Omega. \tag{11}$$

The conditional probability of the kth topic for the ith session is defined in Equation (12). After each iteration, the temporal parameters ψ_z are updated according to Equations (5) and (6).

$$P(z_i = k|\mathbf{z}_{-\mathbf{i}}, \mathbf{w}, \mathbf{t}, \mathbf{u}, \mathbf{X}, \alpha, \beta, \delta, \Psi) \propto \frac{C_{dk}^{DK} + \alpha_k}{\sum_{k'=1}^{K} C_{dk'}^{DK} + \alpha_{k'}} \prod_{j=1}^{T} \frac{(1-t_j)^{\psi_{k1}-1} t_j^{\psi_{k2}-1}}{B(\psi_{k1}, \psi_{k2})}$$
$$\frac{\Gamma(\sum_{w=1}^{W}(C_{kw}^{KW} + \beta_w))}{\Gamma(\sum_{w=1}^{W}(C_{kw}^{KW} + \beta_w + N_{iw}))} \prod_{w=1}^{W} \frac{\Gamma(C_{kw}^{KW} + \beta_w + N_{iw})}{\Gamma(C_{kw}^{KW} + \beta_w)}$$
$$\{ \prod_{q=1}^{Q_i} \frac{\Gamma(\sum_{u=1}^{U}(C_{qku}^{QKU} + \delta_u))}{\Gamma(\sum_{u=1}^{U}(C_{qku}^{QKU} + \delta_u + N_{iqu}))} \prod_{u=1}^{U_{iq}} \frac{\Gamma(C_{qku}^{QKU} + \delta_u + N_{iqu})}{\Gamma(C_{qku}^{QKU} + \delta_u)} \}^{I(X_i=1)} \tag{12}$$

5 Experiments

In this section, we present the experimental results. We utilize a real-world query log from a major commercial search engine to conduct the experiments.

The raw query log is pre-processed according to the discussion in Section 3. The dataset records the search history of 2,417 users during 3 months. In Section 5.1, we quantitatively evaluate the proposed models against LDA and TOT by using three metrics: the perplexity of held-out data, the predictive perplexity of partially observed data. In Sections 5.2, 5.3 and 5.4, we demonstrate the effectiveness of the the proposed models in applications such as date prediction, community discovery and URL annotation.

5.1 Quantitative Evaluation

We now evaluate the effectiveness of the proposed models by two quantitative metrics. The pre-processing procedures such as document grouping and stopword preprocessing are the same for all the models under evaluation. We choose the following methods as the baselines:

- LDA-T: Latent Dirichlet Allocation [2] that only utilizes the query terms.
- LDA-TU: Latent Dirichlet Allocation that utilizes both the query terms and URLs as metawords.
- TOT-T: Topics-Over-Time model [21] that only utilizes the query terms.
- TOT-TU: Topics-Over-Time model that utilizes the query terms and URLs as metawords.

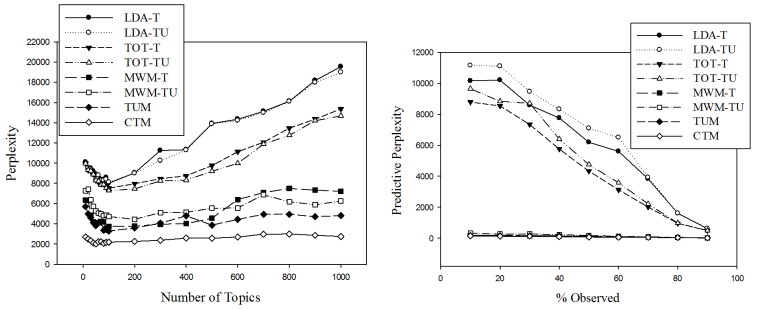
The first metric we use is the perplexity of heldout data. Perplexity is a measure of the ability of a model to generalize to unseen data. Better generalization performance is indicated by a lower perplexity. We compare the proposed models with LDA and TOT by a ten-fold cross validation. We use Equation (13) to calculate the perplexity for each model [18].

$$Perplexity_{heldout}(\mathcal{M}) = (\prod_{d=1}^{D}\prod_{i=1}^{N_d} p(w_i|\mathcal{M}))^{\frac{-1}{\sum_{d=1}^{D}(N_d)}}, \tag{13}$$

where $\mathcal{M}$ is the model learned from the training process.

Figure 2(a) illustrates the average perplexity for each model. MWM, TUM and CTM all provide significantly better fit than LDA and TOT. For example, when the number of topics is set to 100, the average perplexity of LDA-T is 8013, that of MWM-T is 3753, MWM-TU is 4713, TUM is 3287 and CTM achieves the lowest perplexity of 2189. We also observe that the performance of MWM-T and MTM-TU are comparable when the number of topics is small while MWM-TU has better performance than MWM-T when the number of topics is large. The result suggests that incorporating the URL information enables the model to support more topics.

Another metric for comparing the relative strengths of LDA and TOT with our proposed models is how well the models predict the remaining query terms after observing a portion of the user's search history. Suppose we observe the query terms $w_{1:P}$ from a user's query log and aim to find out which model provides a better predictive distribution $p(w|w_{1:P})$ of the remaining query terms. We use Equation 14 to calculate the perplexity of the remaining unseen data.



(a) Perplexity for held-out data (b) Predictive perplexity for partially observed data

Fig. 2. Perplexity Comparison

The results of the comparison are presented in Figure 2(b). We observe that the proposed models significantly outperform LDA. The average perplexity of MWM-T is 119, MWM-TU is 176, TUM is 112 and CTM demonstrates the best performance with an average perplexity of 83.

$$Perplexity_{portion}(\mathcal{M}) = (\prod_{d=1}^{D} \prod_{i=P+1}^{N_d} p(w_i|\mathcal{M}, w_{a:P}))^{\frac{-1}{\sum_{d=1}^{D}(N_d-P)}}. \quad (14)$$

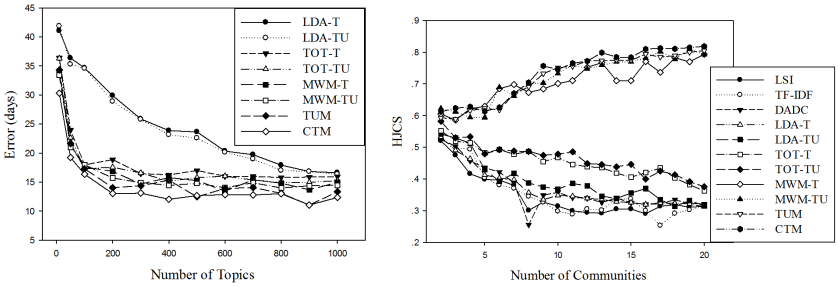
5.2 Data Prediction

We proceed to compare the accuracy of the timestamp prediction of our models given the query terms in a session. We use 6624 held-out search sessions as the evaluation data and then evaluate each model's ability to predict the date of a search session. The Beta distribution for each LDA topic is fitted in a post-hoc fashion. The results of the comparison are presented in Figure 3(a). The average date prediction error of LDA-T is 22.93 days and the average error of LDA-TU is 21.56 days. The average error of MWM-T is 15.14 days, that of MWM-TU is 14.94 days and TUM is 13.87 days and CTM demonstrate the highest date prediction accuracy with an average error of 11.26 days. The above three metrics indicate that the proposed models are better at capturing the temporal trends in web search and thus achieves better performance in date prediction.

5.3 Community Discovery

After processing each user's search history by the proposed models, the ith user's search interests are represented by a topic vector $(\theta_{i1}, \theta_{i2}, ..., \theta_{in})$ where θ_{ik} is a real number that indicates the ith user's endorsement for the kth search topic.

We prepare the ground truth with a small portion of the query log, including 500 users and their 114,400 queries. The queries are manually classified into



(a) Date prediction (b) Community Quality Analysis

Fig. 3. Performance Comparison

21 ODP[1] categories and thus each user is represented by a 21-element vector, where each element is the frequency of the user's queries that belong to the corresponding category. After normalization, the vector serves as the ground truth user profiles.

The first baseline for user profiling is the widely used TF-IDF text representation (denoted as TF-IDF), by which we represent a user's search history by his/her corresponding TF-IDF vector of query terms and URLs. The second baseline for user profiling is Latent Semantic Indexing (denoted as LSI) [7], which is able to extract the conceptual content of a body of text by establishing associations between those terms that occur in similar contexts. In the LSI profiling method, we use the extracted "concept" vector as the user profile. The fourth baseline is the DADC algorithm [13], which utilizes a variant of click graph to analyze the relation between users, search queries and concepts.

We then apply K-means clustering algorithm to the ground truth user profiles and those from the baselines and the proposed models. Our objective is to evaluate whether the profiles obtained from the proposed models can generate communities that are closer to that of the ground truth comparing to the baselines. For the purpose of community quality evaluation, we check the resultant relation[2] between each pair of users against the results obtained from the ground truth. For the clustering result c_p that is obtained from the user profiling method p and two users i and j, if the relation of i and j in c_p is consistent with the result c_{truth} that is obtained from the ground truth, then we consider it as a positive judgment. We repeat this process for each pair of users and then normalize the final counting by the total number of user pairs. We call the normalized value the *Human Judgment Correlation Score* (HJCS) and formally define HJSC as follows:

$$HJCS_p = \frac{\sum_{i,j \neq i} 1(c_p(i,j) = c_{truth}(i,j))}{C_n^2}. \tag{15}$$

[1] `http://www.dmoz.org/`

[2] The relation means whether i and j belong to the same cluster or not.

The higher the HJCS, the closer the correlation between c_p and c_{truth}. The results shown in Figure 3 (b) demonstrate the effectiveness of the proposed models. With the increase of the number of communities, the HJCS of TF-IDF, LSI and DADC demonstrate decreasing trends while the HJCS of the proposed models gradually increase. Since high HJCS is trivial when the number of communities is small (e.g., in the extreme case, HJCS value would be all 100% if the number of communities is set to be 1) while high HJCS is challenging when the number of communities is large, the increasing trend of our models illustrates their high degree of correlation with human judgment and suggests that they are effective to discover small but coherent user communities.

5.4 URL Annotation

We now evaluate the quality of URL annotations that are generated by the proposed models. LDA-TU, TOT-TU, MWM-TU, TUM and CTM support the discovery of the semantic relation between query terms and URLs. Within each search topic, the top-ranked query terms can be considered as the annotation of the top-ranked URLs. In order to quantitatively evaluate the quality of URL annotation obtained from the models, we compare them with *M2* and *baseline+M2*, which are two click graph based methods and achieve the best performance in [6], in the task of URL classification. For a specific topic, we select the top 2 URLs and use the top 10 terms as their annotations. In total 500 URLs are selected for this experiment. Other experimental settings are the same as that discussed in [6] and are skipped here to save space. TUM and CTM achieve classification accuracies of 0.6397 and 0.6535, which significantly outperform M2's 0.5124 and baseline+M2's 0.5558. LDA-TU, TOT-TU and MWM-TU achieve accuracies of 0.4979, 0.5213 and 0.5444, respectively. The result suggests that the TUM and CTM are effective to capture the semantic relation between query terms and URLs. Thus, the resultant search topics are effective for interpreting the URL's content with higher accuracy.

6 Discussion

Based on the evaluation in the previous section, we find that query log analysis needs specialized probabilistic topic models due to its unique characteristics. Conventional topic models such as LDA and TOT can only work suboptimally in this task. We also observe that good probabilistic topic models can outperform the click graph (or its variant) based methods in the task of community discovery and URL annotation. The result indicates that utilizing the information missed in click graph can effectively boost the performance of some applications. Among all the three proposed models, the CTM model achieves the best performances in most cases. This result shows that the two heterogenous items, query terms and URLs follow distinct distributions and are strictly coupled via search queries. However, the performance superiority of CTM is gained with a price to pay. The space complexity of MWM is $O(DK + KW)$, where W is the number of

metawords. The space complexity of TUM is $O(DK + KW + KU)$, where W is the number of query terms and U is the number of URLs. The space complexity of CTM is $O(DK + KW + QKU)$, where W is the number of query terms, U is the number of URLs and Q is the number of queries. Therefore, CTM usually consumed more space than MWM and TUM. Thus, when the memory is limited, MWM or TUM can be used as good alternatives of CTM.

7 Conclusion

In this paper, we introduce three new probabilistic topic models, the *Metaword Model* (MWM), the *Term-URL Model* (TUM) and the *Clickthrough Model* (CTM), in order to analyze search engine query log. The three models explore different assumptions to discover search topics from the users' search history. Parameter inference approaches such as Gibbs sampling are further introduced to estimate the value of latent variables. Our findings demonstrate that probabilistic topic modeling has the advantage of seamlessly integrating different types of information in query log and effectively capturing the latent semantics. Empirical evaluations on a real-life query log unequivocally demonstrate the superiority of the proposed models and their utilities in different applications.

Acknowledgments. This work is partially supported by GRF under grant numbers HKUST 617610 and 618509. We also wish to thank the anonymous reviewers for their comments.

References

1. Ahmad, F., Kondrak, G.: Learning a spelling error model from search query logs. In: Proc. of the HLT- EMNLP Conference (2005)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. The Journal of Machine Learning Research (2003)
3. Deng, H., King, I., Lyu, M.R.: Entropy-biased models for query representation on the click graph. In: Proc. of the ACM SIGIR Conference (2009)
4. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proc. of the National Academy of Sciences of the United States of America (2004)
5. Ha-Thuc, V., Mejova, Y., Harris, C., Srinivasan, P.: A relevance-based topic model for news event tracking. In: Proc. of the ACM SIGIR Conference (2009)
6. Hinne, M., Kraaij, W., Raaijmakers, S., Verberne, S., van der Weide, T., Van Der Heijden, M.: Annotation of urls: more than the sum of parts. In: Proceedings of the ACM SIGIR Conference (2009)
7. Hofmann, T.: Probabilistic latent semantic indexing. In: Proc. of the ACM SIGIR Conference (1999)
8. Huang, J., Efthimiadis, E.N.: Analyzing and evaluating query reformulation strategies in web search logs. In: Proc. of the ACM CIKM Conference (2009)
9. Jiang, D., Leung, K.W.T., Ng, W.: Context-aware search personalization with concept preference. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management

10. Jiang, D., Vosecky, J., Leung, K.W.T., Ng, W.: G-wstd: A framework for geographic web search topic discovery. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management
11. Jo, Y., Oh, A.H.: Aspect and sentiment unification model for online review analysis. In: Proc. of the Fourth ACM WSDM Conference (2011)
12. Kang, D., Jiang, D., Pei, J., Liao, Z., Sun, X., Choi, H.J.: Multidimensional mining of large-scale search logs: a topic-concept cube approach. In: Proc. of the ACM WSDM Conference (2011)
13. Leung, K.W.-T., Lee, D.L.: Dynamic agglomerative-divisive clustering of clickthrough data for collaborative web search. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5981, pp. 635–642. Springer, Heidelberg (2010)
14. Li, J., Huffman, S., Tokuda, A.: Good abandonment in mobile and pc internet search. In: Proc. of the ACM SIGIR Conference (2009)
15. Manning, C.D., Raghavan, P., Schutze, H.: Introduction to information retrieval. Cambridge University Press, Cambridge (2008)
16. Matthijs, N., Radlinski, F.: Personalizing web search using long term browsing history. In: Proc. of the ACM WSDM Conference (2011)
17. Mei, Q., Liu, C., Su, H., Zhai, C.X.: A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In: Proc. of the WWW Conference (2006)
18. Rosen-Zvi, M., Griffiths, T., Steyvers, M., Smyth, P.: The author-topic model for authors and documents. In: Proc. of the UAI Conference (2004)
19. Tong, Y., Chen, L., Ding, B.: Discovering threshold-based frequent closed itemsets over probabilistic data. In: IEEE 28th International Conference on Data Engineering (2012)
20. Walsh, B.: Markov chain monte carlo and gibbs sampling (2004)
21. Wang, X., McCallum, A.: Topics over time: a non-markov continuous-time model of topical trends. In: Proc. of the ACM SIGKDD Conference (2006)

Continuous Topically Related Queries Grouping and Its Application on Interest Identification

Pengfei Zhao, Kenneth Wai-Ting Leung, and Dik Lun Lee

Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong, China

Abstract. When a user performs a search on a search engine, the query reflects a particular interest of the user. The interest may either span a short session of a few minutes, or a long period of time like months or years. In the latter, the user may perform searching related to a particular interest from time to time, making the queries related to that interest sporadically distributed in the search log. Identification of these topically related queries is very meaningful, since it can help the search engine better understand the user's interest and hence deliver better results to the user. In this paper, we propose a method to aggregate topically related queries into interests regardless of where the queries appear in the search log. It first identifies sets of continuous topically-related queries called CTQs and then clusters similar CTQs together to form interests. In order to identify the CTQs accurately, we propose the Pattern-Concept-Time-Based (PCTB) method that utilizes query reformulation patterns, concepts behind the queries and the user's continuous searching behavior to compute the similarity between two queries. To evaluate the effectiveness of our approach, we employ the AOL search log as our test dataset and develop a search middleware on top of Google for extracting concepts related to the queries. Experimental results show that our method can obtain a high precision and recall on identifying CTQs, which in turn improves the performance of interest identification.

1 Introduction

When a user submits a query to a search engine, the user usually has in mind a specific *interest* (also referred to as *information goal* or *information need*). In response to the query, the search engine returns to the user a list of search results. The user then clicks on the results that he/she judges as relevant to his/her interest. If the user finds the search results not relevant enough, he/she may reformulate the query to retrieve a more relevant set of results. The *search engine log* records all of the users' queries, clickthroughs and time, from which a user's interests can be mined. The discovered user interests can then be applied to applications such as search engines and recommendation systems to improve their performance.

Pass, et. al., studied two billion web queries and found that 28% of the queries are modifications of the previous queries [15]. These modifications are always done within a short session and adhere to the same search interest, and are referred to as *query reformulations* or *query refinements*. However, a user's interest may last beyond a session. In other words, queries related to the same interest may be sporadically distributed across

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 224–238, 2013.

Table 1. Example of an Interest

User id	Query	Time
599381	mcdonalds all american high school basketball games	2006-03-15 19:41:01
599381	mcdonalds all american high school basketball game information	2006-03-15 19:42:43
	... other queries ...	
599381	girls intensive basketball camps in california	2006-04-16 01:53:46

multiple sessions in the search log. Consider the example in Table 1, which is a snapshot of the AOL log. The user first submitted the query "mcdonalds all american high school basketball games", indicating that he/she was interested in finding some activities related to basketball. After browsing through the results from the query, the user might find the results not relevant enough and thus reformulated the first query into "mcdonalds all american high school basketball game information" to look for more specific information. Since the second query is a reformulation[1] of the first query and issued only two minutes after the first query, they likely reflected the same interest of the user. After the second query, the user might have found the desired information and stopped searching further. After a month, the user issued another query related to basketball activities. This search log shows that the user has an *interest* in basketball and that the interest went beyond the short session containing the first two queries.

In this paper, a set of consecutive queries initiated by a user with the same search interest is defined as *continuous topically related queries*. We refer to the set as a *CTQ*. Notice that according to our definition the queries within a CTQ do not have to fall within a certain time gap or are reformulated queries. However, the queries in a CTQ must be *consecutive* and *topically related*. We further define an *interest* as a set of topically related queries that may appear anywhere in user's entire search log. To identify a user's interest, we propose to (i) identify the CTQs in a search log, and (ii) cluster the CTQs in user's entire search log into search interests.

We observe that most existing methods for identifying CTQs are based on time thresholds or overlapping keywords between consecutive queries without considering the concepts behind the queries. These methods have two drawbacks. First, methods based on time thresholds only would consider two consecutive queries different if they exceed the pre-defined time threshold even if one is a close reformulation of the other. Second, even if two consecutive queries fall within the time threshold, methods based on keyword overlap will not be able to identify semantically related queries (e.g., "aviation" and "flying") because of the lack of common keywords in the two queries.

Although some methods (e.g., [8]) employed lexical databases such as Wordnet to identify synonyms, they cannot deal with the wide diversity of queries issued by the users. To tackle this problem, we propose a CTQ identification method which extracts concepts representing the semantics of a query and uses them to estimate the similarity between queries. We further propose an effective mechanism to integrate our

[1] Rules for determining query reformulation are discussed in Section 4.2.

concept-based method with the traditional time-threshold and query reformulation methods to achieve high effectiveness. We call our proposed method *Pattern-Concept-Time-Based* (PCTB) method. PCTB is capable of grouping semantically related queries together even if they are different syntactically or separated by a large time gap. Since a CTQ represents a common search goal, it can be considered a "unit interest". We can further cluster the CTQs extracted from the entire search log to form general interests. To evaluate the effectiveness of our approach, we conduct experiments using the AOL search log. The experimental results confirm that PCTB yields better precision and recall comparing to existing CTQ identification methods that are based on time thresholds or overlapping query keywords. The improved performance of CTQ identification also results in a boost of the performance of interest identification.

The major contributions of this paper are summarized as follows.

- We propose a two-level interest identification method which first identifies CTQs and then clusters CTQs into interests. This two-level approach is both effective and efficient since CTQs are composed of consecutive queries and hence CTQ identification incurs only a single scan of the query log. The clustering of CTQs into interests is much more efficient than clustering individual queries, since there are much fewer CTQs than individual queries in the search log.
- We propose a novel CTQ identification method, PCTB, which considers three factors, namely, query reformulation pattern, concepts behind the queries, and time threshold. PCTB combines the advantages on these three factors while avoiding their disadvantages. As a result, PCTB is able to achieve the same high precision inherited from query reformation and at the same time high recall resulted from the consideration of concepts in queries.
- We perform extensive experiments to determine which factors affect the precision of recall of CTQ identification most. Experiments show that time threshold and concept expansion are the important factors. Comparing to the existing methods, combining time and concept yields better performance in CTQ identification and consequently better performance in interest identification.

The rest of the paper is organized as follows. Section 2 reviews the related work. In Section 3, we present how to build the concept database. In Section 4, we present our CTQ identification method, PCTB. Experimental results evaluating the performance of our approach against the state-of-the-art methods are presented in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

In this section, we will briefly introduce the previous work related to CTQ and user interest identification.

2.1 Session Segmentation

In [6], Gayo-Avello gave a comprehensive survey on session segmentation. There are several definitions on "session" in the literature. A widely accepted definition of session

is "a series of queries by a single user made within a small range of time". Even though CTQ identification and session segmentation share some similarity, they have three major differences. First, queries in a session must be issued within a short period of time, but CTQs, according to our definition, do not have such requirement. CTQs only take the time gap between two queries as a strong, but not the only, indicator on whether the two queries are topically related. Second, queries in a session are not necessarily topically related. However, CTQs must satisfy this requirement. Third, the goal of the two techniques are different. Sessions are commonly used for constructing a context of the current query to help the search engine to better understand the user's searching needs. CTQs are more focused on the concepts and semantic meaning behind the query, which is used for clustering topically related queries for better query suggestion and document classification.

Existing session detection methods can be classified into three categories, namely *temporal cutoff*, *lexical relation*, and *hybrid*. Temporal cutoff methods consider different time interval thresholds as the factor for detecting sessions. Lexical relation methods are based on the lexical relationship between continuous queries, and hybrid methods combine both temporal cutoff and lexical relation methods in session detection.

Temporal Cutoff Methods. Query time is a useful indicator for detecting new sessions. In He, et. al [7], the idea of session and session boundary was first introduced. The detection of session boundary was based on different time interval thresholds. Since the time gap between two consecutive queries is an important feature in session detection, many temporal cutoff methods have been proposed based on different time interval thresholds (e.g., 5 minutes cutoff [16], $10-15$ minutes cutoff [6] and 30 minutes cutoff [5]).

Lexical Relation Methods. The lexical relationship between consecutive queries has been used as an indicator for session aggregation. Lau, et. al., suggested to use lexical relation to estimate the relationship between two consecutive queries by studying the common keywords between the two queries [9]. Huang, et. al., [8] proposed a more comprehensive query reformulation strategy to identify search queries addressing the same information need. They included more reformulation patterns (e.g., whitespace and punctuation, stemming, substring, etc.) into the existing patterns (e.g., specialization, generalization, reformulation, etc.). However, these pattern-based methods all suffer the semantic rephrasing problem [8]; that is, two queries without common terms could also be topically related in the semantic level.

Hybrid Methods. Buzikashvili and Jansen [3] proposed to first set up a temporal cutoff (e.g., 15 or 30 minutes) to extract preliminary sessions and then applied a lexical comparison method (called transitive closure) to perform further refinement. Sophisticated session detection methods employing different machine learning methods have been proposed. Ozmutlu developed session detection methods based on neural networks [13], multiple linear regression [12], and conditional probabilities [14]. However, most of them used time interval and search pattern as features in the training process, and the only difference between them is the classification model.

2.2 Query Clustering

As discussed in Section 1, interest identification is essentially a query clustering task. Most clustering methods were done based on the clickthrough data of the search engine. Beeferman and Berger [2] proposed an agglomerative clustering algorithm (denoted as BB in this paper) to exploit the query-document relationship from the clickthrough data. BB performs clustering based on a bipartite graph, in which nodes on the left hand side are queries and those on the right hand side are web pages. When a user clicks a page in a query's search result, an edge is created between the page and the query. The similarity between two queries are calculated based on the number of common pages they share. Likewise, the similarity of two pages depends on the number of queries shared by the pages. During the clustering process, query nodes of high similarity are merged together and then pages of high similarity are merged, and so on. The merging/clustering processes are done iteratively until a termination condition is fulfilled. However, due to the sparsity of edges in the graph, the recall is low.

To alleviate the problem, Leung and Lee [10] introduced the concept-based graph model, in which concepts are extracted from web-snippets to build a query-concept bipartite graph upon which a clustering procedure similar to BB is applied. Experiment showed that it could greatly enhance the recall. Baeza-Yates [1] expanded a query with keywords frequently appearing in the the query's clickthroughs and used cosine similarity to calculate query similarity (denoted as "Baeza" in this paper). The uniqueness of our query clustering method is that we identify CTQs before clustering is performed. This approach can fully utilize the continuity of a user's search behavior to boost the recall while maintaining high precision of CTQs.

3 Building Concept Database

In this section, we will briefly introduce how to build the concept database for finding the concepts behind the queries, which will be used in PCTB presented in the next section. When a query q is submitted to a search engine, web-snippets containing brief summaries of the search results are returned to the user. Our concept extraction method assumes that if a keyword (or more generally a key phrase) exists frequently in the web-snippets of the query q, the keyword represents an important concept/topic related to q, as it co-exists in close proximity with the query in the top-ranked documents. To mine the concepts for q, we extract all keywords from the web-snippets of the top 100 results returned from q. Then, the following support formula, which is inspired by the well-known problem of finding frequent item sets in data mining [4], is employed to measure the interestingness of a particular keyword c_i with respect to q:

$$support(c_i) = \frac{sf(c_i)}{n} \cdot |c_i| \tag{1}$$

where $sf(c_i)$ is the snippet frequency of the keyword c_i (i.e., the number of web-snippets containing c_i), n is the number of web-snippets returned and $|c_i|$ is the number of terms in the keyword c_i. If the interestingness of a keyword c_i (i.e., $support(c_i)$) is greater than a threshold s, we treat c_i as a concept for q. This technique has been used

to construct a large-scale concept database by treating the mined concepts as queries to iteratively expand the concept database [11].

The extracted concepts together with the concept similarities retrieved from the query q form the possible concept space arising from q, which will then be stored in our concept database. The concept space, in general, covers the possible topics appeared in the retrieved web-snippets. For example, when the user searches for the query "apple", the concept space derived from the web-snippets can contain different concepts such as "fruit", "iphone" and "itunes". If the user then searches for the query "ipod", the concept space can contain similar concepts such as "iphone", "itunes" and "music". We can immediately see that the two queries "apple" and "ipod" are related, because they retrieve similar concepts, "iphone" and "itunes", in their search results.

Note that all of the concepts are pre-computed, which means the concepts database has already been built before CTQ identification is performed. Thus, when we need the concepts behind a query, we can directly look up the database without incurring much computational cost.

4 CTQ Grouping and Interest Identification

Continuous Topically Related Queries (CTQs) are sets of consecutive queries, and the queries within each CTQ are all topically related. This definition does not require a query in a CTQ to be a reformulation of its preceding query or within a certain time threshold. A CTQ can be considered a basic unit of user interest. A consequence of this definition is that consecutive CTQs are not topically related or else they should be combined in one CTQ. As discussed in Section 2, most of the existing research on session identification are based on query reformulation and/or time threshold. In this paper, we first propose a method for identifying CTQs based on the concepts behind the queries. We adopt the method described in Section 3 for extracting concepts from the web snippets returned from the query and use the concepts as a representation of the query. Then, we introduce a general CTQ identification method called *Pattern-Concept-Time-Based* (PCTB) method, which combines the merits of query reformulation, concept and time based methods while avoiding their drawbacks. In Section 4.1, we first define the important terms used in this paper. In Section 4.2, we introduce PCTB. Section 4.3 explains why PCTB can offer complementary merits. In Section 4.4, we study interest identification based on the clustering of CTQs.

4.1 Definitions

A query log is a sequence of queries $\{q_0, q_1, ..., q_n\}$, where q_0 and q_n are, respectively, the first and the last query in the query log and q_i is issued at time t_i.[2] A user issues a sequence of queries to look for information pertaining to a particular information need. In doing so, the user modifies his/her queries until the information need is fulfilled. When a new information need arises, the user will issue a new sequence of queries to

[2] In general, a query log contains queries from different users. However, since this paper only studies CTQs of a single user, "query log" hereafter refers to the set of query records for a single user.

fulfill the new need. A query that is modified from and sharing the same information goal as the previous query is called a *reformulated query*, whereas a query that starts a new information need is called a *shift query*. Note that a shift query always signifies the beginning of a new CTQ. We also name it as a "CTQ shift".

4.2 Integration of Pattern, Concept and Time in CTQs

We now describe how to integrate query reformulation, concepts and time threshold in our pattern-concept-time-based (PCTB) CTQ grouping method. The details of PCTB is presented in Algorithm 1. The input of the algorithm is a search log containing a set of consecutive queries. The first step of PCTB (lines 1-10 in Algorithm 1) segments a query log into *cliques* by detecting query reformulation performed by the user using traditional query reformulation strategies. In this paper, we apply thirteen different query reformulation strategies, which include "word reorder", "whitespace and punctuation", "remove word", "add word", "url stripping", "stemming", "acronym", "substring", "abbreviation", "word substitution", "spelling correction", etc. Readers are referred to [8] for the details, but the names of these strategies are mostly self explanatory. For example, "word reorder" means some of the words in a previous query are reordered to produce the next query, whereas "whitespace and punctuation" means whitespaces and punctuation marks are added or removed from a query to produce the next query. This step produces a sequence of cliques which have high precision but low recall (see Section 5.2 for the details).

The second step of our method (lines 11-13 in Algorithm 1) extracts concepts from the queries in each clique using the concept extraction strategy introduced in Section 3. Thus, after this step, every clique corresponds to a set of concepts. In the third step of our method (lines 14-19), the conceptual and time similarity between two consecutive cliques is computed using the following similarity metric:

$$\begin{aligned} Sim(cli_i, cli_{i+1}) = \alpha \cdot CosineSim(V(cli_i), V(cli_{i+1}) \\ +(1-\alpha) \cdot TimeSim(cli_i, cli_{i+1}) \end{aligned} \tag{2}$$

$$TimeSim = \begin{cases} 0.9, & \text{if } T_{i+1} < 60\ mins \\ 0.1, & \text{if } T_{i+1} \geq 60\ mins \end{cases} \tag{3}$$

where $V(cli_i)$ and $V(cli_{i+1})$ are the concept vectors for the cliques cli_i and cli_{i+1}, cosine similarity is used in calculating the concept similarity. The time similarity $TimeSim$ is calculated based on Equation 3, where T_{i+1} is the time gap between two consecutive cliques cli_i and cli_{i+1}, α is the weighting factor between $CosineSim$ and $TimeSim$. In traditional session segmentation methods, different time thresholds have been employed, with typical choices being 15, 30, or 60 minutes. Since our goal is to boost the recall, we choose a large time threshold, namely, 60 mins. In order to estimate the time similarity function, we pick 30% of the ground truth CTQs as validation set to see how CTQs are distributed in different time ranges. Figure 1 shows the distribution. Since the AOL log spans 3 months, the largest time range is 3 months. We can observe that 70% of the CTQs have time gap less than 5 minutes, and about 90% percents have time gap less than 60 minutes.

Algorithm 1. Concept-Based CTQ Identification

Input: Search engine log data L in the form of $(userID, query, URL, timestamp)$

Output: CTQs, each CTQ is a set of entries in the form of $(userID, query, URL, timestamp, concepts, label)$. CTQ will always start with label "false" and contain only one "false".

```
i=1, q_1=false
for each pair of consecutive queries q_i and q_{i+1} in L; do
    if q_{i+1} is a reformulated query of q_i then
        Label q_{i+1} as true;
    else
        Label q_{i+1} as false;
    end if
    i ← i+1
end for
Merge into clique cli_i a sequence of consecutive queries that begins with a query labelled as
"false", followed by zero or more queries labelled as "true" until a query labelled as "false"
or the end of the query search log is met.
for each clique cli_i in L do
    Search the concept database to retrieve the concepts of queries in c_i with the method
    introduced in Section 3; {Each clique can be represented by a vector of concepts}
end for
for each pair of consecutive cliques cli_i and cli_{i+1} in L do
    Calculate the similarity between cli_i and cli_{i+1} using Equation (2).
    if Sim(cli_i, cli_{i+1}) > threshold then
        concatenated cli_i and cli_{i+1}, by reversing the first query of cli_{i+1} from "false" to
        "true";
    end if
end for
```

According to this observation, we set $TimeSim$ to 0.9 if the two consecutive cliques fall within the 60-min time threshold, and 0.1, otherwise. If the overall similarity between two consecutive cliques is larger than a threshold, they are concatenated into one CTQ.

4.3 Benefits of Concept Based CTQ Identification

Traditional query reformulation strategies usually have high precision but low recall. In Huang, et. al. [8], query reformulation strategies can obtain a high precision of 98.2%. We implement their strategies on our own dataset containing 11484 search records, and obtain a precision of 97%, which is consistent with the result reported in [8]. However, the recall is only 61.3%, which is quite low. In [8], the low recall was attributed to the *semantic rephrasing problem*. That is, some queries are semantically the same but they cannot be identified with rules (e.g., morphological rules) or lexical databases (e.g., Wordnet). In PCTB, we use the existing reformulation strategies to segment a query log into *cliques*, which can be considered preliminary groups that need to be refined further. Due to the high precision of the query reformulation strategy, we do not lose accuracy in this step. Then, we apply our concept mining and time similarity to identify consecutive cliques that are judged to be topically related and concatenate them together.

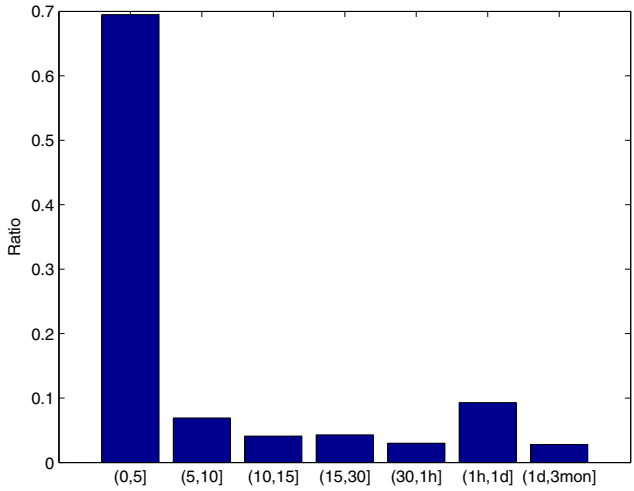


Fig. 1. Queries distribution within the ground truth CT query groups

This step can greatly improve the recall of traditional methods based on query reformulation alone, and hence improve the precision of our CTQ identification task. Concept similarity is calculated by cosine similarity, which is widely used in vector space model to measure the similarity between two documents, whereas time similarity originates from time-based session segmentation methods.

4.4 Interest Identification Based on CTQs

Interest identification aims at clustering all topically related queries in the search log together, whether the queries are consecutive or not. Since a CTQ already represents a single topic, we propose to cluster CTQs with similar topics together to form an interest instead of clustering the individual queries. There are two alternatives to cluster topically similar CTQs into interests. First, since a CTQ is represented by a set of concepts, we propose to construct a CTQ-concept bipartite graph, in which nodes on one side represent CTQs, nodes on the other side represent concepts and edges connect CTQs to their underlying concepts. This is similar to the query-concept bipartite graph in the "QC" method (which is similar to [10] and explained in Section 2). Scattered topically related CTQs can be grouped together based on their common concepts. Since the idea is an extension of QC, we name it "group+QC". The second method is to apply cosine similarity on the concept vectors of the CTQs and cluster CTQs of high similarity into an interest. We name it "group+cos". Finally, as a baseline in performance comparison, we also implement the *pure cosine* clustering method, in which two queries are assigned to the same cluster if the similarity of their concept vectors is larger than a threshold. In general, the high accuracy of CTQs benefits the accuracy of interest identification. More importantly, since clustering on large datasets is very time consuming, clustering methods based on CTQs instead of individual queries can reduce the computational

cost by orders of magnitude. Although the extension is intuitive, CTQs significantly improves the performance of interest identification.

After applying the above clustering techniques, we can aggregate topically related queries from different CTQs together forming larger query clusters representing user interests. User interests can be used to predict the user's future searches or deliver personal (e.g. advertising, news delivery) services according to the interests. Details on the applications of user interests and their performance are beyond the scope of this paper and will be investigated in our future research.

5 Experimental Results

In Section 5.1, we first describe the experimental setup for building the ground truth and collecting experimental data. In Section 5.2, we compare the performance of six CTQ identification methods, namely, time-based (TB), search-pattern-based (PB), time-pattern-based (TPB), concept-based (CB), pattern-concept-based (PCB), and pattern-concept-time-based (PCTB), some of which are traditionally used in session segmentation. PCTB is the method proposed in this paper, while the other methods serve as the baselines in the comparison. In Section 5.3, we apply our CTQ identification method, namely, PCTB, in clustering CTQs to find the user's interests.

5.1 Experimental Setup

We use a subset of AOL search engine log data which contains $11,484$ search records as our experimental data set. Each record contains attributes such as user ID, query, time stamp, and URLs that the user has clicked. In order to evaluate the performance of the CTQ identification methods, we need to create the ground truth, which is the ideal CTQs obtained from the search log with human judgment. The preliminary ground truth was built by Gayo-Avello [6], who segmented the search log manually based on human judgment. The ground truth was manually built by considering the queries, clickthroughs as well as the time distance between queries. This results in 4,253 sessions in the ground truth. Since in our study we require the queries in a CTQ to be topically related but there is no threshold requirement on the time gap between the queries, we went through the ground truth built by Gayo-Avello and concatenated consecutive, semantically related sessions into one CTQ regardless of the time gaps between them. As a result, 197 sessions were combined, resulting in 4,056 groups in our ground truth.

To perform concept extraction, we implemented a Google middleware. When a query is submitted to the middleware, the top 100 search results from Google are retrieved, and the important terms are extracted according to the concept extraction method proposed in Section 3. Table 2 shows the statistics of our experimental dataset.

5.2 Evaluation of CTQ Identification Methods

We now compare the performance of six CTQ identification methods, namely TB, PB, TPB, CB, PCB and PCTB.

Table 2. Statistics of Dataset Used in Experiments

No. of queries	11,484
No. of users	215
Avg. no. of queries issued by a user	53.41
No. of concepts retrieved	226,601
Avg. no. of concepts retrieved for a query	19.73
No. of CTQs	4056
Avg. no. of queries per CTQ	2.83
Avg. no. of concepts per CTQ	55.87
No. of search patterns	15

- Time-based method (TB): TB purely relies on the time gap between two consecutive queries and serves as the baseline. If the time gap exceeds a certain threshold, a new CTQ is formed.
- Pattern-based method (PB): PB identifies CTQs purely based on query reformulation patterns, which is the strategy used by Huang, et. al. [8].
- Time-Pattern-based method (TPB): TPB is a combination of TB and PB.
- Concept-based method (CB): CB identifies CTQs based on the similarity of the concepts extracted for the queries.
- Pattern-Concept-based method (PCB): PCB is a combination of PB and CB.
- Pattern-Concept-Time-based method (PCTB): PCTB is a combination of PB, CB and TB (introduced in Section 4.2).

We employ the six methods to segment the continuous AOL search log into CTQs. The results are evaluated against the ground truth to obtain precision, recall and F-measure. Precision is defined as the number of correctly identified CTQ shifts divided by the total number of identified CTQ shifts. Recall is the number of correctly identified CTQ shifts divided by the number of correct CTQs in ground truth. F-measure is an overall performance measure defined on precision and recall and is defined as $(2 \times precision \times recall)/(precision + recall)$. It is important to note that in our definition of precision and recall the goal is to correctly identify all CTQ shifts (i.e., shift queries) without any mistake (i.e., misidentifying a query that is topically related to its preceeding query as a shift query), whereas the query reformulation strategies in Huang, et. al., [8] try to find sequences of semantic consistent queries. Thus, the precision and recall values obtained in our experiments should be interpreted differently from that of the traditional work.

Figure 2 shows the performance of the six CTQ identification methods at their optimal cutoff thresholds. In the comparison, TB is used as the baseline. As discussed in Section 4.2, PB suffers from the semantic rephrasing problem, meaning that it is hard to identify a semantically related but syntactically different query as a reformulated query, however this can help identify shift query. According to our definition of precison and recall in our CTQ identification background, the misidentification will lower precision but does not hurt recall much since se- mantic shift queries must not be reformulations of their preceding queries, that is why PB achieves high recall. This property can be observed in Figure 2, which shows a low precision and high recall for PB. Although TPB considers the time factor in addition to query reformulation, it has a performance

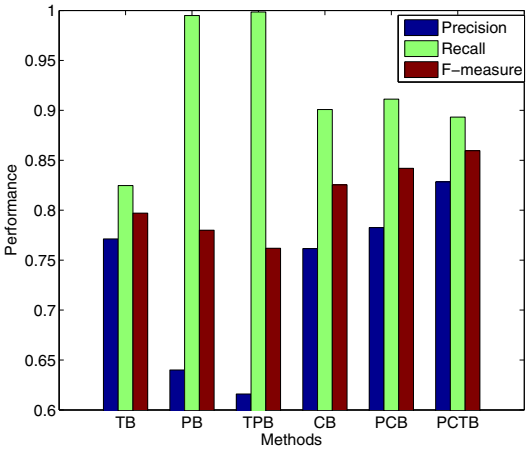


Fig. 2. Performance of different CT queries grouping methods

similar to PB, meaning that time is not a strong factor in deciding CTQs. We can see that CB can effectively boost recall, because semantically similar queries can be easily identified by comparing the concept spaces of two consecutive queries. By combining PB and CB, PCB achieves both good precision and recall. Finally, we observe that by adding the time factor into PCB, we further improve the performance of PCTB, which is the best method with the highest F-measure.

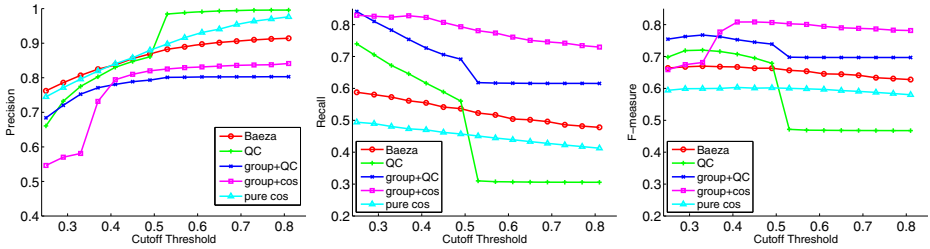
There are two variables in CTQ identification, which are α in Equation 2 and the similarity threshold. In order to check how the performance is affected by these two variables, we collect 11 performance points at different cutoff thresholds, ranging from 0.25 to 0.75 with 0.05 increment in each step, and 8 different α values. Thus, there are 8*11 combinations in total. For clarity, we do not list all of the combinations. For each α value, we instead pick five values, which are the highest, lowest, mean, standard deviation, and the number of points with F-measure values larger than 0.8. Thus, there are 8*5 combinations in total, which are shown in Table 3. We can see from the table that if we purely rely on time similarity, which is TB (i.e. $\alpha = 0$), we obtain a much lower F-measure of 0.77. This shows that a pure time-based similarity function is not optimal for CTQ identification. Moreover, when the similarity function is biased towards the concept side (e.g., α = 0.6 and α = 0.8), a larger number of points with F-measure larger than 0.8 are obtained, meaning that the overall effectiveness is higher.

5.3 Interest Identification

In this section, we compare the performance of "QC", "Baeza", "group+QC", "group+cosine" and "pure cosine". These methods have been introduced in Section 2 and Section 4.4. The five methods are used to cluster queries into topically related queries forming the interests of the user (see 5.1). To create the ground truth for this experiment, we examine the ground truth CTQs obtained in the previous experiment

Table 3. F-measure Statistics at Different α

α	Highest	Lowest	Mean	S.D.	> 0.8
0.8	0.848	0.813	0.839	0.008	11
0.6	0.854	0.687	0.820	0.054	10
0.4	0.861	0.598	0.789	0.077	7
0.2	0.858	0.598	0.771	0.067	4
0.15	0.851	0.738	0.776	0.036	3
0.1	0.858	0.738	0.776	0.036	3
0.05	0.834	0.738	0.766	0.024	1
0.0	0.767	0.761	0.762	0.003	0



(a) Precision (b) Recall (c) F-measure

Fig. 3. Performance comparison of different Query Clustering methods

(see Section 5.1) and manually label CTQs on the same topic but distributed across the search log as an interest. There are totally 2,345 interests in our ground truth and on average 4.9 queries per interest. Clustering results are compared to the ground truth to compute the precision and recall, which are defined as follows.

$$precision(q) = \frac{|Q_{relevant} \cap Q_{retrieved}|}{|Q_{retrieved}|} \tag{4}$$

$$recall(q) = \frac{|Q_{relevant} \cap Q_{retrieved}|}{|Q_{relevant}|} \tag{5}$$

where $Q_{relevant}$ is the set of queries that exist in the predefined cluster for q, and $Q_{retrieved}$ is the set of related queries generated by a query clustering algorithm. The precision and recall values from all queries are averaged to plot the performance graphs.

Figure 3 shows the precision, recall and F-measure of the five methods under evaluation, namely, "QC", "Baeza", "group+QC", "group+cos", "pure cos" with different similarity cutoff thresholds. By comparing QC with group+QC and pure-cos with group+cos in Figure 3(c), we can observe that clustering based on CTQs can consistently and significantly improves the effectiveness of interest identification. Since our CTQ identification method, PCTB, can discover CTQs with high precision and recall as shown in Section 5.2, both group+QC and group+cos benefit from the high precision and recall of PCTB. Thus, the group-based methods perform better than non-group

Table 4. Performance of Different Query Clustering Methods

Method	Precision	Recall	F-measure
QC	0.775	0.672	0.720
Baeza	0.807	0.573	0.670
group+QC	0.753	0.783	0.767
group+cos	0.810	0.807	0.808
pure cos	0.841	0.470	0.603

based methods in identifying user interests. Finally, we can observe that Baeza's performance is somewhere in the middle of the five methods. Although it is non-CTQ based, it performs better than the other two non-CTQ based methods, pure cosine and QC.

Finally, we observe that "group+cos" outperforms "group+QC" in Figure 3(c). This may be attributed to the difference between the two methods' similarity functions. In cosine similarity, the similarity is calculated based on common words as well as their corresponding concept weights. However, in QC, two queries are similar only because they share a certain number of common concepts, but the weights of the concepts are not considered.

Figure 3(a) and 3(b) depict the precision and recall values of the five methods with different cutoff thresholds. With the increase of the cutoff threshold, more evidence is needed in order to merge two clusters. Thus, the precision becomes higher, and the correspondingly recall becomes lower. We can see that CTQ-based clustering methods perform very well with high recall values. Again, since PCTB can effectively cluster consecutive and similar queries together, we can expand the hidden semantics from a single query to a group of similar queries effectively. By exploiting the similarities between CTQs, we can effectively identify and group semantically related queries together, and thus improving the recall of the proposed group-based clustering methods (i.e., "group+cos" and "group+QC").

Table 4 shows the best F-measure values for "QC", "Baeza", "group+QC", "group+cos", and "pure cos" methods. From the results, we can see that CTQ-based methods perform better compared to the other two baseline methods. Furthermore, the complexity of CTQ-based methods is much less compared to QC and Baeza.

6 Conclusions

This paper develops a method for identifying sets of continuous topically related queries, called CTQs. The method integrates three techniques that are based on reformulation patterns, concepts and time thresholds of the queries. Thus, it is called *Pattern-Concept-Time-Based* (PCTB) method. We show that PCTB can integrate the merits of both traditional query reformulation and segmentation methods based on time thresholds. PCTB takes into account the overlap of the concepts behind the queries to group semantically related queries together. To evaluate the effectiveness of our approach, we employ the AOL search log together with a Google middleware for concept extraction. Experimental results show that our approach yields better precision and recall comparing to the existing CTQ identification methods. We apply PCTB to identify user interests by

clustering queries sporadically distributed in the query log but are topically related into user interests. We propose to first identify the CTQs and then cluster the CTQs to produce user interests. Since PCTB produces high-quality CTQs, the subsequent clustering process becomes much more efficient and effectiveness.

As for the future work, we plan to study the effectiveness of applying the identified user interest to some real applications, for example, advertisement targeting, personal news delivery, and so on.

References

1. Baeza-Yates, R., Hurtado, C.A., Mendoza, M.: Query recommendation using query logs in search engines. In: Lindner, W., Fischer, F., Türker, C., Tzitzikas, Y., Vakali, A.I. (eds.) EDBT 2004. LNCS, vol. 3268, pp. 588–596. Springer, Heidelberg (2004)
2. Beeferman, D., Berger, A.: Agglomerative clustering of a search engine query log. In: Proc. of the SIGKDD Conference (2000)
3. Buzikashvili, N., Jansen, B.J.: Limits of the web log analysis artifacts. In: Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection. WWW Conference (2006)
4. Church, K.W., Gale, W., Hanks, P., Hindle, D.: Using statistics in lexical analysis. Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon (1991)
5. Downey, D.: Models of searching and browsing: languages, studies and applications. In: Proc. of the IJCAI Conference (2007)
6. Gayo-Avello, D.: A survey on session detection methods in query logs and a proposal for future evaluation. Information Sciences, 1822–1843 (2009)
7. He, D., Göker, A., Harper, D.J.: Combining evidence for automatic web session identification. Information Processing Management 38 (2002)
8. Huang, J., Efthimiadis, E.N.: Analyzing and evaluating query reformulation strategies in web search logs. In: CIKM (2009)
9. Lau, T., Horvitz, E.: Patterns of search: analyzing and modeling web query refinement. In: Proc. of the UM Conference (1999)
10. Leung, K., Lee, D.: Deriving concept-based user profiles from search engine logs. IEEE Transactions on Knowledge and Data Engineering 99(1) (2007)
11. Leung, K.W.-T., Fung, H.Y., Lee, D.L.: Constructing concept relation network and its application to personalized web search. In: Proceedings of the 14th International Conference on Extending Database Technology, EDBT/ICDT 2011, pp. 413–424 (2011)
12. Ozmutlu, H.C., Ozmutlu, F.S.: Automatic new topic identification in search engine transaction logs (2006)
13. Ozmutlu, H.C., Ozmutlu, S.: Cross-validation of neural network applications for automatic new topic identification. American Society for Information Science and Technology (2008)
14. Ozmutlu, S., Ozmutlu, H.C., Buyuk, B.: Using conditional probabilities for automatic new topic identification. Online Information Review (2007)
15. Pass, G., Chowdhury, A., Torgeson, C.: A picture of search. In: Proc. of the INFOSCALE Conference (2006)
16. Silverstein, C., Marais, H., Henzinger, M., Moricz, M.: Analysis of a very large web search engine query log. SIGIR Forum, 6–12 (1999)

Efficient Responsibility Analysis for Query Answers

Biao Qin, Shan Wang, and Xiaoyong Du

School of Information, Renmin University of China, Beijing 100872, China
{qinbiao,swang,duyong}@ruc.edu.cn

Abstract. Provenance information describes the origins and the history of data in its life cycle. Responsibility captures the notion of degree of causality and tells us which facts are the most influential in the lineage. Since responsibility cannot be computed by a relational query, the analysis of lineage becomes an essential tool to compute responsibility of tuples in the query results. We extend the definitions of causality and responsibility of a tuple t for the answer r to those of a set of tuples for the answer r, and Co-Trees to P-Trees for read-once functions. By using P-Trees, we develop an efficient algorithm to compute responsibilities of tuples in read-once formulas, and a novel algorithm to find top-k responsibility tuples in read-once functions. Finally, experimental evaluation on TPC-H data shows substantial efficiency improvement when compared to the state of the art.

1 Introduction

Causality in databases aims to answer the following question: given a query over a database instance and a particular output of the query, which tuple(s) in the instance caused that output to the query? So causality is related to provenance [4,11], yet it is a more refined notion. A formal, mathematical study of causality has been initiated by the work of Halpern and Pearl [13]. Chockler and Halpern [5] introduced an extension of causality called degree of responsibility that allows us to do a more fine-grained analysis of causality. Since then, causality and responsibility become hot in computer science research community [8,6,20,22].

According to causality [13], variables are partitioned into exogenous and endogenous. Intuitively, exogenous variables define a context determined by external, unconcerned factors, deemed not to be possible causes, while endogenous variables are the ones who affect the outcome and are thus potential causes. Moreover, the endogenous variables are ones whose values are ultimately determined by the exogenous variables. In a database setting, variables are tuples in the database. An endogenous tuple t is a cause for the query results only if there is a hypothetical setting of the other endogenous tuples under which the addition/removal of t causes the query results to change. Therefore, contingency is a set of endogenous tuples which relate to tuple t. If we remove them from the database, then t immediately affects the answer in the new state of the database. In theory, in order to compute the contingency one has to iterate over

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 239–253, 2013.

subsets of endogenous tuples. So it is NP-complete to check causality in general [8]. However, Meliou et al. [22] have shown that the causality of conjunctive queries can be determined in PTIME, and moreover, all causes can be computed by a relational query. Causality is related to lineage of query results, such as why-provenance [7] or where-provenance [2], and very recently, explanations for non-answers [27,3,15].

The definition of responsibility refines the "all-or-nothing" concept of causality by measuring the degree of causality as a function of the size of the smallest contingency set. In applications involving large datasets, it is critical to rank the candidate causes by their responsibility, because answers to complex queries may have large lineages and large numbers of candidate causes. In theory, in order to compute the responsibility one has to iterate over all contingency sets. Hence it is NP-hard to compute responsibility in general [5]. However, Meliou et al. [22] have proved that the complexity depends on the conjunctive query and established a dichotomy between PTIME and NP-complete cases. The class of PTIME queries is called linear queries. If you are interested in linear queries, please refer to [22]. For the PTIME cases, they gave a non-trivial algorithm, consisting of a reduction to the max-flow computation problem. Moreover, they have proved that responsibility cannot be computed by a relational query. So responsibilities must be computed by data-centric techniques. A Boolean formula Φ is read-once [14] if it can be factorized into a form where every Boolean variable appears at most once.

Example 1. If the following query evaluates over databases shown in Figure 1,

$$q_1 : -R_1(A), R_2(B), R_3(C), R_4(A, B, C, D)$$

The lineage of the answer tuple is shown as follow:

$$\begin{aligned} \Phi &= a_1b_1c_1d_1 + a_2b_2c_2d_2 + a_2b_2c_3d_3 \\ &= a_1b_1c_1d_1 + a_2b_2(c_2d_2 + c_3d_3) \end{aligned} \tag{1}$$

Since q_1 is not a linear query, Meliou's algorithm can not perform responsibility analysis for Φ in PTIME. Because each variable occurs at most once, Equation (1) is a read-once formula.

In this paper, we investigate a method to efficiently compute the responsibility for read-once formulas. In order to compute responsibilities of tuples in read-once formulas, we extend Co-Tree [10] to P-Tree with two additional fields for each node. The main contributions of this paper are as follows.

- We extend the definitions of causality and responsibility of a tuple t for the answer r to those of a set of tuples for the answer r, and Co-Trees to P-Trees for read-once formulas.
- By using P-Trees, we devise an efficient PTIME algorithm to compute the responsibilities of tuples in read-once expressions,
- We introduce a novel algorithm to find top-k responsibility tuples in read-once formulas.

R_1

tid	A
a_1	2
a_2	4

R_2

tid	B
b_1	1
b_2	2

R_3

tid	C
c_1	2
c_2	4
c_3	6

R_4

tid	A	B	C	D
d_1	2	2	1	1
d_2	4	4	2	2
d_3	4	6	2	3

Fig. 1. An example database

The rest of this paper is organized as follows: In the next section, we outline query cause and responsibility. In Section 3, we develop an algorithm to compute responsibilities of tuples in read-once functions. In Section 4, we introduce a novel algorithm to find top-k responsibility tuples. In Section 5, we report the experimental results. In Section 6, we discuss related work before concluding with Section 7.

2 Preliminaries

Assume a standard relational schema includes relation $R_1, \cdots, R_k$. We write D for a database instance and q for a query. $D = D^n \cup D^x$, where D^n and D^x represent endogenous and exogenous tuples in D, respectively. $R_i = R_i^n \cup R_i^x$, where R_i^n and R_i^x denote the endogenous and exogenous tuples in R_i, respectively. Meliou et al. [22] adapted the concept of causality to Boolean query as follows: A tuple t is a counterfactual cause if its removal from the database makes q false. A tuple t is an actual cause for result tuple r of q in D if it has a contingency $\Gamma \subseteq D^n$, such that t is a counterfactual cause for r in $D - \Gamma$. Obviously, if we set $\Gamma = \emptyset$, then every counterfactual cause is also an actual cause. The causality definitions [22] can be extended to sub-formulas Φ_i as follows.

Definition 1. *Assume that the lineage of an answer tuple r forms a formula Φ. Let $\Phi_i \in \Phi$ and $\phi_{ij} \in \Phi_i$ and $Vars(\Phi)$ denote all tuples involved in Φ.*

- *A set of tuples $Vars(\phi_{ij})$ is a counterfactual cause with respect to Φ_i, if its removal from the database also removes Φ_i from Φ.*
- *A set of tuples $Vars(\phi_{ij})$ is an actual cause with respect to Φ_i if one can find a contingency under which it becomes a counterfactual cause: more precisely, one has to find a set Γ such that, after removing Γ from the database we bring it to a state where removing/inserting $Vars(\phi_{ij})$ causes Φ_i to switch between $\Phi_i \in \Phi$ and $\Phi_i \notin \Phi$.*

The responsibility definition [22] can be extended to sets directly as follows:

Definition 2. *Let r be an answer or non-answer to a query q, and let a set of tuples $Vars(\Phi_i)$ be a cause. The responsibility of $Vars(\Phi_i)$ for the answer r is*

$$resp(\Phi_i) = \frac{1}{1 + MCS(\Phi_i)}$$

where $MCS(\Phi_i) = min_\Gamma|\Gamma|$ and Γ ranges over all contingency sets for $Vars(\Phi_i)$.

We will illustrate how to compute $MCS(\Phi)$ in Example 3 later. The co-occurrence graph G_c [12] of a Boolean formula Φ is a undirected graph whose set of nodes $V(G_c)$ is the set of variables of Φ and whose set of edges $E(G_c)$ is defined as follows: there is an edge between x and y iff both occur in the same conjunct of Φ. Using the co-occurrence graph, Golumbic et al. [10] gave a fast algorithm that takes as input a formula in irredundant disjunctive normal form, decides whether it is read-once, and if it is, computes the read-once form. Traditionally, Co-Trees [10] have been used to represent read-once formulas. Co-Trees are trees where leaves correspond to Boolean variables while internal node $\circledast$ represents $\wedge$ and $\bigoplus$ represents $\vee$.

3 Responsibility Analysis of Read-Once Formulas

It is NP-hard to compute the degree of responsibility for general Boolean formulas [5]. Based on a new data structure P-Tree, we develop an efficient algorithm called *PResp* to compute the responsibilities of all variables in read-once formulas in P-TIME. We describe the two steps of *PResp* one by one as follows.

3.1 Building P-Trees for Read-Once Formulas

In order to compute responsibility, we introduce a new data structure P-Tree to represent read-once expressions. Similar to Co-Tree, nodes of P-Tree have the following properties:

- Every leaf node is a distinct variable.
- Internal node $\circledast$ represents $\wedge$ and $\bigoplus$ represents $\vee$.
- In P-Tree, $\circledast$ and $\bigoplus$ alternate along every path.

The key technique of Golumbic's algorithm [10] is to recursively partition a co-occurrence graph G_c into conjunction and disjunction partitions alternately and implement a function ρ that maps graphs to read-once formulas. It takes the G_c of a DNF formula as input. If G_c can be divided into disjunction partitions $G_1, \ldots, G_n$ by calculating its connected components, then $\rho(G_c) = \rho(G_1) + \ldots + \rho(G_n)$. Otherwise, we compute the complement of G_c and check again for connected components $G_1, \ldots, G_m$. Then, $\rho(G_c) = \rho(G_1) * \ldots * \rho(G_m)$. We recursively execute with smaller graphs and check for connected components in these graphs and their complements. Before any complementation, the formula is obtained as the disjunction of the subformulas for the connected components. With each complementation, we alternate conjunctions and disjunctions until one of the following cases happens:

- The connected component is a trivial graph, that is, the connected component is a node.
- The complement of a connected component is also a connected graph. In this case, this partition of subgraph can not be mapped into a read-once formula. So the formula can not be transformed into a read-once function.

Since a P-Tree can be mapped into a read-once formula, Φ is mapped to the root node of its P-Tree. Furthermore, a partition of a P-Tree can be mapped into a subformula. So partition and subformula can be used interchangeably in this paper. Every node and its children form a partition in a P-Tree. In order to remember the property of a subformula, we give the following definition.

Definition 3. *In order to make a subformula false, we call the minimal set of variables that should be set to false the minimal false set, which is denoted by MFS. We further denote the size of the minimal false set of a subformula Φ_i by $MFS(\Phi_i)$.*

The difference between $MCS(\Phi_i)$ and $MFS(\Phi_i)$ is that the former represents the minimal number of variables outside $Vars(\Phi_i)$ should be set to false, while the latter denotes the minimal number of variables inside $Vars(\Phi_i)$ should be set to false. Since there are $\oplus$ and $\circledast$ nodes in a P-Tree, we use Φ_+ to denote a partition from $\oplus$ and Φ_* to denote a partition from $\circledast$, respectively. For any node v_i of a P-Tree, $MFS(v_i)$ means the size of the minimal false set of a partition from v_i, and $Vars(v_i)$ means all variables of a partition from v_i. Different from Co-Tree, every node v_i of P-Tree has three fields, which are shown in Figure 2(a). The first field is type which denotes a tuple t, $\oplus$, or $\circledast$. The second field is $MFS(v_i)$. The final field is $MCS(v_i)$. For any node v_i, we use its type to identity it. So different from building Co-Tree, we should compute $MFS(v_i)$ and $MCS(v_i)$ of all nodes in a P-Tree. The following theorem introduces a technique to compute $MFS(v_i)$ of all nodes v_i in a P-Tree bottom-up.

Theorem 1. *There are three kinds of nodes v_i in a P-Tree. We compute their respective $MFS(v_i)$ as follows.*

1. If the first field of a node v_i is a tuple t, then $MFS(t)$ is equal to 1.

2. If the first field of a node v_i is $\circledast$, then $MFS(\circledast) = min(MFS(\Phi_1), \cdots, MFS(\Phi_m))$ for all sub-partitions $\Phi_j \in \Phi_*$.

3. If the first field of a node v_i is $\oplus$, then $MFS(\oplus) = \sum_{j=1}^{m} MFS(\Phi_j)$ for all sub-partitions $\Phi_j \in \Phi_+$.

Proof. 1. In a P-Tree, tuples are only in the leaf nodes. For each leaf node, the partition includes only one tuple. Thus $MFS(t)$ is equal to 1.

2. If the first field of a node v_i is $\circledast$, then the partition Φ_* can be divided into m conjunctive sub-partitions. So $\rho(\Phi_*) = \rho(\Phi_1) * \ldots * \rho(\Phi_m)$. If we set any one of its sub-partition Φ_j to false, Φ_* is also set to false. Thus $MFS(\circledast) = min(MFS(\Phi_1), \cdots, MFS(\Phi_m))$ for all sub-partitions $\Phi_j \in \Phi_*$.

3. If the first field of a node v_i is $\oplus$, then the partition Φ_+ can be divided into m disjunctive sub-partitions. So $\rho(\Phi_+) = \rho(\Phi_1) + \ldots + \rho(\Phi_m)$. If we set all its sub-partitions to false, then Φ_+ is also set to false. Thus $MFS(\oplus) = \sum_{j=1}^{m} MFS(\Phi_j)$ for all sub-partitions $\Phi_j \in \Phi_+$.

This proves the theorem.

By Theorem 1, we introduce Algorithm 1 to build P-Trees for Boolean formulas. The following theorem proves Algorithm 1 can build P-Tree and compute correct $MFS(v_i)$ of all nodes in a P-Tree.

Algorithm 1. Building-PTree(G_c, *Root*, *disjunct* = *true*)

```
1: create a node v_i;
2: if (G_c is a trivial graph) then
3:    type(v_i) = t and MFS(v_i) = 1
4: else
5:    type(v_i) = !disjunct? ⊛: ⊕;
6:    for (every connected component G[i] of G_c) do
7:       convert G[i] to its complement graph Ḡ[i];
8:       M[i] = Building-PTree(Ḡ[i], v_i, !disjunct);
9:       MFS(v_i) = !disjunct ? min(M[0], ..., M[n]) : SUM(M[0], ..., M[n]);
10:   end for
11: end if;
12: add v_i to the child of Root;
```

Theorem 2. *If a formula Φ can be compiled into a read-once form, Algorithm 1 can build its P-Tree from its co-occurrence graph and compute correct $MFS(v_i)$ of all nodes v_i.*

Proof. If a formula Φ can be compiled into a read-once function, the technique of Golumbic et al. [10] ensures that it can be built into a Co-Tree. Our P-Tree is similar to Co-Tree besides two additional fields for every node. Thus our algorithm can build a P-Tree for the formula. Moreover, Theorem 1 ensures that Algorithm 1 can compute correct $MFS(v_i)$ of all nodes in a P-Tree. This proves the theorem.

We prove the time complexity of the Building-PTree algorithm as follows.

Theorem 3. *For a formula Φ which can be compiled into a read-once function, Algorithm 1 can build its P-Tree from its co-occurrence graph and compute $MFS(v_i)$ of all nodes in the time complexity of $O(\alpha N^2)$, where α is the number of tables in the query and N is the number of nodes in the co-occurrence graph.*

Proof. In every recursive iteration of Algorithm 1, it computes the connected components of the co-occurrence graph whose time complexity is $O(N^2)$. It converts every connected component into its complement graph whose time complexity is $O(N^2)$.

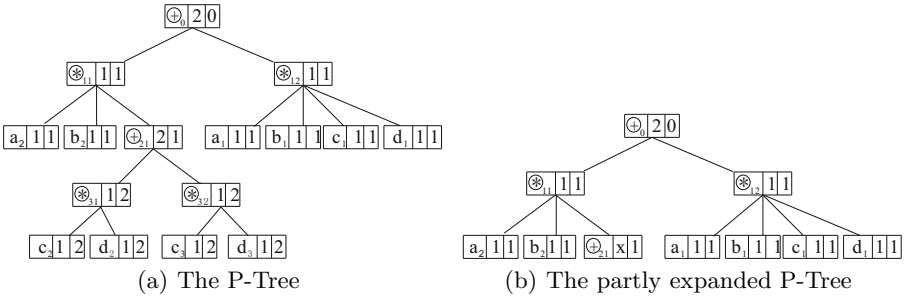
In a P-Tree, the conjunction and disjunction nodes occur alternately. So the algorithm iterates at most 2α times. Therefore, the time complexity of Algorithm 1 is $O(\alpha) * (O(N^2) + O(N^2)) = O(\alpha N^2)$.

The following example demonstrates the technique to build a P-Tree and compute $MFS(v_i)$ of all nodes in the P-Tree.

Example 2. Continuing the running Example. We build its P-Tree as follows. From Equation (1), we know that $\Phi = a_2b_2(c_2d_2 + c_3d_3) + a_1b_1c_1d_1$. Let $\Phi_1 = a_2b_2(c_2d_2 + c_3d_3)$ and $\Phi_2 = a_1b_1c_1d_1$. Since $Vars(\Phi_1) \cap Vars(\Phi_2) = \emptyset$, the G_c of Φ has two connected components $G[1]$ for Φ_1 and $G[2]$ for Φ_2. We first create

a disjunction node $\oplus_0$ which represents Φ and then convert $G[1]$ and $G[2]$ into their respective complement graphs $\overline{G}[1]$ and $\overline{G}[2]$. We then use $\overline{G}[1]$ and $\overline{G}[2]$ as input and build a P-Tree top-down. We create a conjunction node $\circledast_{11}$ to represent Φ_1 and add it to the child of $\oplus_0$. Then we break $\overline{G}[1]$ into three connected components. Since its components a_2 and b_2 are both trivial graphs, we create a leaf node for every variable and then add them to the children of $\circledast_{11}$. At the same time, we create a node $\oplus_{21}$ and then add it to the child of $\circledast_{11}$. We recursively build its children. With the similar method, we can build the right branch of $\oplus_0$. The P-Tree is shown in Figure 2(a).

We use the left branch of the P-Tree as an example to compute $MFS(v_i)$ of all nodes bottom-up. Since c_2, d_2, c_3 and d_3 are all leaf nodes, $MFS(c_2) = MFS(d_2) = MFS(c_3) = MFS(d_3) = 1$. Because $\circledast_{31}$ is a conjunctive node, $MFS(\circledast_{31}) = min(MFS(c_2), MFS(d_2)) = 1$. With a similar method, $MFS(\circledast_{32}) = 1$. Since $\oplus_{21}$ is a disjunctive node, $MFS(\oplus_{21}) = MFS(\circledast_{31}) + MFS(\circledast_{32}) = 2$. Because a_2 and b_2 are both leaf nodes, $MFS(a_2) = MFS(b_2) = 1$. Since $\circledast_{11}$ is a conjunctive node, $MFS(\circledast_{11}) = min(MFS(a_2), MFS(b_2), MFS(\oplus_{21})) = 1$.



(a) The P-Tree (b) The partly expanded P-Tree

Fig. 2. The P-Tree and its partly expanded P-Tree of Equation (1)

3.2 Computing Responsibilities of Tuples

Theorem 4. *Assume that a P-Tree has a node v_i and its parent node v_j. We compute $MCS(v_i)$ as follows:*

1. *For a root node v_j, $MCS(v_j)$ is equal to 0.*
2. *If the first field of v_j is $\circledast$, then $MCS(v_i) = MCS(v_j)$.*
3. *If the first field of v_j is $\oplus$, then $MCS(v_i) = MCS(v_j) + MFS(v_j) - MFS(v_i)$.*

Proof. 1. For a root node v_j, $Vars(v_j)$ is a counterfactual cause with respect to r. So $MCS(v_j) = 0$.

2. If the first field of v_j is $\circledast$, v_j can be divided into m conjunctive partitions. So $\rho(v_j) = \rho(v_1) * \cdots \rho(v_i) \cdots * \rho(v_m)$. Thus $MCS(v_i) = MCS(v_j)$.

3. If the first field of v_j is $\oplus$, then v_j can be divided into m disjunctive partitions. So $\rho(v_j) = \rho(v_1) + \cdots \rho(v_i) \cdots + \rho(v_m)$. If $Vars(v_i)$ is an actual cause for r, then $Vars(v_j)$ is an actual cause for r and $Vars(v_i)$ is an actual cause for partition v_j. Thus, $MCS(v_i) = MCS(v_j) + MFS(v_j) - MFS(v_i)$.

This proves the theorem.

By Theorem 4, the Calc-Responsibility algorithm computes $MCS(v_i)$ of all nodes in a P-Tree top-down and further computes the responsibilities of all variables in read-once formulas.

Algorithm 2. Calc-Responsibility(*Root*)

```
1: if (Root is a leaf node) then
2:    resp(x) = 1/(1+MCS(x));
3: else if (type(Root) is ⊛) then
4:    for (every child node v_i of Root) do
5:       MCS(v_i) = MCS(Root);
6:       Calc-Responsibility(v_i);
7:    end for
8: else
9:    for (every child node v_i of Root) do
10:      MCS(v_i) = MCS(Root) + MFS(Root)−MFS(v_i);
11:      Calc-Responsibility(v_i);
12:   end for
13: end if;
```

The following theorem proves Algorithm 2 can compute correct $MCS(v_i)$ of all nodes in a P-Tree.

Theorem 5. *If a formula Φ can be compiled into a read-once form, Algorithm 2 can compute correct $MCS(v_i)$ of all nodes in a P-Tree and correct responsibilities of all variables in Φ.*

Proof. If a formula Φ can be compiled into a read-once form, then Algorithm 1 can build its P-Tree. Moreover, Theorem 4 ensures that Algorithm 2 can compute correct $MCS(v_i)$ of all nodes in a P-tree and correct responsibilities of all variables in Φ. This proves the theorem.

We prove the time complexity of Algorithm 2 as follows.

Theorem 6. *Assume that a formula Φ has N variables. If Φ can be compiled into a read-once form, then Algorithm 2 can compute the responsibilities of all variables in the time complexity of $O(N)$.*

Proof. By the property of a P-Tree, we conclude that the number of nodes in the P-Tree of Φ is not more than the sum of N and the number of conjunction and disjunction operators. Since both conjunction and disjunction operators are binary, the sum of conjunction and disjunction operators is less than N. Therefore, the number of nodes is less than $2N$.

Algorithm 2 computes the responsibilities of all variables at one time depth-first traversal of the P-Tree. So its time complexity is $O(N)$. This proves the theorem.

The following example demonstrates the technique to compute the responsibilities of all input tuples for the answer tuple r.

Example 3. Continuing Example 2. We have built a P-Tree of Φ and computed $MFS(v_i)$ of all nodes in the P-Tree. By Theorem 4 item 1, we set $MCS(\oplus_0) = 0$. We use the left branch of the P-Tree as an example to compute the responsibilities of all tuples in a depth-first method. Since $\oplus_0$ is a disjunction node, $MCS(\circledast_{11}) = MCS(\oplus_0) + MFS(\oplus_0) - MFS(\circledast_{11}) = 1$. Then, we look on $\circledast_{11}$ as a root node. Since $\circledast_{11}$ is a conjunction node, $MCS(a_2) = MCS(b_2) = MCS(\oplus_{21}) = MCS(\circledast_{11}) = 1$. So $resp(a_2) = resp(b_2) = 0.5$. Finally, we look on $\oplus_{21}$ as a root node. With a similar method, we can compute $MCS(v_i)$ of all its children and the responsibilities of all its leaf nodes. The final P-Tree is shown in Figure 2(a).

4 Computing Top-k Responsibility Tuples in a Read-Once Formula

The *MCS*s of the nodes in a P-Tree have the following properties:

Theorem 7. *The MCSs of the nodes in a P-Tree monotonically increase as we go down the P-Tree.*

Proof. The proof is very easy to see using recursion. In P-Trees, every leaf node is a distinct variable, and $\oplus$ and $\circledast$ alternate along each path. Assume that there are a node v_i and its parent node v_j in a P-Tree. There are the following two cases:

- If the first field of v_j is $\circledast$, then $MCS(v_i) = MCS(v_j)$ by Theorem 4 item 2. Thus $MCS(v_i) \geq MCS(v_j)$.
- If the first field of v_j is $\oplus$, then $MCS(v_i) = MCS(v_j) + MFS(v_j) - MFS(v_i)$ by Theorem 4 item 3. Since the first field of v_j is $\oplus$, $MCS(v_j) = \sum_{\Gamma} |\Gamma|$ by Theorem 1 item 3, where Γ ranges over its all children. Thus $MFS(v_j) > MFS(v_i)$ and $MCS(v_i) \geq MCS(v_j)$.

This proves the theorem.

We know that the *MFS* of every leaf node is 1. Thus if a $\circledast$ has a leaf node then its *MFS* is equal to 1 by Theorem 1 item 2. $\oplus$ does not influence the *MFS* of its parent node $\circledast$. Once we meet this kind of $\circledast$, we stop expanding its children and compute the *MFSs* bottom-up. In this way, we can compute the *MFS*s of all nodes except the unexpanded $\oplus$s. Since $\oplus$ may become a unexpended node, the P-Tree has a new kind of leaf node.

We know that the *MCS* of the root is equal to 0. Thus by Theorem 4, we can compute the *MCS* of every node in the P-Tree top-down. Since we should compute *MFS* bottom-up, we cannot compute the *MFS* of the unexpanded $\oplus$. Since the parent node of $\oplus$ is $\circledast$, the *MCS* of $\oplus$ is equal to the *MCS* of $\circledast$ by Theorem 4 item 2. Thus for the unexpanded $\oplus$, we can compute its correct

Algorithm 3. Building-PTree-Topk(G_c, *Root*, *disjunct* = *true*, *expand* = *true*)

```
1: create a node v_i;
2: if (G_c is a trivial graph) then
3:    type(v_i) = t and MFS(v_i) = 1;
4: else if (!expand) then
5:    type(v_i) = ⊕ and MFS(v_i) = x;
6:    add v_i to a queue;
7: else
8:    type(v_i) = !disjunct? ⊛: ⊕;
9:    isStop = (G_c includes a trivial graph && !disjunct)? true: false;
10:   for (every connected component G[j] of G_c) do
11:      convert G[j] to its complement graph Ḡ[j];
12:      if (isStop) then M[j] = Building-PTree-Topk(Ḡ[j], Root, !disjunct, false);
13:      else M[j] = Building-PTree-Topk(Ḡ[j], Root, !disjunct, true);
14:      MFS(v_j) = !disjunct ? min(M[0], ..., M[n]) : SUM(M[0], ..., M[n]);
15:   end for
16: end if;
17: add v_i to the child of Root;
```

MCS too. We sort the unexpanded nodes by their *MCS* in a queue. We expand the P-Tree of a formula step by step until we find the top-k responsibility tuples. We give the definition of the expanding step as follows.

Definition 4. *An expanding step of a P-Tree is to expand all branches until each of them meets a* $\circledast$ *whose children include at least one leaf node and children* $\oplus$*s will not be expanded in this step.*

If we do not find the top-k responsibility tuples in the first expanding step, we should further expand the P-Tree. We expand the unexpanded $\oplus$ with the smallest *MCS* from the queue. For the unexpanded $\oplus$, we have known its *MCS* but do not know its *MFS*. So the unexpanded $\oplus$ has the same property as the root node. Thus we can expand the unexpanded $\oplus$ as a root node for the following expanding step. And we can compute the correct *MFS* for each node except the new unexpanded $\oplus$s and the correct *MCS* for all nodes.

Algorithm 3 expands an expanding step of the P-Tree for every branch and then we can compute the *MCS* of every node by the Calc-Responsibility algorithm. Combining Theorem 7, we can check if we have found out the top-k responsibility tuples. If so, we can stop expanding the P-Tree. Otherwise, we should further expand the P-Tree. Thus we can find the top-k responsibility tuples without building the whole P-Tree of the formula.

The following example demonstrates the technique to find out the top-k responsibility tuples without building the whole P-Tree.

Example 4. We plan to compute the top-5 responsibility tuples of Equation (1). We build its P-Tree as follows. With the same method as in Example 2, we create a conjunction node $\circledast_{11}$ to represent Φ_1 and add it to the child of $\oplus_0$. Then we break $\overline{G}[11]$ into three connected components. Since its components a_2 and

Table 1. Lineage characteristics (scale factor 0.1)

Queries	Lineage size	#variables
Q3	125,154	158,504
Q10	11,433	21,833
Q15	180,785	181,785
Q16	80,000	100,001
Q18	165,324	216,368

b_2 are trivial graphs, we create one leaf node for every variable and then add them to the children of $\circledast_{11}$. Since $\overline{G}[11]$ includes trivial graphs and node $\circledast_{11}$ is conjunctive, node $\oplus_{21}$ will not be expanded. Thus node $\oplus_{21}$ is a unexpanded node. Then we create a conjunction node $\circledast_{12}$ to represent Φ_2 and add it to the child of $\oplus_0$. With the similar method to build the sub-tree of $\circledast_{11}$, we can build the sub-tree of $\circledast_{12}$. Its P-Tree is shown in Figure 2(b).

We use the left branch of the P-tree as an example to compute the *MFS*s of all nodes bottom-up. Since nodes a_2 and b_2 are both leaf nodes, their respective *MFSs* are equal to 1. We do not know the exact $MFS(\oplus_{21})$ but we do know that it is greater than 1. Since node $\circledast_{11}$ is a conjunctive node, $MFS(\circledast_{11})$ = $\min(MFS(a_2), MFS(b_2), MFS(\oplus_{21})$. Thus $MFS(\circledast_{11}) = 1$. We can further compute the *MCS*s of all nodes of the P-Tree similar as in Example 3. Since we have found the top-5 responsibility tuples, we will not expand node $\oplus_{21}$ anymore.

5 Experiments

The experiments were conducted on two intel Xeon CPUs each 2.0GHz, 4.0G Memory and running Windows Server Enterprise. We use boost library [1] to implement graph data structure and min-cut algorithm. The experiments have been done on TPC-H data with scale factor 0.1. Table 1 reports the sizes of the lineage (ie, number of clauses) for each of our queries and the number of variables needed for responsibility analysis. We first report the experiments for read-once formulas and then for top-k responsibility tuples of read-once formulas. In the experiments, we only consider Boolean queries.

5.1 The Experiments for Read-Once Formulas

Since the Building-PTree algorithm and the Cal-Responsibility algorithm are the two steps to calculate the responsibility, we call the combination of the two steps the PTree algorithm and denote it by *PResp*. We adopt the linear algorithm proposed in [22] as a baseline and denote it by *LResp*. From the query-centric perspective, Meliou et al. introduced *LResp* to perform responsibility analysis for linear queries. The lineages of hierarchical queries is read-once [24]. If a query is not hierarchical but linear, *PResp* can not perform its responsibility analysis but *LResp* can perform its responsibility analysis.

The following experiments are carried on the 22 TPC-H queries [24]. Since most of them involve complex aggregates, we modify the query and pick the queries with more than one joining relation. Since $Q3$, $Q10$, $Q15$, $Q16$ and $Q18$ are hierarchical [24], their results are all read-once. We change the inequality conditions of those queries. We further find that $Q3$, $Q10$, $Q15$, $Q16$ and $Q18$ are all linear. We compare the performance of *PResp*s of the above queries with those of their respective *LResp*s. The experimental results are shown in Figure 3, where we find that *PResp* performs one order of magnitude better than *LResp*, especially for $Q3$ and $Q16$. This is the following reason. Meliou et al. [22] employed max-flow/min-cut algorithm to compute the responsibility. The time complexity of their algorithm is $O(VM + V^2 \log V)$ [26], where V is the number of nodes and M is the number of edges. However, the time complexity of *PResp* is $O(\alpha N^2)$, where α is the number of tables in the query and N is the number of variables of the P-Tree. The relationship of the time complexity between *PResp* and *LResp* is $\gamma = \frac{VM+V^2 logV}{\alpha N^2}$. In [22], the hypergraph uses two nodes and one edge to denote a variable. From Table 1, we find the number of variables are large. Thus $logV$ is not small. In addition, V is bigger than N. Thus *PResp* has better performance than *LResp*.

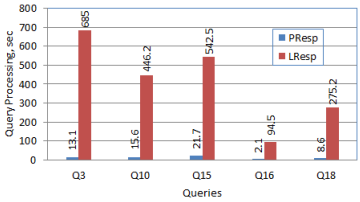


Fig. 3. Results on the TPC-H queries

5.2 The Experiments of Top-k Responsibility Tuples

Since the Building-PTree-Topk algorithm and the Cal-Responsibility algorithm are the two steps to find the top-k responsibility tuples, we call the combination of the two steps the TopK responsibility algorithm and denote it by *Topk*. In order to compute the top-k responsibility tuples for a query result, the naive method is to compute the responsibility of every tuple and then sort them by responsibility. We use *PResp* as a baseline to compute the responsibility of every tuple and then sort them by responsibility.

Assume that we want to compute the top-1000 responsibility tuples. We compare the performance of *Topk* of queries $Q3$, $Q10$, $Q15$, $Q16$ and $Q18$ with those of their respective *PResp*. The experimental results are shown in Figures 4, where we find that *Topk* has better performance than *PResp*. Since *Topk* can find the top-1000 responsibility tuples without building the whole tree, it has better performance than the naive algorithm. However, *Topk* does not perform more than one order of magnitude better than *PResp*. There are two reasons. First, since the data population is larger in the first several expanding steps

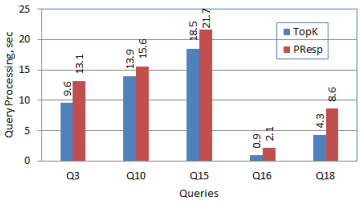


Fig. 4. Top-1000 results for TPCH scale factor 0.1

while it is smaller in the latter ones, the first several expanding steps cost much more than the latter ones. Second, *Topk* may expand many times before finding the top-1000 elements.

6 Related Work

The classical counterfactual causality (if X had not occurred, Y would not have occurred) was proposed by Hume [23], and the best known modern counterfactual analysis of causation was due to Lewis [19]. In recent work, Meliou et al. [20] initiated a discussion on causality in databases. By using the definition of actual causes by HP [13], Meliou et al. [21] proposed functional causes as a refined definition of causality for explaining query answers. Moreover, Meliou et al. [22] developed the notion of causality of input tuples on the result tuples of a query. Informally, a tuple t is a cause for a query result if there exists a state of database in which the presence/absence of t changes the query result for that new state of database. The responsibility [5] of a tuple t on the query result relates to the number of contingency sets [6]. Kanagal et al. [17] proposed a unified framework to handle sensitivity analysis and explanations for robust query evaluation in probabilistic databases. Their concept of influence has some connection with responsibility.

Provenance has been widely studied in the database literature [4,11,7,2,29] and several models of provenance have been proposed, based on Boolean formulas (lineage), semirings and so on. Cui et al. [7] studied lineage, in which an output tuple is associated with a set of input tuples that influenced the output tuple. Their work seems to be the first model to be defined for relational queries based on a semantic criterion. Subsequently, Buneman et al. [2] defined two forms of provenance called why- and where-provenance for queries in a deterministic tree data model. Meliou et al. [22] pointed to the close connection between why-provenance and why-so causality. The how-provenance and semiring-valued relational models were introduced by Green et al. [11], along with a high-level discussion of the possibility of extracting lineage and why-provenance from how-provenance. The Trio project [29] investigated combining tuple-level lineage with uncertainty and probability information.

The goal of *Knowledge Compilation* [16,9] is to represent a Boolean expression in a format in which it can answer a range of online-queries in PTIME. Read-once Boolean formula is one of the tractable expressions, which have been studied for

some time, albeit under various names [14,28,18]. It turns out that Golumbic et al. [10] gave a fast algorithm that takes as input a formula in irredundant disjunctive normal form, decides whether it is read-once, and if it is, computes the read-once form. Moreover, Sen et al. [25] proved that one only needs to test if the co-occurrence graph is a cograph to judge whether the Boolean formulas of the result tuples produced by conjunctive queries without self-joins are read-once. In recent work, Fink et al. [9] compiled semimodule and semiring expressions into so-called decomposition trees. Then the computation of the probability distribution of positive relational algebra queries with aggregates can be done in time linear in the product of the size of the probability distributions represented by its nodes. In this paper, we propose a P-Tree to compute the responsibilities of tuples in read-once formulas.

7 Conclusions and Future Work

Responsibility refines the concept of causality by measuring the degree of causality as a function of the size of the smallest contingency set and serves to rank potentially many causes by their relative contributions to the effect. In this paper, we extend the Co-Tree to P-Tree for read-once expressions. Based on P-Trees, the proposed algorithm can efficiently compute responsibilities of variables in read-once formulas. Moreover, we introduce a novel algorithm to find top-k responsibility tuples. We plan to use parallel algorithm to build P-Trees and compute the responsibility of every tuple in the future.

Acknowledgements. The authors would like to thank Alexandra Meliou for the source code of their algorithm for computing responsibility of linear queries. This work is partially funded by the National Natural Science Foundation of China under Grant No. 61170012, the National Basic Research Program of China (973 Program) under Grant No. 2012CB316205, and the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China under Grant No. 10XNJ048.

References

1. `http://www.boost.org/`
2. Buneman, P., Khanna, S., Tan, W.: Why and where: A charaterization of data provenance. In: ICDT 2001, pp. 316–330 (2001)
3. Chapman, A., Jagadish, H.V.: Why not? In: SIGMOD 2009, pp. 523–534 (2009)
4. Cheney, J., Chiticariu, L., Tan, W.C.: Provenance in databases: Why, how, and where. Foundations and Trends in Databases 4(1), 379–474 (2009)
5. Chockler, H., Halpern, J.: Responsibility and Blame: A Structural-Model Approach. Journal of Artificial Intelligence Research 22(1), 93–115 (2004)
6. Chockler, H., Halpern, J., Kupferman, O.: What causes a system to satisfy a specification. ACM Transactions on Computational Logic 9(3), 1–26 (2008)
7. Cui, Y., Widom, J., Wiener, J.L.: Tracing the lineage of view data in a warehousing enviornment. ACM TODS 25(2), 269–332 (2000)

8. Eiter, T., Lukasiewicz, T.: Complexity results for structure-based causality. Artificial Intelligence 142(1), 53–89 (2002)
9. Fink, R., Han, L., Olteanu, D.: Aggregation in probabilistic databases via knowledge compilation. PVLDB 5(5), 490–501 (2012)
10. Golumbic, M., Mintz, A., Rotics, U.: Factoring and recognition of read-once functions using cographs and normality and readability of functions associated with partial k-trees. Discrete Applied Mathematics 144(10), 1465–1477 (2006)
11. Green, T., Karvounarakis, G., Tannen, V.: Provenance semirings. In: PODS 2007, pp. 31–40 (2007)
12. Gurvich, V.A.: Criteria for repetition-freeness of functions in the algebra of logic. Soviet Math. Dokl. 43(3), 721–726 (1991)
13. Halpern, J., Pearl, J.: Causes and Explanations: A Structural-Model Approach–Part I: Causes. British Journal for Philosophy of Science 56(4), 843–887 (2005)
14. Hayes, J.: The fanout structure of switching functions. Journal of the ACM 22(4), 551–571 (1975)
15. Huang, J., Chen, T., Doan, A., Naughton, J.F.: On the provenance of non-answers to queries over extracted data. PVLDB 1(1), 736–747 (2008)
16. Jha, A., Suciu, D.: Knowledge compilation meets database theory: compiling queries to decision diagrams. In: ICDT 2011, pp. 162–173 (2011)
17. Kanagal, B., Li, J., Deshpande, A.: Sensitivity Analysis and Explanations for Robust Query Evaluation in Probabilistic Databases. In: SIGMOD 2010, pp. 675–686 (2010)
18. Karchmer, M., Linial, N., Newman, I., Saks, M., Wigderson, A.: Combinatorial characterization of read-once formulae. Discrete Mathematics 114(1-3), 275–282 (1993)
19. Lewis, D.: Causation. The Journal of Philosophy 70(17), 556–567 (1973)
20. Meliou, A., Gatterbauer, W., Halpern, J., Koch, C., Moore, K., Suciu, D.: Causality in databases. IEEE Data Engineering Bulletin 33(3), 59–67 (2010)
21. Meliou, A., Gatterbauer, W., Moore, K., Suciu, D.: WHY SO? or WHY NO? Functional Causality for Explaining Query Answers. In: MUD 2010, pp. 3–17 (2010)
22. Meliou, A., Gatterbauer, W., Suciu, D.: The complexity of causality and responsibility for query answer and non-answer. PVLDB 4(1), 34–45 (2011)
23. Menzies, P.: Counterfactual theories of Causation. Stanford Encylopedia of Philosophy (2008)
24. Olteanu, D., Huang, J., Koch, C.: SPROUT: Lazy vs. Eager Query Plans for Tuple-Independent Probabilistic databases. In: ICDE 2009, pp. 640–651 (2009)
25. Sen, P., Deshpande, A., Getoor, L.: Read-once functions and query evaluation in probabilistic databases. PVLDB 3(1), 1068–1079 (2010)
26. Stoer, M., Wagner, F.: A Simple Min-Cut Algorithm. Journal of the ACM 44(4), 585–591 (1997)
27. Tran, Q., Chan, C.: How to ConQueR why-not questions. In: SIGMOD 2010, pp. 15–26 (2010)
28. Valiant, L.: A theory of the learnable. Communications of the ACM 27(11), 1134–1142 (1984)
29. Widom, J.: Trio: A system for integrated management of data, accuracy, and lineage. In: ICDR 2005, pp. 262–276 (2005)

Minimizing Explanations for Missing Answers to Queries on Databases*

Chuanyu Zong, Xiaochun Yang, Bin Wang, and Jingjing Zhang

College of Information Science and Engineering,
Northeastern University, Liaoning, 110819, China
zongchuanyu@research.neu.edu.cn, {yangxc,binwang}@mail.neu.edu.cn

Abstract. In recent years, explaining missing answers in the query results achieved from the extracted data has become an attention topic. Several techniques have been developed to do the explanation, however, they posted lots of disadvantages, including incorrect explanations, unreasonable explanations, and non-minimized explanations. In this paper, we propose a novel explanation technique to avoid such unexpected explanations and we guarantee that the generalized explanation is minimum. We propose two algorithms by considering different kinds of modification object values for the attributes satisfying referential integrity constraint. Experimental results show that our approach can efficiently minimize explaining missing answers.

1 Introduction

Although information extraction [1] can help users to find requiring information data efficiently and quickly, great data uncertainty has been brought at the same time. Users may feel confused that the answer they expect is not in the query results when querying on extracted data, then they may ask why? It is becoming more and more important to support users with data provenance explanations for query results. Data provenance [2,3] which is also called data lineage describes the origins and the evolution history of data, and it provides users with provenance explanations for the query answers received from the data. Most of the data provenance technologies can only explain why the answers are in the query results, but they can not answer why the answers are not in the query results. In other words, they can't explain the problem of missing answers, such as query rewriting technology [4], provenance semiring [5], and so on.

1.1 Motivation

To explain missing answers, literature [6] has proposed an explaining model based on value-modification. However, literature [6] has some disadvantages as

* The work is partially supported by the National Basic Research Program of China (973 Program) (No. 2012CB316201), the National Natural Science Foundation of China (Nos. 60973018, 60973020, 61129002), the Doctoral Fund of Ministry of Education of China (No. 20110042110028), the Fundamental Research Funds for the Central Universities (Nos. N110804002, N110404015).

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 254–268, 2013.

	ANO	ANAME
a_1	A1	AC
a_2	A2	CS
a_3	A3	CE
a_4	A4	SZ

(a) Airline

	RNO	LEAVE	TO
r_1	R1	Shenyang	Hangzhou
r_2	R2	Shenyang	Nanjing
r_3	R3	Shanghai	Guangzhou
r_4	R4	Beijing	Chengdu

(b) Route

	ANO	RNO	ECONOMY	BUSINESS
p_1	A1	R1	1400	2100
p_2	A2	R2	990	1485
p_3	A3	R3	880	1320
p_4	A3	R1	1300	1950
p_5	A4	R4	820	1230

(c) Price

	NAME	SUM
s_1	Shenyang	2
s_2	Shanghai	1
s_3	Beijing	1

(d) Statistic

Fig. 1. Extracted Air Ticket Information Tables (`AIR CHINA`, `CHINA SOUTHERN`, `CHINA EASTERN` and `SHENZHEN AIRLINES` in short: `AC`, `CS`, `CE`, `SZ`)

follows: i) It returns some incorrect explanations which result in missing existed query answers. ii) Some unreasonable explanations which contain overabundance modifications are returned when the query statement contains query inequality. iii) It dose not consider that some modifications of data violate the function dependency or aggregation constraint. iv) It returns all non-minimized explanations for the missing answers which make users confused. Most of the users may just want to get the most possible explanations, in other words, they are just interested in those explanations whose number of modified components in tuples is the smallest while modifying the original data to get missing answers.

Example 1. Assume that an IE application extracts air tickets information for airlines, air routes and ticket prices, and we extract four tables which are shown in Fig. 1. Fig. 1(a) shows a table storing names of airlines. Fig. 1(b) shows a table storing the departure and destination of air routes. Fig. 1(c) shows a table storing two kinds of prices which are the economy and business price of an air route supplied by an airline. Fig. 1(d) shows a view counting the number of air routes whose departures are same. In our work, we suppose that the value of attribute `Price.BUSINESS` is 1.5 times as large as `Price.ECONOMY`, and the value of `Statistic.SUM` aggregately depends on attribute `Route.LEAVE`.

Now suppose that a user has a question about names of the airlines which leave from `Shenyang` and the economy price is within `1000`. The IE application can answer the question by executing the following query Q:

Select a.ANAME, From Airline a, Route r, Price p Where a.ANO=p.ANO And r.RNO=p.RNO And p.ECONOMY $\leq$ `1000` And r.LEAVE=`Shenyang`.

Given extracted data mentioned above, the query returns `CS`. The user may be confused that missing answer `CE` dose not appear in the query results, his friend just bought an airline ticket he wants from `CE` a few days ago. Using explaining model in [6], the user can get some explanations for `CE` as shown in Table 1. As shown in Explanation e_1, if modifying `A2` to `A3` and `990` to $n(n \leq$ `1000`$)$ in p_2, `CE` will appear in the query results. Obviously e_1 is incorrect, because the modification of `A2` makes the existed answer `CS` no longer appear in the query results. And at the same time, the modification of `990` violates the functional

Table 1. False Explanations for CE

	ANO	ANAME	RNO	LEAVE	TO	ANO	RNO	ECONOMY	BUSINESS
e_1	A3	CE	R2	Shenyang	Shanghai	A2 → A3	R2	990 → n	1485
e_2	A3	CE	R3	Shanghai → Shenyang	Guangzhou	A3	R3	880 → n	1320

Table 2. Correct Explanations for CE

	ANO	ANAME	RNO	LEAVE	TO	ANO	RNO	ECONOMY	BUSINESS
e_3	A3	CE	R1	Shenyang	Hangzhou	A3	R3 → R1	880	1320
e_4	A3	CE	R1	Shenyang	Hangzhou	A3	R1	1300 → n	1950 → 1.5n
e_5	A3	CE	R3	Shanghai → Shenyang	Guangzhou	A3	R3	800	1320

dependency, because it only modifies the value of attribute Price.ECONOMY without Price.BUSINESS. Furthermore, explanation e_1 is also unreasonable, because 990 in p_2 has already been within 1000 and it no longer needs to be modified to n. Explanation e_2 violates aggregation constraint, because it only modifies the value of attribute Route.LEAVE. Note that the value of Statistic.SUM should have been updated at the same time, but it dosen't.

1.2 Related Work and Contributions

There have already existed some explaining models. For example, while the missing answers need appearing in the query results, [6] processes the problem by presenting the users how to modify the original data, which is related to our work. However, our work conquers the disadvantages mentioned above in [6], and we propose a new framework to minimize explaining missing answers.

Literature [7] explains missing answers by identifying the "culprit" operations which can exclude the missing answers in the query statement.

Another model is called Artemis [8,9]. While the missing answers need appearing in the query results, it presents users what kinds of tuples should be inserted into the original database to explain the missing answers. Both of this model and our work are based on modifying the original database. But the modifications in their approach are much different from ours, which only modifies the value of attributes in original database.

The recent model is based on query-refinement [10]. It explains missing answers by automatically generating a refined query whose query results include both the original query answers and the missing answers. Another work [11] uses the same approach to address the why-not questions on top-k queries.

Our work is also related to the problem of database repairing [12]. The problem we process is orthogonal to the database repairing problem. We attempt to find modifications in a consistent database, while the database repairing problem only considers how to answer queries over an inconsistent database.

We can get some correct explanations for CE as shown in Table 2, although e_3, e_4 and e_5 all can explain CE, e_3 only needs to modify one component in tuples, e_4 needs to modify two components and e_5 needs to modify three components, which is because two cascade modifications are triggered when modifying

`Shanghai` to `Shenyang`. One challenge in our work is how to guarantee existed query answers to be lossless and avoid overabundance modifications. Moreover, the other challenge is to quickly get the most possible explanations.

Our work is motivated by [6]. We propose a new technique to minimize explaining missing answers and conquer those disadvantages in [6]. Our contributions can be summarized as follows. i) We present a new framework to minimize explanations based on value-modification. ii) We overcome those disadvantages in [6]. iii) We improve the quality of explanations for missing answers. iv) We propose two algorithms to minimize explaining missing answers. v) We evaluate our algorithms using experiments in extracted real world data.

The rest of the paper is organized as follows. Section 2 formally defines our problem and presents some related definitions. In Section 3, we present our explaining framework and algorithms to minimize explaining missing answers. Section 4 evaluates our approach and makes a comparison with [6]. Section 5 concludes the paper.

2 Preliminaries and Problem Definition

The setting in our work with data in the database extracted from an IE system is the same with [6]. An attribute is called a *trusted attribute* if all its values in a table are correct, and the value of it can not be modified in the process of generating explanations. The users can manually choose to trust individual attributes. However, in general it can be done automatically that referenced attributes should be *trusted attributes*. Because usually modifying values of referenced attributes would violate referential integrity constraint, then we need to modify much more components to avoid violating referential integrity constraint. This is not worth the candle for minimal explanations. Therefore, the referenced attributes can automatically be chosen as *trusted attributes*.

The model in [6] separates the original component in tuples from the modified component by using an arrow (for example, $u \rightarrow u'$ indicates that u must be changed to u' to make the missing answer appear in the query results). In our work, we will continue using this separation approach.

In this paper, we only consider the equi-join between two query tables R_1 and R_2 like $R_1.c_1 = R_2.c_2$, and the attributes c_1 and c_2 should satisfy referential integrity constraint. The referencing relation between the attributes in those tables may constitute a directed graph when the query statement involves multiple query tables. Suppose that a query statement involves five tables $v1$, $v2$, $v3$, $v4$, $v5$, and their referential relation is illustrated in Fig. 2. For simplicity, this paper only considers that the directed edge E_{v1-v3} in Fig. 2(a) represents that one attribute in $v3$ references to one attribute in $v1$. The directed edges E_{v2-v3}, E_{v3-v4} and E_{v4-v2} constitute a cycle as shown in Fig. 2(b).

Definition 1. *(Cascade modification). Given an explanation e, if some modifications in e violate a functional dependency f or an aggregation constraint α, other components involved in f or α must be modified at the same time.*

(a) Acyclic (b) Cyclic

Fig. 2. Two directed graphs for different table referencing relation

For example, consider explanation e_5 in Table 2 and tables in Fig. 1, we modify `Shanghai` to `Shenyang` in r_3 which would trigger two cascade modifications that modifying the value of `Statistic.SUM` to `0` in s_2 and to `3` in s_1.

Cascade modification can help us to improve the quality of explanations. Now we present the definitions of m-value and minimizing explanations.

Definition 2. *(m-value). The m-value of an explanation e is defined as the number of modified components in e plus the number of cascade modified components triggered by some modifications in e.*

Problem Definition. Given a query and a missing answer t, the minimizing explanations problem for t is to find explanations whose m-value is the smallest among all the explanations for t.

For example, consider the explanations e_3, e_4, and e_5 in Table 2, the m-value of e_3 is 1, the m-value of e_4 is 2 including one cascade modification and the m-value of e_5 is 3 including two cascade modifications. The minimizing explanations problem is to quickly find the minimal explanation e_3 for `CE`.

3 Minimizing-Explanation Algorithms

3.1 Explanation Framework

The query Q, database DB, query answers R and missing answer ΔR are taken as input to our explaining framework, and $DB + \Delta DB$ is returned after minimal modifying DB which is shown in Fig. 3. We need to find a minimal modification plan for DB to make sure that the query results will include R and ΔR when executing Q on $DB + \Delta DB$. At the same time, this minimal modification plan must guarantee to avoid OM (overabundance modification) and violating FD (functional dependencies) and AC (aggregation constraints).

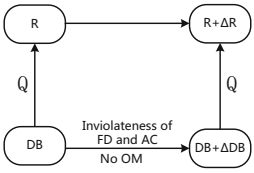


Fig. 3. Explaining framework

To get ΔR, we can determine a modification object value for every attribute column in each table in DB. According to the query predicates and the missing answer, the object values of some attribute columns can be projected to a constant. And the object values of other attribute columns that can not be projected will be symbolized as a wildcard '?' (which can stand for anything satisfying the query or constraint predicate). They may also be symbolized as a variable satisfying a boolean expression when the query statement contains inequality. These object values of each attribute column in the table can constitute a modification object value set for the table. Consider Example 1, according to Q and `CE` we can project `Airline.ANAME` to `CE`, `Route.LEAVE` to `Shenyang`, and `Price.ECONOMY` to a variable n ($n \leq$ `1000`). The object values of other attribute columns are symbolized as a wildcard '?'. So we can determine the object value set of Airline as ('?',`CS`). To minimize modifying DB to $DB + \Delta DB$, we need to choose tuples in Airline which are the most similar to the modification object value set of Airline. So we choose a_2 in Airline to generate the minimal explanations.

To get the minimal modification plan, according to Q, DB and ΔR we define a modification object for each query table. We can find tuples which are the most similar to their modification objects in each table, then modify them and generate $DB + \Delta DB$ according to their modification object. In the following, we give the definition of modification object to find the minimal modification plan.

3.2 Modification Object

Definition 3. *(Modification object). A modification object (m-object for short) is an ordered object value set* (m_1, m_2, m_3, m_4..., m_n) *for a query table, m_i represents the modification object value of the i-th attribute column in the table, and it may be a constant, a wildcard '?' or a variable satisfying a boolean expression.*

If m_i can not be projected to a constant, we will use a wildcard '?' to symbolize its value. If the attribute corresponding to the value m_i satisfies referential integrity constraint (we name this attribute as reference attribute and the attribute it referencing as referenced attribute), we will use a variable to replace the wildcard '?', because the object value of a reference attribute is determined by that of its referenced attribute. If m_i needs to satisfy a query inequality, it will be symbolized as a variable such as $n \leq$ `30`.

Consider Example 1, we can get the modification objects of query tables shown in Table 3. Because Q and `CE` can not determine the constant object values of `Airline.ANO, Route.RNO, Price.ANO`, and `Price.RNO`, we use the variable A_x, R_x, A_x, and R_x to symbolize them respectively and use the wildcard to symbolize `Route.TO` and `Price.BUSINESS`. The query predicate in Q can determine that the value of `Price.ECONOMY` should be within `1000`, so we symbolize its object value with a variable n ($n \leq$ `1000`).

Table 3. Modification object of tables

	Table	Modification Object
t_1	Airline	(A_x,CE)
t_2	Route	(R_x,Shenyang,'?')
t_3	Price	(A_x,R_x,$n(n \leq 1000)$,'?')

Table 4. Modification priority of tuples

	Table	Modification Priority
t_1	Airline	$a_3 > (a_1, a_4)$
t_2	Route	$(r_1, r_2) > (r_3, r_4)$
t_3	Price	$(p_3) > (p_4, p_5) > p_1$

3.3 The Function of Modification Object

Definition 4. *(Tuple dissimilarity). Given a relation R, let t be a tuple of R, $t \in R$, and the m-object R_m, the tuple dissimilarity (t-diss for short) of t is the number of different components between t and R_m plus the number of components need cascaded modifying triggered by modifying the components in t.*

If m_i is '?', it can be equivalent to any components in tuples. If m_i is a variable satisfying a boolean expression, and there is one component just also satisfying the boolean expression, we consider that the component is equivalent to m_i. If the object value of a reference attribute in a table is a variable, we should firstly determine the object value of its referenced attribute, then determine its object value, and compute the t-diss of the tuples in the table.

For example, given query tables as shown in Fig. 1 and the corresponding m-objects as shown in Table 3. The t-diss of a_3 is 0 and the t-diss of r_3 is 3. The object values A_x and R_x are all variables, we should determine their values firstly, then determine the m-object of Price, and compute the t-diss of the tuples in Price. Suppose that A_x is A3 and R_x is R1, the t-diss of p_4 is 2. However if we determine A_x as A3 and R_x as R2, the t-diss of p_4 is 3.

To avoid missing the existed query answers and improve the quality of explanations, we have to determine those tuples in the query tables which can be used to generate the explanations for the missing answers.

Theorem 1. *Given a tuple t_1 contributes to the query answer, if the t-diss of t_1 is not equal to 0, then t_1 can not be used to generate explanations.*

Proof. If the t-diss of t_1 is not equal to 0, we will use t_1 to generate explanations. The components in t_1 which are different from their object values must be modified. However, modifying t_1 will result in missing existed answer, so t_1 could not be used to generate explanations.

While one query answer is generated by executing the same query on joining different tuples, which is the special case. If the tuples involing in the joins are quite different from each other, we will use those tuples to generate explanations. If the joins contain some identical tuples, we will discard all the identical tuples and use those different tuples to generate explanations.

Theorem 2. *Given a trusted attribute p in R and the m-object R_m of R, if the object value of p in R_m is a constant representing as v, then we can only choose the tuple whose value of p is equivalent to v to generate explanations.*

Proof. Attribute p is a *trusted attribute* means its value can't be modified. If the object value of p is a constant representing as v, we can not modify the value of p to v which is not equivalent to v to generate explanations and we can only use those tuples whose value of p is equivalent to v to generate explanations.

When the query statement contains inequality, literature [6] does not check the component in the tuple using to generate explanations satisfy the inequality or not, but directly modify the component to a variable. In that case, the explanations returned by the approach in [6] contain some overabundance modifications. In our work, we modify components in tuples with their object values to generate explanations. And we need to check if the component is equivalent to its object value before modifying the component. Once the component is equivalent to its object value, we do not modify this component. Therefore, overabundance modifications should be avoided.

3.4 Modification Priority

To generate the minimal explanations, we propose the definition of modification priority as the order to modify the tuples in a query table.

Definition 5. *(mpTuples). The mpTuples are defined as the tuples with the highest modification priority in a table.*

Theorem 3. *Given a tuple t_1 in the table R and the m-object R_m of R, the smaller the t-diss of t_1 is, the higher the modification priority of t_1 has.*

Proof. The t-diss of t_1 in R is the smallest means that the number of modified components in t_1 will be the smallest if we modify t_1 to generate explanations. Therefore, we prior to modify t_1 to generate the minimal explanations, the modification priority of t_1 is the highest among all the tuples in R.

Example 2. Consider Example 1, the modification priority of tuples in Airline discarding a_2, Route discarding r_2, and Price discarding p_2 are shown in Table 4. t_3 shows the modification priority of tuples in Price discarding p_2 when determining A_x as `A3` and R_x as `R2`.

3.5 Exploring Minimal Explanation

In this section, we propose two algorithms to minimize explanations.

Suppose that the nodes $v2$, $v3$, and $v4$ in Fig. 2(b) represent three query tables R_1, R_2, and R_3. If the object values of the reference attributes in these tables all contain variables, then the values of those variables can not be determined. Because the object value of a reference attribute in R_2 needs to be determined by that of its referenced attributes in R_1. However, determining the object values of R_1 needs to determine that of R_3 firstly and determining the object values of R_3 needs to determine that of R_2 firstly. These variables form a endless loop. So we can not determine the m-object of these tables and the modification priority

of the tuples in these tables, and our approach can not be used to generate the minimal explanations.

Because the operation between two query tables is accomplished by joining one attribute and its reference attribute, whether the object value of a non-reference attribute is a variable or not has no effect on the join, but the reference attribute dose. Thereby, whether the object values of reference attributes in query tables contain variables or not, we present two algorithms.

3.5.1 An Algorithm for Constant Reference Attribute Values

Theorem 4. *Given a query Q, database D, and a missing answer t. If none of the object values of the reference attributes in m-objects of all query tables is a variable, then the explanation set E generated by joining the modified mpTuples in each table is the minimal explanation set for t.*

Proof. It can be proved by contradiction. If none of the object values of the reference attributes is a variable, the process of choosing mpTuples in each query table dose not affect each other. If existing one explanation e the m-value of which is less than that of explanations in E, the modification priority of one tuple t_1 being modified to generate e must be higher than that of the tuple modified to generate E which comes from the same table with t_1. There exist two kinds of mpTuples with different t-diss in one query table, which violates the definition of mpTuples and is unreasonable, so E is a minimal explanation set for t.

Algorithm 1: MinExplain - ConsRefAttr

Input: The query Q, database D, missing answer t and *trusted attributes*;
Output: A minimal explanation set E ;

1 Initialize a set E' to store the mpTuples of each query table;
2 **for** *each table R mentioned in Q* **do**
3 $R_m \leftarrow$ get the m-object of R;
4 $R' \leftarrow$ process contributing tuples for query answers;
5 $R' \leftarrow$ process *trusted attributes* in R';
6 $R'_{set} \leftarrow$ get the mpTuples in R' according to R_m;
7 $E' \leftarrow$ modify R'_{set} according to R_m;
8 $E' \leftarrow$ process cascade modifications;
9 $E \leftarrow$ make a cartesian product between E and E';
10 **return** E;

When none of the object values of the reference attributes in query tables is a variable, we can minimize explaining missing answers by using Algorithm 1. Algorithm 1 makes a cartesian product between the modified mpTuples of each table R mentioned in Q and E iteratively (lines 2-9). We firstly get the m-object of R (line 2). Then we process contributing tuples for query answers (line 4). If R contains *trusted attributes*, we process them according to their object values in R_m (line 5). We get mpTuples in R (line 6). Then we modify those tuples according to their m-object, if some modifications violate functional dependency or aggregation constraint, some cascade modifications will be triggered (lines 7-8). Finally, we make a cartesian product between modified mpTuples in R and E to generate the minimal explanation set E (line 9).

3.5.2 An Algorithm for Variable Reference Attribute Values

It is mentioned above in Algorithm 1, we choose mpTuples in each query table to generate the minimal explanations. However, it can affect the generation of mpTuples in table R where the object values of the reference attributes contain one or more variables. We define join plan as a set of tuples which can accomplish join operations. We have two approaches to determine the value of those variables. One is to resolve them by the value of their referenced attributes corresponding referenced tables of R as mentioned in Example 2. The other is to determine them by choosing one tuple in R. Consider Example 1, we can choose p_3 in Price to determine A_x as `A3` and R_x as `R3`. So we can choose a_3 in Airline and r_3 in Route to constitute a join plan with p_3. We define the sum of the t-diss of the tuples in a join plan as the t-diss of the join plan, so the t-diss of the join plan constituted by p_3, a_3 and r_3 is 3.

To determine the mpTuples of R we define those join plans as the minimum join plans for R with the smallest t-diss. Those join plans join the mpTuples of R and other tables which are in front of R in the topological order about the referencing relation of all query tables.

We have two approaches to get the minimum join plans for R as mentioned above. Suppose that we get the mpTuples of R as mp_1 and generate some minimum join plans jp_1 by using the first approach, and get another mpTuples of R as mp_2 and generate some minimum join plans jp_2 by using the second approach. Then we compare the t-diss of jp_1 and jp_2, if the t-diss of jp_1 is smaller than that of jp_2, we determine mp_1 as the mpTuples of R, if the t-diss of jp_1 is equal to that of jp_2, we determine mp_1 and mp_2 as the mpTuples of R, otherwise, we determine mp_2 as the mpTuples of R. Consider Example 1, jp_1 contains of two join plans, one of which consists of a_3, r_1, and p_3 and the other consists a_3, r_2, and p_3. Their t-diss is 1. jp_2 consists of p_4, a_3, and r_1, and its t-diss is 2. Thereby, we determine p_3 as the mpTuples in Price.

Theorem 5. *If the object values of the reference attributes in R contain one or more variables, the tuples in R with the smallest t-diss determined by mpTuples in the referenced tables of R are the mpTuples in R by using the first approach.*

Proof. It can be proved by contradiction. Given three query tables R_1, R_2, and R_3, and their referencing relation just like the relation of three nodes A, B, and C in Fig. 2(a). Assume one join plan pl_1 is that we choose the mpTuples of R_1 whose t-diss is 0 and the mpTuples of R_2 whose t-diss is 1, and we can determine the mpTuples of R_3 whose t-diss is 2, so the t-diss of pl_1 is 3. If existing alternative join plan pl_2 that we choose the mpTuples of R_1 whose t-diss is 1 and the mpTuples of R_2 whose t-diss is 1, and we can determine the mpTuples of R_3 whose t-diss is 0, so the t-diss of pl_1 is 2. Based on the consideration above, we can determine a tuple of R_3 whose t-diss is 1 in pl_1, which pl_1 and pl_2 have only one different attribute values in R_3 referencing attribute in R_1. But this contradicts our assumption that we choose the tuples of R_3 whose t-diss is the smallest. Therefore the muTuples of R in pl_1 will be the mpTuples of R.

In the procedure **BackTrack**, we need to backtracking traversal every tuple in R', and we propose an approach to prune this procedure. We backtrack to choose one tuple t_1 in R to get the mpTuples of R, if the sum of accumulative t-diss has already been larger than the t-diss of the join plans generated by the first approach, we can break and choose the next tuple in R to backtrack. For instance, we select P_1 in Price whose t-diss is 2, which has already been larger than the sum of t-diss of a_3, r_1, and p_3 equaling to 1 generated by the first approach, then we break and select p_2.

Algorithm 2: MinExplain - VarRefAttr

Input: The query Q, database D, missing answer t, referenced tables R_r of a table R and *trusted attributes*;
Output: A minimal explanation set E;
1 Initialize a set E_{set} to store the mpTuples of each query table, T_{set} to store tuples of tables having been polled out;
2 $S \leftarrow$ generate the topological order of the referencing relation graph of D;
3 **while** S *is not empty* **do**
4 $\quad R_m \leftarrow$ poll a table R and generate its m-object ;
5 $\quad R' \leftarrow$ process contributing tuples and *trusted attributes* ;
6 $\quad$ **if** *reference attributes in* R_m *associate with one or more variables* **then**
7 $\quad\quad R_{m_{set}} \leftarrow$ get all the possible R_m according to the mpTuples of R_r;
8 $\quad\quad R' \leftarrow$ process the contributing tuples and *trusted attributes* again;
9 $\quad\quad R_{mp} \leftarrow$ get the modified mpTuples of R' according to $R_{m_{set}}$;
10 $\quad\quad E_1 \leftarrow$ join E_{set} and R_{mp};
11 $\quad\quad E_2 \leftarrow$ **BackTrack**(R', T_{set});
12 $\quad\quad E' \leftarrow$ compare the t-diss of E_1 with E_2 to resolve the mpTuples for R';
13 $\quad$ **else**
14 $\quad\quad E' \leftarrow$ determine the mpTuples of R;
15 $\quad$ add E' to E_{set}, add R' to T_{set};
16 make joins between mpTuples in E_{set} to generate E and modify E;
17 **return** E;

Algorithm 2 firstly gets the topological order of the referencing relation graph of D by continuously deleting the node with indegree 0 and puts those table names into a queue S orderly (line 2). It sequentially removes one table R from S and takes the m-object as R_m of R (line 4). R' is generated by processing the contributing tuples to existed answers and *trusted attributes* (line 5). If object values of reference attributes in R_m contain one or more variables, we will determine all the possible R_m with the mpTuples in R_r and process the contributing tuples and *trusted attributes* again (lines 6-8). We get the mpTuples of R' and modify the reference attributes value in mpTuples of R', then join E and mpTuples of R' to generate E_1 (lines 9-10). In the procedure **BackTrack**, we backtrack from R' to T_{set} to get the mpTuples for R' and represent the minimal join plans as E_2 (line 11). We make a comparison of the t-diss of E_1 with E_2 to determine the mpTuples of R', If the t-diss of E_2 is smaller than or equal to the t-diss of E_1, we need to update E (line 12). Otherwise we directly choose the mpTuples in R' to generate E' (line 14). We add E' to E_{set}, and add R' to T_{set} (line 15). Finally, we do a join operation between mpTuples of each table in E_{set} and modify them to get the minimal explanation set E(lines 16).

Table 5. Different queries over database D_B

Qid	Query	Missing Answer
Q_0	$\Pi_{ANAME}(Airline \bowtie \sigma_{ECONOMY\leq(1000)}(Price) \bowtie \sigma_{LEAVE=Beijing}(Route))$	CS
Q_1	$\Pi_{ANAME}(Airline \bowtie \sigma_{ECONOMY\leq(1000)}(Price) \bowtie \sigma_{LEAVE=Shenyang}(Route))$	SC
Q_2	$\Pi_{ANAME}(Airline \bowtie \sigma_{ECONOMY\leq(1000)}(Price) \bowtie \sigma_{LEAVE=Tianjin}(Route))$	CE
Q_3	$\Pi_{ANAME}(Airline \bowtie \sigma_{ECONOMY\leq(1000)}(Price) \bowtie \sigma_{LEAVE=Nanjing}(Route))$	SD
Q_4	$\Pi_{TO}(\sigma_{ANAME=CS}(AirLine) \bowtie \sigma_{ECONOMY\leq(1000)}(Price) \bowtie Route)$	Hangzhou
Q_5	$\Pi_{TO}(\sigma_{ANAME=CE}(AirLine) \bowtie \sigma_{ECONOMY\leq(1000)}(Price) \bowtie Route)$	Kunming

Table 6. Comparison results of explanations for $Q_0 - Q_5$

Queries	Approaches	All	Correct	Missing answer	Overabundance	Violate function	Violate aggregation
Q_0	MinExplain	3	3	0	0	0	0
	NonAnswers	8460	0	2328	4371	8460	7191
Q_1	MinExplain	12	12	0	0	0	0
	NonAnswers	8460	0	3588	4371	8460	6532
Q_2	MinExplain	8	8	0	0	0	0
	NonAnswers	8460	0	5280	4371	8460	6463
Q_3	MinExplain	6	6	0	0	0	0
	NonAnswers	8460	0	914	4371	8460	7978
Q_4	MinExplain	10	10	0	0	0	0
	NonAnswers	8460	0	672	4371	8460	0
Q_5	MinExplain	6	6	0	0	0	0
	NonAnswers	8460	0	3020	4371	8460	0

4 Experiments

In this section, we report our experimental results. We have implemented both MinExplain using in this paper and NonAnswers in [6]. All the algorithms were implemented with Java 1.7. Eclipse 4.2 was used as our Java IDE tool. The experiments were run on a PC with an Intel 3.10GHz Quad Core CPU i5 and 8GB memory with a 1TB disk, running a Windows 7 operating system. We use MySQL 5.5 as our DBMS and run it locally on the same machine.

4.1 Results of Explanations

We extract real data from one web site about air tickets information for airlines, air routes and ticket prices. The schema of the database D_B just like the one shown in Fig. 1. D_B consists of four tables, which are Airline, Route, Price, and Statistic. The data we use contain ten Airline tuples, twenty Route tuples, sixty Price tuples, and eight Statistic tuples.[1]

Assume that the attributes `Airline.ANO` and `Route.RNO` are *trusted attributes*. Table 5 shows six queries used in our experiments, in which the last column shows the missing answer questions.

Table 6 shows the number of explanations for queries $Q_0 - Q_5$ returned by MinExplain and NonAnswers. Taking query Q_0 as instance, NonAnswers returns 8460 explanations, while MinExplain only returns 3 minimal explanations. Table 6 also analyzes the quality of all explanations returned by NonAnswers and MinExplain. Those explanations returned by NonAnswers contains 0 correct explanations, 2328 explanations resulting in missing existed answers, 4371 explanations containing overabundance modifications, 8460 explanations violating functional dependency and 7191 explanations violating aggregation constraint.

[1] The data was extracted from `http://www.qunar.com/`

Table 7. Analysis of explanations for Q_0

Approaches	ANO	ANAME	RNO	LEAVE	TO	ANO	RNO	ECONOMY	BUSINESS	m-value
MinExplain	A2	CS	R3	Beijing	Guangzhou	A2	R7 → R3	950	1425	1
	A2	CS	R4	Beijing	Chengdu	A2	R7 → R4	950	1425	1
MinNonAnswer	A2	CS	R3	Beijing	Guangzhou	A2	R7 → R3	950 → n	1425	2
	A2	CS	R4	Beijing	Chengdu	A2	R7 → R4	950 → n	1425	2
	A2	CS	R7	Nanjing → Beijing	Hainan	A2	R7	950 → n	1425	2

Table 8. Modified mpTuples in query tables for Q_0

Algorithm \ table	Airline	Route	Price
ConsRefAttr	(A2,CS)	(R3,Beijing,Guangzhou) (R4,Beijing,Chengdu) (R16,Beijing,Guilin)	(A1,R1,760,1140) (A2,R7,950,1425) (A3,R9,900,1350) .....
VarRefAttr	(A2,CS)	(R3,Beijing,Guangzhou) (R4,Beijing,Chengdu) (R16,Beijing,Guilin)	(A2,R7 → R3, 950,1425) (A2,R7 → R4, 950,1425 (A2,R7 → R16, 950,1425

However, MinExplain only returns correct explanations. The results for the other five queries are similar to Q_0, and the details can be seen in Table 6.

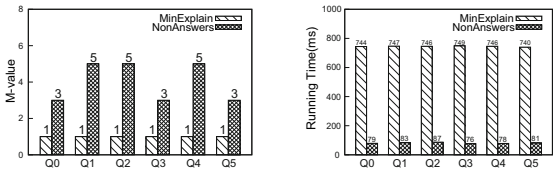
We combine our MinExplain with the approach NonAnswers and get a MinNonAnswer approach. The minimal explanations for Q_0 use MinExplain and MinNonAnswers respectively, which are shown in Table 7. MinExplain can return three minimal explanations and the m-value is 1. MinNonAnswer returns one more incorrect minimal explanations. For instance, the first and second explanations returned by MinNonAnswer both exist overabundance modification that modifying 950 to n, so their actual m-value should be 1. The third explanation returned by MinNonAnswer violates aggregation constraint, so its actual m-value should be 3 excluding the overabundance modification for Price.ECONOMY. While our approach can conquer those disadvantages and directly return correct minimal explanations for CS.

We generate the minimal explanations for missing answer CS in query Q_0 by using Algorithm 1 and Algorithm 2 respectively. We can get the modified mpTuples in each table as shown in Table 8. Algorithm 1 returns zero explanation for CS, because no join can be executed. Moreover, if executing a join, Price should contain one tuple just like (A2,R3,n,'?') or (A2,R4,n,'?') or (A2,R16,n,'?'). But once Price contain one tuple mentioned above, CS would be an existed answer instead of a missing answer. Algorithm 2 returns three explanations for CS just like those explanations returned by MinExplain in Table 7. So when the object values of those reference attributes in the query tables contain one or more variables, we should use Algorithm 2 to generate the minimal explanations.

4.2 Effects of M-Values and Minimal Explanation Number

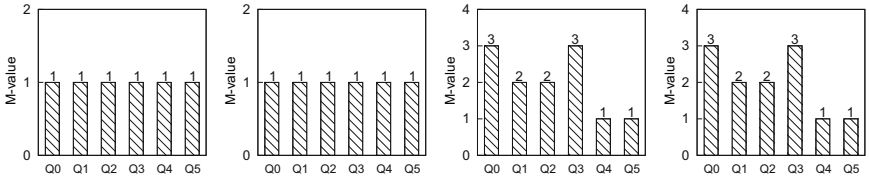
The m-value of returning minimal explanations for the same database is mainly affected by executing different queries, and the m-value of returning minimal explanations for the same missing answer is mainly affected by choosing different *trusted attributes* manually.

Fig. 4(a) illustrates the m-values for queries $Q_0 - Q_5$ by using MinExplain and NonAnswers, because the number of modified components of explanations



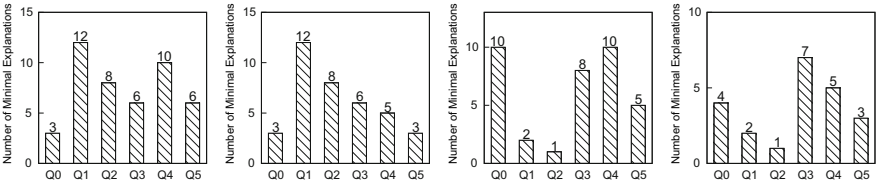
(a) M-values. (b) Running time.

Fig. 4. Comparison of m-values and running time under different settings



(a) No trust. (b) Price.ANO. (c) Price.RNO. (d) Price.ANO.RNO

Fig. 5. Comparison of m-values for $Q_0 - Q_5$ when choose different *trusted attributes*



(a) No trust. (b) Price.ANO. (c) Price.RNO. (d) Price.ANO.RNO

Fig. 6. Comparison of the number of minimal explanations for $Q_0 - Q_5$

returned by NonAnswers are very different, we use the average value of those numbers as the m-value of the explanations returned by NonAnswers. It can be seen that our approach can effectively return the minimal explanations.

Fig. 4(b) shows the running time for queries $Q_0 - Q_5$ by using MinExplain and NonAnswers respectively. We can see that the running time for queries $Q_0 - Q_5$ using our approach MinExplain is faster than that of using NonAnswer. One reason is that MinExplain discards those tuples which contributing to the existed query answers to avoid missing existed query answers. The other principle reason is that we only choose mpTuples in every query table to generate the minimal explanations for the missing answers, while NonAnswers choose all the tuples in every query table. We also can see that MinExplain is not very sensitive to different queries for the same query database.

Fig. 5 shows the m-values for queries $Q_0 - Q_5$ while trusting different attributes in Price. As shown in Fig. 5(a) and Fig. 5(b), we can get the minimal explanations whose m-value is 1 when we have no *trusted attributes* in Price or only trust `Price.ANO`. And it also shows that the minimal explanations don't modify the values of `Price.ANO`. But we can not modify `Price.RNO` to generate the minimal explanations while trusting `Price.RNO`. As shown in Fig. 5(c) and Fig. 5(d), we can only generate the minimal explanations whose m-value is 3

just like the third explanation excluding overabundance modification returned by NonAnswers in Table 7. Although additional *trusted attributes* can reduce the number of components which can be modified as mentioned before, it can also increase the m-values of the minimal explanations.

The number of minimal explanations for same missing answers is also mainly affected by choosing different *trusted attributes* manually.

Fig. 6 shows the number of minimal explanations for queries $Q_0 - Q_5$ while trusting different attributes in Price. Taking Q_0 as instance, Fig. 6(a) and Fig. 6(b) show the number of minimal explanations is unchanged while trusting `Price.ANO` which illustrates that the minimal explanations don't modify the value of `Price.ANO`. Fig. 6(c) and Fig. 6(d) show the number of minimal explanations is becoming smaller while trusting `Price.ANO`, which illustrates that some minimal explanations in Fig. 6(c) modify the value of `Price.ANO`. The number of other queries is showed in Fig. 6 in detail.

5 Conclusion and Future Work

We have developed a new framework to explain missing answers. We give two algorithms for minimal explanations. Our experiments show that our approach can effectively return minimal explanations for missing answers correctly. As parts of future work, we will further study minimize explaining missing answers, including an alternative definition of the minimal explanations for missing answers, other faster and effective algorithms.

References

1. Chang, C., Kayed, M., Girgis, M.R., Shaalan, K.F.: A Survey of Web Information Extraction Systems. TKDE, 1411–1428 (2006)
2. Buneman, P., Khanna, S., Tan, W.C.: Why and Where: A Characterization of Data Provenance. In: ICDT, pp. 316–330 (2001)
3. Cheney, J., Chiticariu, L., Tan, W.C.: Provenance in Databases: Why, How and Where. Foundations and Trends in Databases, 379–474 (2009)
4. Alavic, B., Alonso, G.: Perm: Processing provenance and data on the same data model through query rewriting. In: ICDE, pp. 174–185 (2009)
5. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance Semirings. In: PODS, pp. 31–40 (2007)
6. Huang, J., Chen, T., Doan, A., Naughton, J.F.: On the Provenance of Non-Answers to Queries over Extracted Data. In: PVLDB, pp. 736–747 (2008)
7. Chapman, A., Jagadish, H.: Why not'?'. In: SIGMOD, pp. 523–534 (2009)
8. Herschel, M., Hernández, M.A., Tan, W.C.: Artemis: A System for Analyzing Missing Answers. In: PVLDB, pp. 1550–1553 (2009)
9. Herschel, M., Hernández, M.A.: Explaining Missing Answers to SPJUA Queries. In: PVLDB, pp. 185–196 (2010)
10. Tran, Q.T., Chan, C.-Y.: How to ConQueR Why-not Questions. In: SIGMOD, pp. 15–26 (2010)
11. Zhian, H., Eric, L.: Answering Why-not Questions on Top-k Queries. In: ICDE, pp. 750–761 (2012)
12. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. TPLP, 68–79 (2003)

A Compact and Efficient Labeling Scheme for XML Documents

Rung-Ren Lin[1], Ya-Hui Chang[4,⋆], and Kun-Mao Chao[1,2,3]

[1] Department of Computer Science and Information Engineering
[2] Graduate Institute of Biomedical Electronics and Bioinformatics
[3] Graduate Institute of Networking and Multimedia
National Taiwan University, Taipei, Taiwan
{r91054,kmchao}@csie.ntu.edu.tw
[4] Department of Computer Science and Engineering
National Taiwan Ocean University, Keelung, Taiwan
yahui@ntou.edu.tw

Abstract. As XML data nowadays are extensively used in the applications of data exchange and other fields, supporting efficient query processing on XML data, particularly in determining the structural relationships between two elements, is in great demand recently. To avoid the time-consuming tree traversal tasks, many labeling schemes have been proposed to assign each node a unique label, so that the structural relationships between nodes, such as the ancestor-descendant relationship, can be efficiently determined by comparing their labels. However, to the best of our knowledge, none of the existing labeling schemes can support *all* structural relationships in constant time and also require the least amount of space. In this paper, we propose a labeling scheme based on the concept of the complete tree, which is called the CT (complete-tree) labeling scheme. This labeling scheme is simple and the resultant labels are compact. We formally analyze its properties and perform an empirical evaluation between the CT labeling scheme and other state-of-the-art labeling schemes on different data sets. The experimental results show that the space requirement of our CT labeling scheme is superior to others in most cases. It is also demonstrated that this scheme can efficiently support all structural relationships and may perform even better than other labeling schemes.

Keywords: XML, labeling scheme, structural relationship.

1 Introduction

As XML data nowadays are extensively used in the applications of data exchange and other fields, supporting efficient query processing on XML data, particularly in determining the structural relationships between two elements, is in great demand recently. According to W3C specification [17], an XML document can

⋆ To whom all correspondence should be sent.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 269–283, 2013.

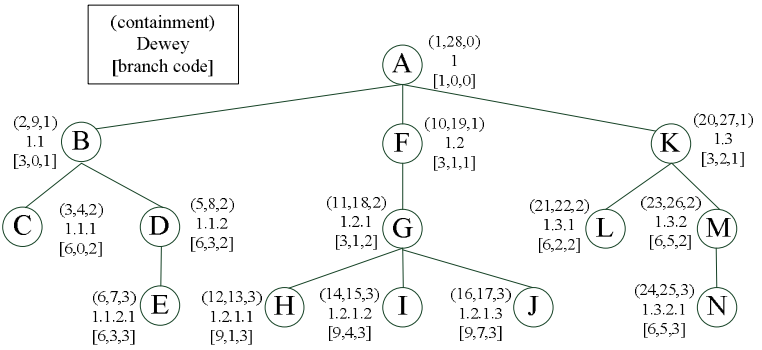


Fig. 1. A sample XML tree with the containment, Dewey, and branch labels

be modeled as a tree structure, where elements are represented as nodes, and the nested relationships between elements are represented as edges. In the associated query language such as XPath [18], eleven axes[1] are then designed to support path traversals in an XML tree, such as *ancestor*, *parent*, *preceding-sibling*, and *following-sibling*, etc. Since tree traversal may be time-consuming when the XML tree is large, labeling schemes are proposed to avoid such tasks. Specifically, given an XML tree, every node is assigned a unique label by the rules of the labeling schemes, so that we can determine the structural relationships of two nodes, such as the ancestor-descendant (AD), parent-child (PC), and sibling relationships, by directly comparing their labels without traversing the XML trees.

Among all the existing labeling schemes, *the containment labeling scheme* [5][16] is one of the most widely used ones. It refers to the type of labels which utilizes the *containment* relationship between labels to determine the structural relationship of nodes. Consider nodes G and H in Figure 1. The first two components of $G's$ containment label form the range $(11, 18)$, while those of $H's$ containment label form the range $(12, 13)$. Since the former range *contains* the latter one, we can conclude that G is an *ancestor* of H. On the other hand, *the prefix-based labeling scheme* [6][8][10][12] is also commonly applied. This scheme has the Dewey number as the representative, which includes the parent node's Dewey number as the prefix. For example, as depicted in Figure 1, node $H's$ Dewey number is 1.2.1.1, and its parent, node G, has the Dewey number 1.2.1.

These two labeling schemes are popular because they are basically simple and efficient, but there is still room left for improvement. Particularly, the containment labeling scheme can support AD and PC relationships efficiently, but cannot support the sibling relationship without extra information. On the other hand, the prefix-based labeling scheme can directly support all structural relationships, but is less efficient in supporting the AD and PC relationships than the containment labeling scheme. Recently, the *branch code* labeling scheme [14] is proposed. It keeps the number of siblings of all its ancestors, and can efficiently support the AD, PC, and sibling structural relationships, but it may be

[1] We omit the discussion of the *namespace* and *attribute* axes in this paper.

inefficient in determining the relationship of *the document order*, which plays an important role in processing XML queries when the ordering constraint is required. Another concern is related to space requirement, which is a critical issue for large XML documents.

In this paper, we propose a simple labeling scheme based on the concept of the complete tree. The main idea is that we expand the original tree to a "virtual" complete tree according to the maximum number of siblings of each level. We then assign a unique label, called *Complete-Tree-label*, or *CT-label* in short, for every node in the complete tree. The labels of the nodes in the original XML tree are assigned based on the labels associated with the complete tree. By this way, the CT-label *implicitly* encodes the maximum number of siblings of all levels into a single number, so it is very compact. For example, node H in Figure 1 has the CT-label "27", which is a single number. In contrast, the containment, Dewey, and branch code labels for node H are (12, 13, 3), 1.2.1.1, and [9, 1, 3], which require three to four numbers respectively. We also design a set of functions to help determine the eleven structural relationships based on CT-labels. All of these functions can be performed in constant time and thus are very efficient. To summarize, the contributions of this paper are as follows:

- We propose a simple labeling scheme, which assigns the CT-label to each node of an XML tree.
- The fundamental properties of our CT labeling scheme are analyzed and used to assist in efficiently determining the structural relationships between nodes.
- We have performed an empirical evaluation among the containment, Dewey, branch code, and CT labeling schemes. The results show that the proposed CT-labels are compact, since each node is only assigned a number and thus requires the least space most of the time.
- The experimental results also show that the CT labeling scheme is very efficient when computing all the structural relationships. Particularly, it always outperforms the Dewey labeling scheme when computing the AD, PC, and sibling relationships.

The rest of this paper is organized as follows. Section 2 introduces the containment, Dewey, and branch code labeling schemes. The proposed CT labeling scheme is described in Section 3. In Section 4, we show the experimental results of comparing these four labeling schemes. At last, related works and conclusions are given in Section 5 and Section 6, respectively.

2 Preliminaries

We explain the containment, the prefix-based, and the branch code labeling schemes in this section. Figure 1 will be used as the running example, and

the Dewey numbers will act as the representative of the prefix-base scheme hereafter. Due to space limitation, we only briefly discuss how these labeling schemes determine some basic structural relationships. Interested readers can refer to the original papers for more details.

In the containment labeling scheme, every node is assigned a label of three values $(start, end, level)$. The value of *start* increases according to the pre-order traversing, while the value of *end* is assigned according to the post-order sequence. Accordingly, the document order of two nodes can be naturally derived by the *start* values of the containment labels. In addition, the range formed by *start* and *end* of a given node is contained in the range of its ancestor. On the other hand, the *level* value, which starts from 0 in this paper, describes the depth of the node, and is used to assist the determination of the PC relationship.

The Dewey number is the representative of the prefix-based labeling scheme. In the Dewey labeling scheme, every node inherits the label from its parent and adds a new number at the end. The new number is generally enumerated from 1. For example, node H inherits node G's label (1.2.1) and adds "1" at the end, which leads H's label as 1.2.1.1. Node I also inherits node G (1.2.1) and adds "2" at the end since I is the second child of G, and thus the Dewey number of node I is 1.2.1.2. Therefore, we can determine the AD and PC relationships based on a node's prefix, and the last component can be used to identify the sibling order.

As to the branch code, each label primarily has three values $[g, h, level]$. The g value keeps the "number" of its ancestors' siblings, and the h value keeps the "order" of its ancestor among their siblings. Specifically, given a node n and its parent n_p, n's g value equals to n_p's g value multiple the number of n's siblings. n's h value equals to n_p's h value plus n_p's g value times the order of n among its siblings. For instance, nodes A, B, and D are ancestors of node E. The number of siblings of these three ancestors are 1, 3, and 2, respectively. Besides, the order of these three ancestors among their siblings are 0, 0, and 1, in which the order is counted from 0. Therefore, n's g value equals to $3 \cdot 2 \cdot 1 = 6$, and n's h value equals to $((0 + 1 \cdot 0) + 3 \cdot 1) + 6 \cdot 0 = 3$. The *level* value is the same as that of the containment scheme, so node E has the branch code $[6, 3, 3]$. By values g and h, we can determine the AD and siblings relationships in constant time. However, this labeling scheme costs $O(d)$ time to determine the document order, where d is the depth of the XML tree.

Note that the AD, PC, document order, and siblings relationships are fundamental to the eleven XPath axes. Specifically, the AD relationship can support the following axes: *ancestor*, *ancestor-or-self*, *descendant*, and *descendant-or-self*; the PC relationship can support the two axes: *parent* and *child*; the document order relationship can support the *following* and *proceeding* axes; the sibling relationship and the document order can support the two axes: *following-sibling* and *proceeding-sibling*. The *self* axe can be trivially determined based on the equality of labels. Therefore, our focus will be on these four particular structural relationships.

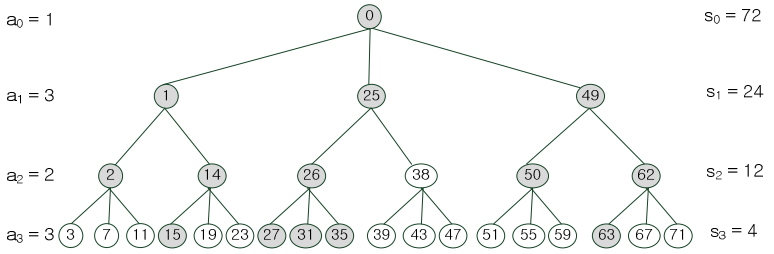


Fig. 2. The corresponding complete tree of the original XML tree

3 The CT Labeling Scheme

We illustrate our CT (complete-tree) labeling scheme in this section, including the basic concept, properties of CT labels, and the pseudocodes of determining the structural relationships of two CT-labels. The tree structure of Figure 1 is continually used as the sample tree of our CT labeling scheme, and the CT-label of a node is also used to represent the node for simplicity.

3.1 Basic Concept

Given an XML tree, the maximum number of siblings of each level is a constant. Let d be the maximum depth of the XML tree and a_k be the maximum number of siblings of level k. For example, $(a_0, a_1, ..., a_{d-1})$ of the sample XML tree in Figure 1 are (1, 3, 2, 3). Recall that the depth is enumerated from 0 in our framework. The concept of assigning CT labels could be decomposed into four steps as follows:

1. Expand the original XML tree to a complete tree according to the maximum number of siblings of each level $(a_0, a_1, ..., a_{d-1})$. Figure 2 shows the complete tree of the sample XML tree.
2. Label the leaf nodes. The CT-label of the left-most leaf node is numbered as $d - 1$. For example, d is 4 in the sample XML tree in Figure 1, and the leftmost child will be assigned 3. Subsequently, every leaf node is increased by d from left to right. Therefore, the CT-labels of leaf nodes form an arithmetic sequence with common difference d.
3. Label the internal nodes. The CT-label of every internal node is equal to the CT-label of its first child minus 1.
4. Map the original XML tree to the complete tree. Since the original one is a subtree of the expanded complete tree, the one-to-one mapping is feasible between these two trees. The nodes with gray color in Figure 2 construct the same tree structure of the original XML tree. Figure 3 displays the sample XML tree with CT-labels.

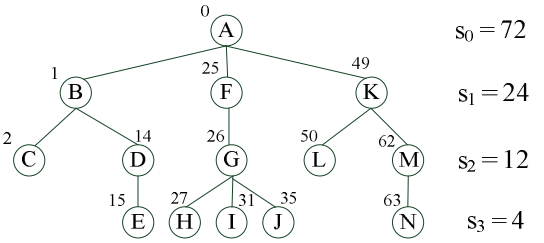


Fig. 3. A sample XML tree with CT-labels

Note that the first step, which expands the original XML tree to a complete tree, is actually not required, because it is possible to label the CT-labels for all the nodes by preorder traversing the original XML tree. We will have more details in Section 3.3.

3.2 Properties

Now we discuss some properties which may be used for determining the structural relationships of two CT-labels. Note that these properties are discussed based on the complete tree.

Property 1. *The CT-labels from left to right of the same level form an arithmetic sequence.*

Proof. *Undoubtedly, the CT-labels of leaf nodes are an arithmetic sequence. Recall that the CT-label of every internal node is the CT-label of its first child minus 1. Since the CT-labels are based on the complete tree, the "gaps" formed by their first child are identical. That is, the CT-labels of the first child of the same level are also an arithmetic sequence. Therefore, this property still holds for those nodes in the same level.* □

Recall that a_k represents the maximum number of siblings of level k. Let s_k be the common difference of level k ($0 \leq k \leq d-1$). We know that $s_{d-1} = d$. Besides, by the characteristic of the complete tree, we obtain that $s_k = s_{k+1} \cdot a_{k+1} = \Pi_{i=k+1}^{d-1} a_i \cdot d$, where $0 \leq k \leq d-2$. For example, (s_0, s_1, s_2, s_3) of Figure 2 are $(72, 24, 12, 4)$.

Property 2. *The depth of a CT-label x is equal to the remainder of x divided by d.*

Proof. *The initial term of the arithmetic sequence of level k is exactly k, and the remainder of k divided by d is also k. Consider the CT-labels of level k. Since s_k (the common difference of level k) is a multiple of d, the remainder of any CT-label of level k divided by d is certainly k.* □

Property 3. *Given two distinct CT-labels x and y, and suppose x is of depth k, x is the ancestor of y if and only if $x < y$ and $x + s_k - k > y$.*

Proof. *Let T be the subtree rooted at x. It is clear that x is the minimum CT-label of T, and the maximum CT-label is the right-most leaf node of T. In addition, the number of leaf nodes of T is equal to $\Pi_{i=k+1}^{d-1} a_i$. Suppose z is the right-most leaf node of T. We can derive z by the arithmetic sequence formula as follows:*

$$
\begin{aligned}
& z = x + d - 1 - k + (\Pi_{i=k+1}^{d-1} a_i - 1) \cdot d \\
\Rightarrow\ & z = x - 1 - k + \Pi_{i=k+1}^{d-1} a_i \cdot d \\
\Rightarrow\ & z = x - 1 - k + s_k \\
\Rightarrow\ & x + s_k - k > z
\end{aligned}
$$

If y is a descendant of x, y must be in the range between x and z. That is, $x < y \leq z$. Hence, x is the ancestor of y if and only if $x < y$ and $x + s_k - k > y$. Besides, the parent-child relationship can also be determined if the depth of x is exactly equal to the depth of y minus one. □

Property 4. *Let $T(j,k)$ be the subtree rooted at the j^{th} node of level k. Suppose $L_{min}(L_{max})$ is the minimum (maximum) CT-label of $T(j,k)$. We can derive that $\lfloor L_{min}/s_k \rfloor = \lfloor L_{max}/s_k \rfloor = j - 1$.*

Proof. *As mentioned in Property 3, L_{min} is the root of $T(j,k)$ and is equal to $(k + (j-1) \cdot s_k)$, since it is the j^{th} node of level k. In addition, L_{max} is equal to $(L_{min} - 1 - k + s_k)$. We now consider the floor of L_{min} and L_{max} divided by s_k:*

$$
\begin{aligned}
& \lfloor L_{min}/s_k \rfloor \\
=\ & \lfloor (k + (j-1) \cdot s_k)/s_k \rfloor \\
=\ & \lfloor k/s_k \rfloor + j - 1 = j - 1
\end{aligned}
$$

$$
\begin{aligned}
& \lfloor L_{max}/s_k \rfloor \\
=\ & \lfloor (L_{min} - 1 - k + s_k)/s_k \rfloor \\
=\ & \lfloor (k + (j-1) \cdot s_k - 1 - k + s_k)/s_k \rfloor \\
=\ & \lfloor (s_k - 1 + (j-1) \cdot s_k)/s_k \rfloor \\
=\ & \lfloor (s_k - 1)/s_k \rfloor + j - 1 = j - 1
\end{aligned}
$$

□

According to Property 4, if the floors of two CT-labels divided by s_k are identical, it indicates that they have the same ancestor at level k. We next give an example for Properties 2, 3, and 4.

Example 1. *Consider Figure 2. Recall that the depth is counted from 0, and thus the maximum depth is 4. By Property 2, the depths of nodes with CT-labels 26 and 51 are 2 and 3, respectively. By Property 3, node 35 is the descendant of node 26 since $26 < 35$ and $26 + 12 - 2 > 35$. According to Properties 4, nodes 26 and 38 are siblings because both nodes 26 and 38 are at level 2 and $\lfloor 26/24 \rfloor = \lfloor 38/24 \rfloor$. However, nodes 14 and 26 are not siblings which could be verified in a similar way.* □

3.3 Pseudocodes

In this subsection, we first provide the pseudocode which assigns the CT-label for each node by preorder traversing the XML tree. We then show the pseudocodes of determining the structural relationships of two CT-labels, which are based on

Input: An XML tree T.
Output: Tree T with CT-label for each node.

```
CT_labeling(T)
1  for every node n in T do
2      n.num ← 0
3  for i ← 0 to d − 1 do
4      a_i ← 0
5  for every child node n_c of root do
6      GetMaxSiblings(n_c, 1)
7  s_{d−1} ← d
8  for i ← d − 2 to 0 do
9      s_i = s_{i+1} · a_{i+1}
10 root.label ← 0
11 LabelNodes(root, 0)

GetMaxSiblings(n, k)  /*n: a node, k: current level*/
1  n_p ← n.parent
2  n_p.num ← n_p.num + 1
3  if a_k < n_p.num then
4      a_k ← n_p.num
5  for every child node n_c of n do
6      GetMaxSiblings(n_c, k + 1)

LabelNodes(n, k)
1  j ← 0
2  for every child node n_c of n do
3      n_c.label ← n.label + 1 + j · s_{k+1}
4      j ← j + 1
5      LabelNodes(n, k + 1)
```

Fig. 4. The pseudocode of the *CT_labeling* procedure

the properties discussed in the previous subsection. Note that the original XML tree has never really been expanded to the complete tree during the assignment of the CT-label for each node. The reason is that once the CT-label of the parent node is given, we can directly infer the CT-labels of its children. Suppose y is the j^{th} child of the parent node x. Then, y is equal to $x + 1 + (j-1) \cdot s_{d_y}$, where d_y is the depth of y.

The pseudocode of assigning the CT-labels for an XML tree is shown in Figure 4. Variable $n.num$ of line 2 is used to record the current number of n's children. Hence, $n.num$ is initialized as 0 in the beginning. In line 6, procedure $GetMaxSiblings$ is called recursively to get the number of maximum siblings of each level by traversing the tree in preorder. We next compute the common difference for each level in lines 7 to 9. Procedure $LabelNodes$ also traverses the XML tree in preorder, and gives the CT-labels for all the children of node n. Observe that we traverse the tree exactly twice in Figure 4. One is to compute the maximum number of siblings of each level, and the other is to compute the

```
IsAncestor(x, y)
1  if x < y then
2      k ← x mod d
3      if x + s_k − k > y then
4          return true
5  return false

IsParent(x, y)
1  if x < y then
2      k ← x mod d
3      if x + s_k − k > y and k + 1 = y mod d then
4          return true
5  return false

IsSibling(x, y)
1  d_x ← x mod d
2  d_y ← y mod d
3  if d_x = d_y and ⌊x/s_{d_x−1}⌋ = ⌊y/s_{d_x−1}⌋ then
4      return true
5  return false
```

Fig. 5. The pseudocode of determining the structural relationships of two CT-labels

CT-label for each node. In addition, the cost of visiting a node can be done in constant time. Therefore, the time complexity of procedure *CT_labeling* is in linear proportion with the size of the XML tree.

The pseudocodes of determining the structural relationships of two CT-labels are shown in Figure 5. The document order relationship is not listed because it can be directly determined by comparing the value of two CT-labels. The $IsAncestor(x, y)$ procedure returns *true* if x is the ancestor of y. According to Property 3, if the statements of both lines 1 and 3 are true, x is the ancestor of y. Otherwise, procedure $IsAncestor$ returns $false$. The pseudocode of procedure $IsParent$ is quite similar to that of procedure $IsAncestor$. However, it additionally checks whether the depth of x is equal to the depth of y minus one, which is an essential condition of the PC relationship. In the $IsSibling(x, y)$ procedure, the depths of x and y are represented as d_x and d_y. We first check whether x and y are at the same level. We then check whether $\lfloor x/s_{d_x-1} \rfloor$ is equal to $\lfloor y/s_{d_x-1} \rfloor$. If so, it indicates that x and y have the same parent. Accordingly, x and y are siblings. All the three procedures mentioned above run in $O(1)$ time. Further, the ancestor of x at the k^{th} level cab be located by the formula "$a_k + \lfloor x/s_k \rfloor \cdot s_k$".

Table 1 summarizes the eleven XPath axes, and the time complexity of determining the corresponding structural relationship performed by the four labeling schemes, respectively. The last column also explains how our CT-labels determine the particular relationship. For example, the first row describes how to determine if a node x is the *parent* of the other node y. In our framework, we can directly invoke the function *IsParent* to achieve this task, which has the time

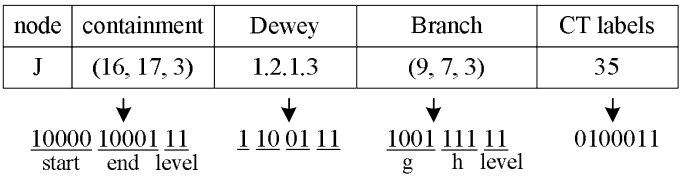


Fig. 6. The sample bit-strings of node J for the four labeing schemes

Table 1. The time complexity of determining all structural relationships by the four labeling schemes

XPath axes	containment labeling	Dewey labeling	Branch labeling	CT labeling	CT-labels implementation
parent	$O(1)$	$O(d)$	$O(1)$	$O(1)$	$IsParent(x, y)$
ancestor	$O(1)$	$O(d)$	$O(1)$	$O(1)$	$IsAncestor(x, y)$
ancestor-or-self	$O(1)$	$O(d)$	$O(1)$	$O(1)$	$x = y$ **or** $IsAncestor(x, y)$
child	$O(1)$	$O(d)$	$O(1)$	$O(1)$	$IsParent(y, x)$
descendant	$O(1)$	$O(d)$	$O(1)$	$O(1)$	$IsAncestor(y, x)$
descendant-or-self	$O(1)$	$O(d)$	$O(1)$	$O(1)$	$x = y$ **or** $IsAncestor(y, x)$
following	$O(1)$	$O(1)$	$O(d)$	$O(1)$	$x > y$
preceding	$O(1)$	$O(1)$	$O(d)$	$O(1)$	$x < y$
following-sibling	N/A	$O(d)$	$O(1)$	$O(1)$	$x > y$ **and** $IsSibling(x, y)$
preceding-sibling	N/A	$O(d)$	$O(1)$	$O(1)$	$x < y$ **and** $IsSibling(x, y)$
self	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$x = y$

complexity $O(1)$. The time complexity of the other schemes are also listed here for comparison. As shown in this table, our CT-label has the best time complexity in determining *all* structural relationships. The containment labeling scheme also has the best time complexity in many cases, but it cannot determine some of the structural relationships. In contrast, the Dewey labeling scheme usually cannot achieve constant-time computation, and the branch labeling is particularly inefficient in processing the *following* and *preceding* axes. In the next section, we will deliver more detailed performance studies.

4 Performance Studies

In this section, we discuss the implementation issues and compare the space and time performance among the containment labeling scheme, Dewey numbers, branch codes, and the CT labeling scheme. All the labeling schemes were implemented in C++ with the environment of Windows 7 and Visual Studio 2005. We applied six data sets[2] to perform the experiments. Table 2 displays the basic information of the data sets, including the file size of the XML tree, the number of nodes, the maximum depth, and the average depth. In addition, the experiments were performed on a 2.83GHz quad-core CPU with 4.0GB RAM.

[2] These data sets could be downloaded from:
`http://www.xml-benchmark.org/downloads.html/`
`http://www.cs.washington.edu/research/xmldatasets/`

Table 2. The basic information of the data sets

Data Set	Doc. Size	Nodes	Max/Avg Depth
Sigmod Rec.	467 KB	19,909	7/5.7
Reed	277 KB	18,855	5/3.7
NASA	23.8 MB	904,556	9/6.1
XMark	25 MB	571,775	12/7.1
Mondial	1.7 MB	124,736	7/4.6
DBLP	127 MB	7,146,530	7/3.5

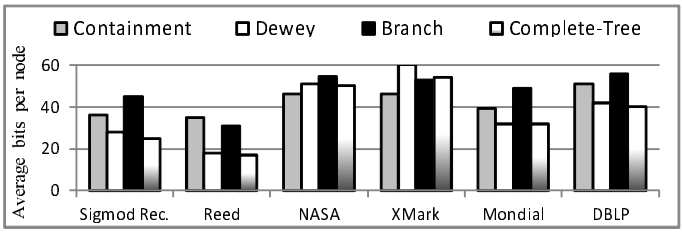


Fig. 7. The label size of the data sets

4.1 Space Analysis

In this subsection, we explain the data structures of the four labeling schemes used in our experiments and discuss their space requirement. For each labeling scheme, we basically use the *minimum* numbers of bits to represent the required values.

First, for the containment labeling scheme, suppose the XML tree has N nodes in total. The maximum value of *end* will be equal to $2N$, so we need $2 \cdot \log 2N$ bits to record *start* and *end*, where "log" represents the base-2 logarithm in this paper. In addition, *level* requires log$d$ bits where d is the maximum depth of the XML tree. Therefore, every containment label requires $2 \cdot \lceil \log 2N \rceil + \lceil \log d \rceil$ bits. For instance, there are totally 14 nodes in the sample XML tree of Figure 1. Hence, every containment label requires $2 \cdot \lceil \log 28 \rceil + \lceil \log 4 \rceil = 12$ bits. The bit-string of node J corresponding to the containment labeling scheme is shown in the lower left of Figure 6. The *start* value is located before *end* and *level* because it allows us to directly compare the document order without decoding the value of *start* from the bit-string.

As to the Dewey numbers, they are implemented according to the data structure discussed in [15]. Specifically, every Dewey number has d entries, and each entry uses the *smallest* number of bits needed to store the *maximum* Dewey component number. Hence, every Dewey number requires $\lceil \log(a_0+1) \rceil + \lceil \log(a_1+1) \rceil + ... + \lceil \log(a_{d-1}+1) \rceil$ bits.[3]

Recall that each branch code is composed of three values $(g, h, level)$. It is relatively complicated to give a standard formula for its storage requirement since the space cost of each node varies with the number of ancestors' siblings. In our

[3] Each Dewey component number is enumerated from 1 in our experiments, because 0 is used to denote the end of the Dewey component number.

experiments, we basically use the *smallest* number of bits needed to store values g and h. Besides, the cost of value *level* is the same as that of the containment scheme.

Finally, in our CT labeling scheme, the maximum value of the CT-label appears at the right-most leaf node of the complete tree. Recall that the initial term of the leaf level is $d-1$, and the common difference is d. Since there are $\Pi_{i=0}^{d-1} a_i$ leaf nodes in total, the CT-label of the right-most leaf node is $\Pi_{i=0}^{d-1} a_i \cdot d - 1$. Hence, every CT-label requires $\lceil (\log a_0 + \log a_1 + ... + \log a_{d-1} + \log(d-1)) \rceil$ bits. Compare the branch code and the CT-label. Though the value of a CT-label is always larger than the g (or h) value of a branch code, the difference between them is usually small in logarithmic scale. In addition, it requires to store both g and h when using branch codes. Hence, the space cost of the CT labeling scheme is generally better than that of branch code. Please see the two sample bit-strings in the lower right of Figure 6 for comparison.

Based on the data structures discussed above, the resultant label sizes of the six data sets are shown in Figure 7. We can see that the containment labeling scheme and branch codes have the worst performance when the depth of the data set is small, while the Dewey labeling scheme is the worst when the depth of the data set is large. Note that the space cost of the containment labeling scheme has nothing to do with the depth of the XML tree, but the other three labeling schemes do, since they either explicitly or implicitly represent ancestors' information. This is the reason why the XMark data set has less nodes than those of the DBLP data set, but the Dewey number, the branch code, and the CT-label need even more space. However, our CT-labeling scheme still performs better than the other two. To summarize, our CT labeling scheme has the best performance when the depth of the data set is small, and only requires a little more space than the containment labeling scheme when the depth of the data set is large.

4.2 Time Analysis

The time performance is also measured based on the data sets listed in Table 2. After the XML tree and the four types of labels for every node are constructed by parsing through an XML document, we randomly create a million pairs of nodes as the input source. Each pair of node tests the structural relationships using the four labels. In addition, according to our experiments, we found that the *module* operator (the symbol "%" in C++ language) is slower than the other CPU instructions, such as addition, shift, and AND operators. Since computing the remainder of two integers plays an important role in our framework, we modify the representation of the maximum depth of the XML tree d to accelerate the computing of the depth of a CT-label. Specifically, the real depth is represented as $2^{\lceil \log d \rceil}$ so that we can use the AND operator to compute the depth of a CT-label instead of using the module operator.

The experimental results of the time performance for each data set are displayed in Figure 8. The processing time is the total time consumed to determine a structural relationship for all the one-million-pair nodes. We can see that it

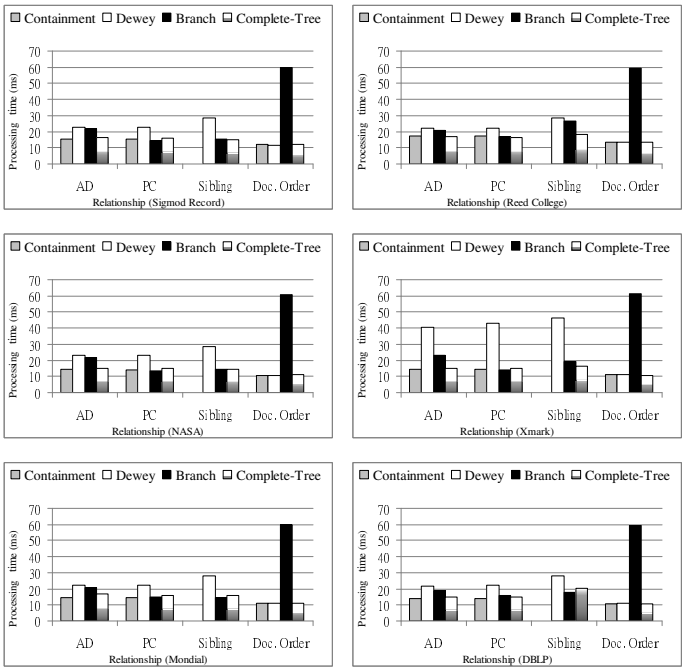


Fig. 8. The processing time of determining the structural relationships with different data sets

costs more time for the Dewey numbers to determine the AD and PC relationships than the other labeling schemes. The reason is that we have to compare each component of the Dewey number while the other labeling schemes are able to compute the relationships in constant time. Besides, we can also see that the branch code is much inefficient than the others in determining the document order. The reasons are that the time complexity of the branch scheme is $O(d)$, and many time-consuming module operators may be used. The containment labeling scheme cannot be used to compute the sibling relationship, so we compare only the Dewey numbers, branch codes, and CT-labels for the sibling relationship. It is obvious that our CT labeling scheme works more efficiently than the Dewey number on determining the sibling relationship. The reason is quite similar to that of the AD and PC relationships.

In summary, the experimental results show that our CT labeling scheme works almost as efficiently as the containment labeling scheme for computing the AD and PC relationships, but the CT labeling scheme further provides the ability of determining the sibling relationship. In addition, the experimental results also show that the CT labeling scheme often works more efficiently than the Dewey number and the branch code for computing the AD, PC, and sibling relationships.

5 Related Work

Many labeling schemes have been proposed to process XPath efficiently. For example, the authors in [2] used P-labeling and D-labeling to process the queries involving consecutive child axes and descendant axes traversal, and the authors in [1] discussed how to process XPath based on relational databases. Besides, the techniques discussed in [7][9][11] have been proposed to process twig patterns in XML databases. We do not compare our labeling schemes with all of them since their labeling scheme tend to be more complex and each label requires more information than the other schemes discussed in this paper.

Note that all of the four labeling schemes discussed in this paper are not *fully dynamic*, that is, re-labeling is sometimes required if we perform update operations on XML documents. Specifically, in the containment labeling scheme, the whole XML tree should be re-labeled once a node is newly added. In the Dewey labeling scheme, all the nodes that follow the newly added node should be re-labeled. As to the branch code labeling scheme, the subtree rooted at the parent of the newly added node will be re-labeled. For our CT-labeling scheme, if the newly inserted node exceeds the range covered by the corresponding complete tree, the whole tree will need to be re-labeled. For example, if we add a new node following node E of Figure 3, it can be assigned the number 19. However, if we add a new node following node D, there is no room for new labels and the whole tree should be re-labeled.

Several *dynamic* labeling schemes have been proposed to avoid the task of re-labeling[3][4][8][12][13]. Particularly, some of them are the extensions of Dewey numbers. For example, in the ORDPATH labeling scheme proposed by O'Neil *et al.* [8], every label is composed of only odd numbers at the initial states. Hence, there exists a gap between two adjacent siblings for future insertion. This labeling scheme is currently used in Microsoft SQL Server. On the other hand, Wu *et al.* used the concept of the vector order to develop a dynamic labeling scheme called Dynamic DEwey (DDE) [12]. In short, given two DDE sibling labels, the newly inserted label is the addition of the component at the same level, and we can compare the structural relationships of two DDE labels by their initial terms. However, these methods cannot be applied to the other three labeling schemes.

In summary, the proposed CT labeling scheme is proper for read-only XML documents. If we expect there will be node insertions in the future, we can leave some gaps in advance, as the containment labeling scheme does. Although not fully dynamic, this scheme can already work for most cases. Particularly, note that the contents of XML documents usually do not change in most application scenarios. For example, we will not change the XML document if it is a purchase order received through data exchange.

6 Conclusions

In this paper, we propose a labeling scheme based on the concept of the complete tree. According to the experimental results, the CT labeling scheme is very efficient in processing *all* structural relationships. It is also shown that our labeling

scheme is more compact than the others in most cases. In the future, we will investigate how to make the CT labeling scheme fully dynamic. The challenge lies in how to make the CT-label compact and flexible at the same time.

References

1. Chen, L.J., Bernstein, P.A., Carlin, P., Filipovic, D., Rys, M., Shamgunov, N., Terwilliger, J.F., Todic, M., Tomasevic, S., Tomic, D.: Mapping XML to a Wide Sparse Table. In: ICDE, pp. 630–641 (2012)
2. Chen, Y., Davidson, S.B., Zheng, Y.: BLAS: an Efficient XPath Processing System. In: SIGMOD, pp. 47–58 (2004)
3. Härder, T., Haustein, M.P., Mathis, C., Wagner, M.: Node Labeling Schemes for Dynamic XML Documents Reconsidered. Data & Knowledge Engineering 60, 126–149 (2007)
4. Li, C., Ling, T.W.: QED: a Novel Quaternary Encoding to Completely Avoid Relabeling in XML Updates. In: CIKM, pp. 501–508 (2005)
5. Li, Q., Moon, B.: Indexing and Querying XML Data for Regular Path Expressions. In: VLDB, pp. 361–370 (2001)
6. Lin, R.-R., Chang, Y.-H., Chao, K.-M.: Identifying Relevant Matches with NOT Semantics over XML Documents. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part I. LNCS, vol. 6587, pp. 466–480. Springer, Heidelberg (2011)
7. Lu, J., Lin, T.W., Chan, C.-Y., Chen, T.: From Region Encoding to Extended Dewey: on Efficient Processing of XML Twig Pattern Matching. In: VLDB, pp. 193–204 (2005)
8. O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Schaller, G., Westbury, N.: ORDPATHs: Insert-Friendly XML Node Labels. In: SIGMOD, pp. 903–908 (2004)
9. Rao, P., Moon, B.: PRIX: Indexing And Querying XML Using Prufer Sequences. In: ICDE, pp. 288–300 (2004)
10. Tatarinov, I., Viglas, S., Beyer, K.S., Shanmugasundaram, J., Shekita, E.J., Zhang, C.: Storing and Querying Ordered XML Using a Relational Database System. In: SIGMOD, pp. 204–215 (2002)
11. Tatikonda, S., Parthasarathy, S., Goyder, M.: LCSTRIM: Dynamic Programming Meets XML Indexing and Querying. In: VLDB, pp. 63–74 (2007)
12. Wu, L., Ling, T.W., Wu, H., Bao, Z.: DDE: From Dewey to a Fully Dynamic XML Labeling Scheme. In: SIGMOD, pp. 719–730 (2009)
13. Wu, X., Lee, M.-L., Hsu, W.: A Prime Number Labeling Scheme for Dynamic Ordered XML Trees. In: ICDE, pp. 66–78 (2004)
14. Xiao, Y., Hong, J., Cui, W., He, Z., Wang, W., Feng, G.: Branch Code: A Labeling Scheme for Efficient Query Answering on Trees. In: ICDE, pp. 654–665 (2012)
15. Xu, Y., Papakonstantinou, Y.: Efficient Keyword Search for Smallest LCAs in XML Databases. In: SIGMOD, pp. 527–538 (2005)
16. Zhang, C., Naughton, J.F., DeWitt, D.J., Luo, Q., Lohman, G.M.: On Supporting Containment Queries in Relational Database Management Systems. In: SIGMOD, pp. 425–436 (2001)
17. DOM Level 3 Core Specification (2004), `http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/`
18. XQuery and XPath Full Text 1.0 (2011), `http://www.w3.org/TR/xpath-full-text-10/`

Querying Semi-structured Data with Mutual Exclusion

Huayu Wu[1,2], Ruiming Tang[3], and Tok Wang Ling[3]

[1] State Key Lab of Software Engineering, Wuhan University, China
[2] Institute for Infocomm Research, A*STAR, Singapore
huwu@i2r.a-star.edu.sg
[3] School of Computing, National University of Singapore
{tangruiming,lingtw}@comp.nus.edu.sg

Abstract. Data analytics applications, content-based collaborative platforms and office applications require the integration and management of current and historical data from heterogeneous sources. XML is a standard data format for information. Thanks to its semi-structured-ness, it is a good candidate data model for the integration and management of heterogeneous content. However, the management of historical and collaboratively created data compels considering extensions of the original XML model to constraint-based, probabilistic and temporal aspects. We consider here an extension of the XML data model with mutual exclusion between nodes for the purpose of the management of versions in XML databases. XML query processing algorithms for ordinary XML data focus on the parent-child, ancestor-descendant, sibling and lowest common ancestor relationships between nodes. In this paper, we extend existing labeling schemes and query processing algorithms for the processing of queries over an extension of the XML data model with mutual exclusion. We focus on structured twig pattern query, and show that the same technique can be applied to keyword queries as well. We empirically evaluate the performance of the proposed techniques.

1 Introduction

1.1 Context

Data analytics is increasingly used by organizations to make better business decision, improve customer services, and verify existing or discover new operational models. At the same time, different applications regularly or continuously collect, integrate and maintain data to serve analytics tools. Those data may be large in size, and pose speciality compared to normal business data. The capacity and throughput of modern systems tackles data size, while special handling may be required to meet the speciality of such data.

We use two examples to illustrate data speciality. In many situations, data need to be collected and integrated from multiple sources. Uncertainty is unavoidable in such kind of data integration. To store and query uncertain data, *probabilistic data management* ([4][5]) were studied. The second example is *multi-versioned data management*. Data may change over time. With the development of hardware

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 284–298, 2013.

capacity, many old-versioned data are no longer stored as "log", but stored in the same database as the current version. Analytical tasks often visit different versions of data by "time-travel" queries [16].

Although there are adequate works extending the traditional relational database to support storage and query for data with special characteristics, more and more modern data are represented, exchanged and archived in semi-structured or even unstructured format rather than relational model. XML has become a standard format for information exchange and storage. Our study focuses on semi-structured XML database.

There are a number of extensions to XML in order to represent data with different speciality. Continuing with the above two examples, probabilistic XML model [15] and temporal XML model [17] were proposed to represent uncertain XML data and multi-versioned XML data respectively. Because of the simple but strict syntax requirement of XML, any extension to XML data only puts in additional tags to meet its speciality. The two XML fragments in Fig. 1 show a probabilistic XML data and a temporal XML data, in which the colored italic element tags and attributes are introduced to indicate distribution and probability, and time window in the two documents respectively. In the probabilistic XML data in Fig. 1(a), the container throughput of the port of Hong Kong in each year may exist with a probability, and are independent to each other. Furthermore, the information for 2009 came from two sources with different probabilities, and the two pieces of information are mutually exclusive to each other. In the temporal XML data in Fig. 1(b), the information of 2008 and 2009 have been updated at different times, and different versions of information are all kept with a time window to indicate their valid period.

```
<ports>
 <port name= "Hong Kong">
  <container_throughput>
   <ind>
    <2007 pro= "0.7">23.998</2007>
    <2008 pro= "0.8">24.494</2008>
    <mux pro= "0.9">
     <2009 pro= "0.8" source= "HKPDC">21.040</2009>
     <2009 pro= "0.2" source= "Asian News">20.9</2009>
    </mux>
   </ind>
  </container_throughput>
 </port>
 ......
</ports>
```

(a) Probabilistic XML

```
<ports>
 <port name= "Hong Kong">
  <container_throughput>
   <2007>23.998</2007>
   <2008 time= "[2008, 2009]">24.494</2008>
   <2008 time= "[2009, now]">24.89</2008>
   <2009 time= "[2009, 2011]">21.040</2009>
   <2009 time= "[2011, now]">20.9</2009>
  </container_throughput>
 </port>
 ......
</ports>
```

(b) Temporal XML

Fig. 1. Fragments of probabilistic and temporal XML data

1.2 Motivation and Contribution

Most works on query processing in extended XML models only focus on functionality extension (e.g., [8][7] for temporal XML and [11][10] for probabilistic XML), ignoring the performance gap between their novel algorithms and the many efficient algorithms proposed for ordinary XML data. Twig pattern matching and keyword search are two typical topics in XML query processing,

for structured and unstructured XML queries respectively. Among the state-of-the-art algorithms for both twig pattern matching and XML keyword search, document labeling and inverted list for labels are widely adopted to achieve better performance. In particular, for twig pattern matching, efficient structural join algorithms e.g., [6], were proposed over inverted list streams, while for keyword search, relevant inverted list streams are scanned in order to find LCA nodes e.g., [19]. It is proven that using document labeling and inverted lists, both I/O and computational cost during query processing can be significantly improved [9], compared to some naive approaches, e.g., document sequential scan.

Despite the considerable research efforts dedicated to query processing over the ordinary XML model, few of them were adapted to extended XML models. One of the key reasons is that the current labeling schemes (e.g., the containment scheme [20] and the prefix scheme [18]) only facilitate ordering, parent-child (PC), ancestor-descendant (AD) and lowest common ancestor (LCA) checking. These operations are sufficient for query processing in ordinary XML data. However, for XML data in extended models, another relationship among document nodes, *mutual exclusion*, must be verified. Unfortunately, the existing labeling schemes cannot serve this operation.

For example, to process a twig pattern query (expressed in XPath) A[//B]//C, for each satisfied A-typed node we not only need to ensure that there is a B-typed node and a C-typed node appearing as its descendants, but also need to ensure that the B-typed node and the C-typed node are not in the same mux distribution if the data is probabilistic, or their time windows have overlapped period if the data is temporal. Similarly, for any keyword query, we not only need to find the LCAs (or extended LCAs) of the query keywords, but also need to check that the nodes matching the query keywords can exist simultaneously.

In this paper, we aim to build a bridge between efficient query processing approaches for ordinary XML data and query processing problems in extended XML models. We take two typical extended XML models, *probabilistic XML* and *temporal XML* under consideration. These two models attract most research attention in extended XML, and they share the common characteristic, i.e., mutual exclusion among nodes. In this paper, we abstract these two models as the extended XML model with mutually exclusive nodes, called XML-ME, and argue that the mutual exclusion among document nodes poses challenges in query processing. The presentation will focus on structured twig pattern query, and we show that the same technique can be easily applied to keyword search. The contribution of this paper can be summarized as:

- We study the underlying difference between ordinary XML model and extended XML models, w.r.t. query processing. We discover that though all kinds of XML data inherit the same syntax and presentation, the different node semantics bring in speciality to data in different models, which also becomes an obstacle preventing most efficient query processing approaches proposed to ordinary XML data being adopted by extended XML models.
- We propose simple extension to the existing XML labeling schemes, so that the mutual exclusion relationship between document nodes can be verified.

We also extend the existing query processing algorithms accordingly, to make them adoptable for extended XML models with mutual exclusion.
– We conduct experiments to validate the performance improvement of our extension.

1.3 Organization

The rest of the paper is organized as following. In Section 2, we revisit some background knowledge and related work. In Section 3, we propose extension to the existing labeling scheme to indicate constraints for mutual exclusion checking. We also discuss how the existing inverted list based algorithms can be extended to process queries over XML-ME data. In Section 4, we conduct experiment to validate our algorithms. Finally, we conclude this paper in Section 5.

2 Background and Related Work

2.1 XML Models

Semi-structured XML data can be naturally modeled as a tree. Each element, attribute and value becomes a tree node. The nesting relationship between element and sub-element, as well as the relationship between element and its attribute, between element and its enclosed text, and between attribute and its value, is modeled as tree edge. By considering the ID reference, some applications also model XML data as graph to facilitate their processing.

Similar to the relational databases, XML model is also extended for different database purposes. Probabilistic XML and temporal XML are two typical extended XML models. They introduce additional elements and/or attributes with special semantics to serve their functions. Although the syntax of the text presentation remains, the introduced functional elements and attributes are normally specially treated in applications. In probabilistic XML models (see the survey [2]), the distributional nodes are specially marked out of ordinary XML nodes, and do not affect the PC relationship across them, i.e., they are not counted as real document nodes. In temporal XML models, the time window can be either modeled as a normal tree node, or modeled as an edge label [17]. However, no matter how the graphic model emerges, the only purpose is to indicate the functional nodes, and facilitate the application processing.

2.2 XML Queries

XML queries are divided into structured queries and unstructured queries. In structured queries, the query language (e.g., XPath and XQuery) needs to specify the structural constraints in order to form a query. It is generally considered that *twig pattern* (small tree pattern) is the core pattern for structured XML queries. Unstructured query, or called keyword query, is simply comprised of a set of keywords. Due to the weak query semantics can be expressed, different from structured queries, unstructured query does not aim to find exact answers, but to search for more relevant and meaningful approximated answers.

Inverted list is an important data structure for both twig pattern query processing and keyword search in ordinary XML data. Generally, an XML document is labeled with a certain labeling scheme. One requirement for the labeling schemes is that the PC, AD and LCA relationship between document nodes must be identifiable based on their labels. Then the labels for each type of document node are organized in an inverted list, in document order. Query processing are performed by scanning only query-relevant inverted list streams, and perform e.g., structural join for twig pattern query processing and LCA computation for keyword search. This approach can significantly reduce I/O cost (as not all nodes are scanned), and proven optimal for computation in some cases [9].

Query processing over extended XML models mainly focus on handling special functions. For example, [8][7] focus on supporting time window to XML queries and [14][17] focus on indexing temporal XML data; while research in probabilistic XML querying focus on linking search result to probabilistic models [11][10][3]. They paid less attention to the performance issues, in comparison with some efficient or even optimal algorithms proposed to ordinary XML data and queries. For example, a recent work on keyword search over probabilistic XML data [12] was proposed, but it focuses more on probability computation and adopts simple document scan approach to find SLCA nodes.

3 Query Processing over XML-ME Documents

It is quite normal that simple XML queries, e.g., twig pattern queries or keyword queries, are issued to those XML data in extended models. Since there are many sophisticated and efficient algorithms to process such queries over ordinary XML documents, we aim to adapt the existing algorithms for XML data in extended models, by keeping the optimality and efficiency of the algorithms. We choose the inverted list based algorithms, as they are proven more efficient than other approaches, e.g., document scan and subsequence matching.

As mentioned earlier, we take two extended XML models that are widely used in modern applications under consideration, i.e., probabilistic XML and temporal XML. Due to the space limitation, we only present algorithms for twig pattern matching. We discuss how the same technique can be easily applied to keyword search.

3.1 Mutually Exclusive Nodes and XML-ME Model

Definition 1. *(**XML instance**) An instance of an XML document* D *is a copy of* D *in which every node is in one of its possible states defined in* D.

Definition 2. *(**Mutually Exclusive Nodes**) In an XML document, two nodes are mutually exclusive if and only if they do not exist simultaneously in any instance of this document.*

For probabilistic XML data, mutually exclusive nodes do not exist simultaneously in any single possible world; for temporal XML data, the time windows for a pair of mutually exclusive nodes do not have overlapping period.

Definition 3. *(**Ordinary XML document**) Ordinary XML document refers to the XML document in which each node has only one state, i.e., there is only one instance.*

Theorem 1. *Ordinary XML documents do not contain mutually exclusive nodes.*

Since an ordinary XML document only has one instance, all nodes can exist simultaneously.

Definition 4. *(**XML-ME model and document**) We name the extended XML model that allows mutual exclusion among elements as XML-ME model. Any document in XML-ME model is called an XML-ME document.*

For example, a probabilistic XML document is an XML-ME document with any two nodes under the same *mux* distributional node mutually exclusive to each other. A temporal XML document is also an XML-ME document, in which two nodes with non-overlapping time period are mutually exclusive to each other. Query processing over XML-ME documents needs to take mutual exclusion checking into account. Any search result with mutually exclusive nodes co-existing is meaningless.

Theorem 2. *In an XML-ME document, two mutually exclusive nodes cannot be in the same root-to-leaf path. In other words, two mutually exclusive nodes cannot be in AD relationship.*

If two nodes u and v are in AD relationship in an XML-ME document, assuming v is u's descendant, then the existence of v implies the existence of u. Thus u and v are not mutually exclusive.

3.2 Mutual Exclusion Checking

In the inverted list based algorithms, labels stored in each inverted list are only sufficient for PC, AD and LCA checking. No matter which labeling scheme is employed, mutual exclusion between document nodes cannot be verified.

The naive approach is to maintain a mutually exclusive map between each pair of document node. Then after the normal processing of a query, the result can be post-processed to filter those answers with mutually exclusive nodes. However, the number the nodes in an XML document can be huge. No matter how we compact the map, it will still introduce a great overhead. As a result, the mutual exclusion checking needs to be done on-the-fly, which is similar to PC, AD and LCA checking.

In order to perform mutual exclusion checking during query processing over XML-ME documents, we need to complement the existing labeling scheme by introducing an additional indicator, namely *mux indicator*, and implement a function to evaluate the mux indicators from any two document nodes.

Example 1. For probabilistic XML data, any two nodes under the same *mux* distributional node are mutually exclusive to each other. We assign an ID to each

mux distributional node and let its child nodes inherit the ID as the mux indicator. Then the checking function can be implemented as that any two document nodes with the same mux indicator are mutually exclusive.

For temporal XML data, the mux indicator can be the time window of each node. The checking function will be that any two document nodes having mux indicators without overlapping period are mutually exclusive.

3.3 Twig Pattern Matching

In this section, we discuss how the existing twig pattern matching algorithms can be extended to handle queries over XML-ME documents. TwigStack [6] was the first algorithm on holistic twig pattern matching, which introduced inverted list streams and stacks to achieve good performance and proven optimal for certain query cases. Later extensions inherit similar ideas and make such a structural join based approach a typical way to process twig pattern queries. In our paper, we take TwigStack as an example to illustrate how such a structural join algorithm can be extended for XML-ME documents. Our extension is adoptable to all other stack-based algorithms.

We invent a new document labeling scheme, which complements the existing schemes by introducing two more components. The new label for each document nodes is a triplet:

(positional label, mux indicator, annotation)

positional label is from the existing labeling scheme, e.g., containment scheme [20] or prefix scheme [18]. *mux indicator* is described in the previous section. The last component *annotation* is used for certain XML-ME documents for which output result needs to be annotated. We will show an example later. For other documents, this annotation field will be empty.

The original TwigStack algorithm has two phases, path matching and path merging. By Theorem 2, because two nodes along the same path will not be mutually exclusive, in the first phase, we do not need to check for mutual exclusion. However, during path matching, we need to combine the mux indicators and annotations from each node in each potential path answer. Then in the second phase, when we merge the path answers, we can easily check for mutual exclusion. We combine the mux indicators and annotations for each path when the path is outputted from stacks. Algorithm 1 shows the procedure. Note that this procedure follows the algorithms proposed in [6], and for simplicity, we assume that the path query contains only AD edges. For more general path query with PC edges, readers can refer to the discussion in [6].

The two introduced functions in line 5 and 6 should be implemented based on different requirements of different documents. We use two examples to illustrate how the functions are implemented for probabilistic XML and temporal XML data.

Example 2. For probabilistic XML data, the mux indicator is simply the ID of the corresponding mux distributional node, if any. The annotation of each node contains the absolute probability of this node appearing in one possible world.

Algorithm 1. outputPath(SN, SP)

Input: the position *SP* of the stack *SN*, which we are interested in
Output: a path answer

```
1: let S[i] be the i-th stack of the query nodes from the root to the current path leaf
2: let index[i] be the position in the i-th stack that we are interested in, where the
   bottom of each stack has position 1
3: index[SN] = SP
4: if SN == 1 then
5:    aggregateMuxIndicator()
6:    aggregateAnnotation()
7:    return path S[n].index[n]-...-S[1].index[1] with combined mux indicator and up-
      dated annotation
8: else
9:    for i = 1 to S[SN].index[SN].pointer_to_parent do
10:      outputPath(SN-1, i)
11:   end for
12: end if
```

This value can be computed by multiplying the probabilities of all probabilistic nodes along the same path from the root to the current node. Suppose in a probabilistic XML document, a node A is a child of a mux node *m1* with probability of 0.5, and *m1* does not have any other distributional node as its ancestor. Then the mux indicator of A is *m1* and its annotation is 0.5. If A has a child B which links with A through another mux node *m2* with probability of 0.8 (note that distributional node does not affect the PC relationship between the two nodes across it), B's mux indicator will be *m2* and its annotation will be 0.5*0.8=0.4. The *aggregateMuxIndicator* function for probabilistic XML is implemented as merging all mux indicators of the nodes along the give path. Then the merged set for path A/B will be {m1, m2}. The *aggregateAnnotation* function returns the annotation of the leaf node along the path, i.e., 0.4, which means the absolute probability of the path appearing in a possible world.

Example 3. For temporal XML data, the mux indicator is the time window of the node, if any. The annotation of a node is not needed. Suppose in a document, a node A has a time window of [2008, 2010] and its child node B has a time window of [2009, 2012], then the *aggregateMuxIndicator* function takes the overlapped period, i.e., [2009, 2010]. This means only within this period, the two nodes (i.e., the path A/B) can exist together.

In the next step, we present how the aggregated mux indicator and annotation of each path answer can be used to return final answers.

Approach 1. In the first approach, we check for the aggregated mux indicator of each path answer during path merging. Also, we update the final annotation for the whole twig answers. The algorithm is presented in Algorithm 2.

Similarly, the function satisfy(I_p, I_q) and update(A_p, A_q) should be implemented for ad hoc XML-ME documents.

Algorithm 2. pathMerge

Input: two sets of path answers P and Q to be merged
Output: merged result, possibly with annotation

```
1: suppose the n-th node the branching node to merge two paths from P and Q
2: for all pair of paths p and q from P and Q respectively do
3:    let p[i] and q[i] be the i-th nodes in path p and q
4:    if p[n] and q[n] have the same positional label then
5:       let I_p and I_q be the aggregated mux indicator, A_p and A_q be the aggregated
         annotation in p and q respectively
6:       if satisfy(I_p, I_q) then
7:          update(A_p, A_q)
8:          return merged result with updated annotation
9:       end if
10:   end if
11: end for
```

Example 4. For probabilistic XML data, the satisfy() function over two aggregated mux indicators returns true if the two mux indicators do not have any common element. In other words, the two paths do not have any pair of nodes that are under the same mux distributional node. Then the update() function updates the overall annotation, i.e., the probability of the merge result, by multiplying the probabilities of the two paths and dividing it by the probability of their common prefix path.

For temporal XML data, the satisfy() function returns true if the two aggregated mux indicators have overlapped time period, i.e., the two paths can existing simultaneously.

Although the proposed extension works on top of any optimal twig pattern matching algorithms (e.g., [6] for twig queries with only AD edges and [13] for twig queries with AD edges, as well as PC edges under non-branching nodes), it cannot guarantee any optimality. This is because the optimal twig matching algorithms only consider structural join, but no mux indicator. To achieve optimality for twig pattern matching, we need to push mux indicator checking down to the getNext() function, which is the core function to ensure optimality in those twig pattern matching algorithms over ordinary XML data.

Approach 2. In the second approach, we aim to maintain the optimality of the existing algorithms when they process twig pattern queries over XML-ME data. The idea in the existing algorithms to achieve optimality for matching a certain class of twig pattern queries is to introduce the getNext() function, which guarantees that when a node is pushed onto the stack its corresponding query node must have a subtwig matching solution.

In the existing algorithms, to ensure a query node has a subtwig matching solution, they only check with AD (and also PC in some extended work [13]) relationship. In our situation, in addition, we also need to check the mutual exclusion relationship among all nodes matching a certain subtwig. Thus, in

Algorithm 3. getNext(q)

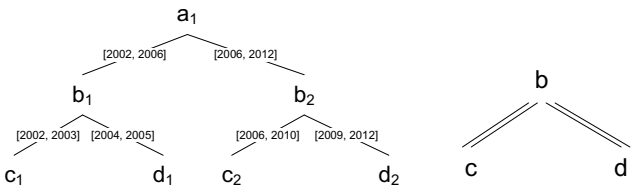
```
1: if isLeaf(q) then
2:    return (q, next(T_q).muxIndicator)
3: end if
4: let mux[i] store the aggregated mux indicator for the subtwig rooted at i-the child
   of q
5: for q_i in children(q) do
6:    (n_i, mux_i) = getNext(q_i)
7:    mux[i] = mux_i
8:    if n_i != q_i then
9:       return (n_i, mux_i)
10:   end if
11: end for
12: n_min = minarg_{n_i} nextL(T_{n_i})
13: n_max = maxarg_{n_i} nextL(T_{n_i})
14: while nextR(T_q) < nextL(T_{n_max}) do
15:   advance(T_q)
16: end while
17: if nextL(T_q) < nextL(T_{n_min}) ∧ !checkMux(mux[]) then
18:   return (q, aggregatedMuxIndicator(mux[]))
19: else
20:   return (n_min, mux[min])
21: end if
```

the extended getNext() function, we aggregate the mux indicators of the nodes that potentially form a subtwig match (*aggregatedMuxIndicator()*), and include a function to check whether the mux indicators of a set of nodes allow them to appear in the same answer (*checkMux()*). The pseudo-code for the extended getNext() function is presented in Algorithm 3. Similarly, we continue with the getNext() function proposed in [6], e.g., using the containment scheme for positional labeling and checking the position of two nodes based on their labels, etc.

Example 5. Consider a temporal XML data fragment in Fig. 2(a). The label of each edge indicates the time period of the corresponding subtree. Suppose we have a twig pattern query to process, as shown in Fig. 2(b). Using Approach 1, during the first phase of pattern matching, paths b_1-c_1, b_1-d_1, b_2-c_2 and b_2-d_2 are returned. Then in the second phase, these paths are merged into twig answer, during which, the aggregated mux indicators (i.e., time period) of each path are checked. Finally, only b_2-c_2 and b_2-d_2 are returned and form a twig answer. Using Approach 2, during the first phase, the getNext() function will not return node b_1 (and all paths with b_1), as it does not have any subtree matching the query twig and satisfying mutual exclusion constraint.

Theorem 3. *Algorithm 3 complements structural join algorithms that use getNext() function, and maintains the optimality of these algorithms.*

Algorithm 3 modifies the original getNext() function by adding the constraint to check for mutually exclusive nodes, in order to control the node pushed onto



(a) Temporal XML data (b) Query

Fig. 2. Example temporal XML data and twig pattern query

stacks. In other words, any node returned by the new function must contain sub-matches in which there is no mutually exclusive nodes. Also any node containing mutually exclusive nodes in its sub-matching will not be returned. Thus, the optimality of the original algorithm can be maintained. For example, in TwigStack, by considering mutual exclusion relationship between nodes, the algorithm is still optimal for queries with AD edges only, by using our extended getNext() function.

3.4 Keyword Search

In XML keyword search, similarly, the query-relevant inverted list streams are sequentially scanned, and LCA or extended LCA (e.g., SLCA [19], etc.) of a group of nodes, each of which comes from a stream, is computed. To handle XML-ME documents, we still follow the same labeling scheme as described in the previous section, and use it to check for mutual exclusion relationship among a group of nodes, to decide whether their LCA is needed to be computed. If the mux indicators of a group of nodes cannot pass the corresponding checking function, this group can be ignored, without computing LCA.

Example 6. Consider a keyword query {c, d} issued to the temporal XML data in Fig. 2(a). Without considering mutual exclusion, the LCAs for both pairs (c_1, d_1) and (c_2, d_2) need to be computed. With our labeling scheme, we can check that c_1 and d_1 are mutually exclusive. Thus, we can skip computing LCA for (c_1, d_1).

Since our extension for mutual exclusion checking is orthogonal to LCA computation, it inherits the advantages of any existing efficient LCA-based algorithms, e.g., skipping nodes along an inverted list stream. We do not further discuss LCA computation with mutual exclusion, as technically it is the same as twig pattern matching.

4 Experiment

In our experimental evaluation, we compare three approaches to process twig pattern queries over XML-ME data. To be fair, we use the same algorithm, TwigStack, as the fundamental structural join algorithm for all the three approaches. In the first approach, we simply consider the document as ordinary

XML document and perform structural join to find query matches. Then we post-process the answers to filter those with mutually exclusive nodes. We call this approach *Post-filtering*. The second approach follows the algorithm proposed in Section 3.3, in which we filter the paths that are exclusive to other paths during path merging. We call this approach *Merge-filtering*. In the last approach, we implement the algorithm in Section 3.3. We call it *TwigStackX*.

4.1 Settings

We use the XMark [1] benchmark XML data (the standard 110MB document downloaded from [1]) for our experiment. We randomize the mutually exclusive nodes in the document to make it an XML-ME document. In particular, we randomly assign an mux indicator to a portion of document nodes. Two or more nodes with the same mux indicator are considered exclusive to each other. We control two factors, *muxDegree* and *contraDegree*. *muxDegree* is the percentage of document nodes to be assign mux indicator, while *contraDegree* controls the probability that two nodes have the same mux indicator, under the condition that both of them have mux indicators.

We randomly compose five queries for the evaluation. The queries are shown in Table 1. Note that our purpose is to evaluate the proposed algorithms to handle XML-ME data. We vary the muxDegree and contraDegree of the document to show the efficiency of the algorithms. The variety of the queries is less important.

Table 1. Queries used in experiments

Query	XPath Expression
Q1	//item[location=“United State”]/id
Q2	//person[//profile/age=“40”]/name
Q3	//open_auction[//type=“Featured”]//initial
Q4	//regions/africa/item[//mailbox//mail/from]//keyword
Q5	//closed_auction[seller][buyer]/price

We set *muxDegree* to be in {0.2, 0.4, 0.6} and *contraDegree* to be in {1/3, 1/4}. We obey Theorem 2, i.e., there is no pair of nodes along the same path having the same mux indicator.

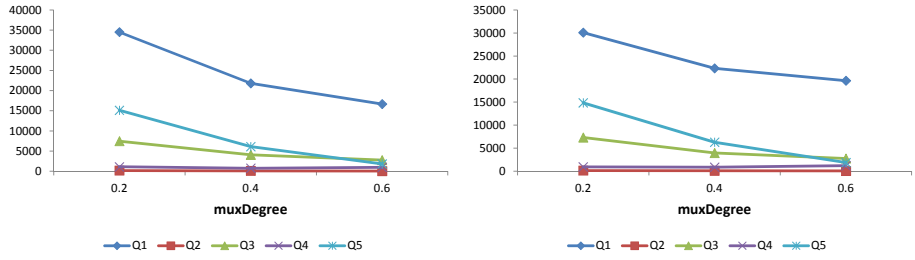
4.2 Intermediate Path Set

In this section, we monitor the intermediate paths returned by each algorithm. The detailed result is shown in Table 2. For *Post-filtering* and *Merge-filtering*, no matter how muxDegree and contraDegree changes, the intermediate path set does not change. The reason is that both of the two algorithms will generate intermediate paths as that in ordinary XML data, and then check for mutual exclusion. In particular, *Post-filtering* checks on twig answers, while *Merge-filtering* checks during path merging. This also explains why they have the same intermediate path result.

Table 2. Intermediate path size evaluation

Query	Post-filtering	Merge-filtering	TwigStackX (muxDegree-ContraDegree)					
			0.2-1/3	0.4-1/3	0.6-1/3	0.2-1/4	0.4-1/4	0.6-1/4
Q1	38044	38044	34512	21800	16676	30091	22333	19651
Q2	270	270	186	112	42	160	122	70
Q3	11434	11434	7470	4086	2762	7328	3960	2780
Q4	1257	1257	1114	753	975	956	877	1210
Q5	29250	29250	15111	6129	1818	14850	6300	1893

For TwigStackX, the intermediate path result set is smaller. Furthermore, when the muxDegree increases, the intermediate set decreases, as better viewed in Fig. 3. This is easy to understand. As more nodes have mux indicators, there will be more mutual exclusion cases exist. Then the getNext() function will filter more nodes before pushing nodes onto stacks. One interesting case is that for Q4, this rule does not hold. When muxDegree increases from 0.4 to 0.6, the intermediate path set for Q4 also increases. The reason is that Q4 is quite deep along a path, and the branching node is at the low position. Based on Theorem 2, when the muxDegree is high, most mux indicators are assigned to the common part of the two paths for Q4. Then there will be less path filtered during pattern matching.



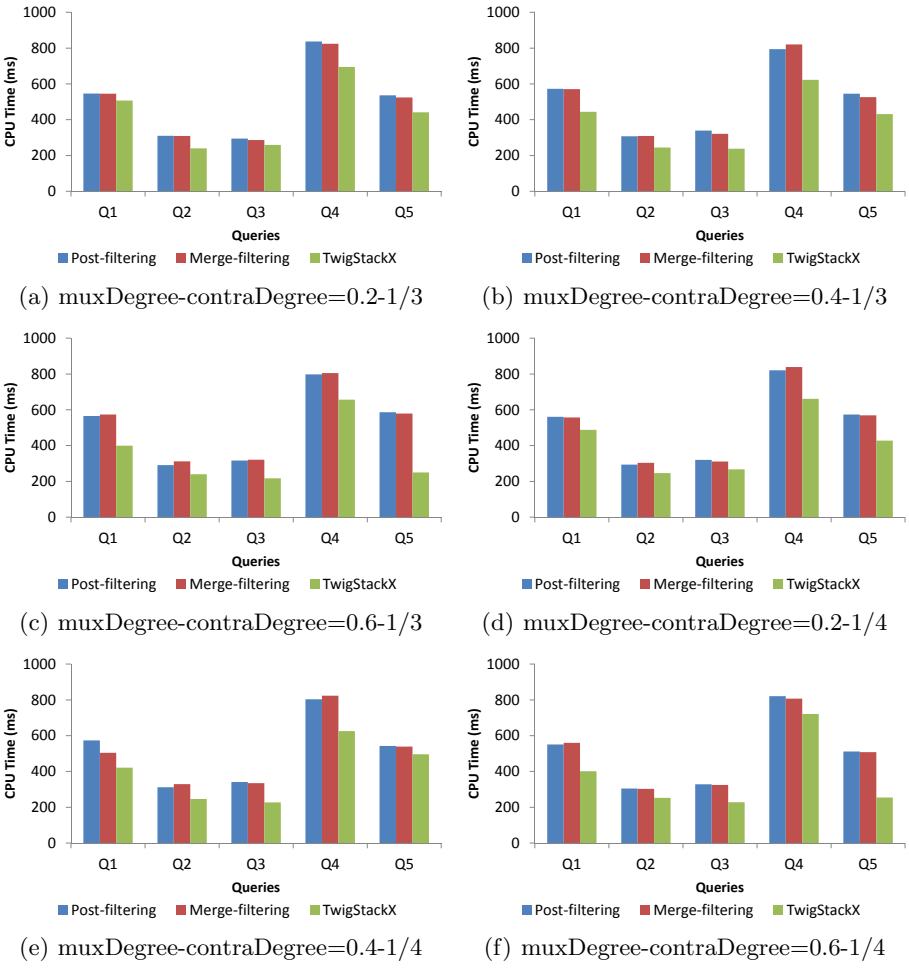
(a) contraDegree=1/3 (b) contraDegree=1/4

Fig. 3. Intermediate path size changes for TwigStackX

4.3 Performance

In this section, we evaluate the performance of the three approaches. As we set three values to muxDegree and two values to contraDegree, we show the performance comparison in all six cases. The results are shown in Fig. 4.

From the figure we can see that the performance for Post-filtering and Merge-filtering are quite similar, though they filter the mutually exclusive nodes at different stages of query processing. TwigStackX is more efficient than the first two approaches. Also, by comparing Fig. 4(a) to 4(c) and Fig. 4(d) to 4(f), for each query, the processing time is almost proportional to the muxDegree, i.e., the intermediate path size (except for Q4). We can conclude that the size of intermediate path is the main factor affecting query processing performance.



(a) muxDegree-contraDegree=0.2-1/3

(b) muxDegree-contraDegree=0.4-1/3

(c) muxDegree-contraDegree=0.6-1/3

(d) muxDegree-contraDegree=0.2-1/4

(e) muxDegree-contraDegree=0.4-1/4

(f) muxDegree-contraDegree=0.6-1/4

Fig. 4. Performance comparison in six cases

5 Conclusion

In this paper, we investigate the problem of query processing over semi-structured XML data with mutually exclusive nodes, e.g., probabilistic XML data and temporal XML data. We abstract such extended XML model as XML-ME model. We show that the existing XML query processing techniques are insufficient and not extendable to handle queries over XML-ME data. Motivated by this, we invent a new labeling scheme for XML-ME data, and devise and compare query processing algorithms to solve this problem. We focus on structured twig pattern query, and illustrate that the same technique can be applied to keyword query processing.

For future work, we plan to look at ad hoc queries (e.g., query with time constraints) to the XML-ME data. We will study how the performance issues in, e.g., twig pattern matching affect the ad hoc query processing.

References

1. http://www.xml-benchmark.org/
2. Abiteboul, S., Kimelfeld, B., Sagiv, Y., Senellart, P.: On the expressiveness of probabilistic XML models. VLDB J. 18(5), 1041–1064 (2009)
3. Abiteboul, S., Senellart, P.: Querying and updating probabilistic information in XML. 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. 1059–1068. Springer, Heidelberg (2006)
4. Agrawal, P., Benjelloun, O., Sarma, A.D., Hayworth, C., Nabar, S., Sugihara, T., Widom, J.: Trio: A system for data, uncertainty, and lineage. In: VLDB, pp. 1151–1154 (2006)
5. Antova, L., Jansen, T., Koch, C., Olteanu, D.: Fast and simple relational processing of uncertain data. In: ICDE, pp. 983–992 (2008)
6. Bruno, N., Koudas, N., Srivastava, D.: Holistic twig joins: optimal XML pattern matching. In: SIGMOD, pp. 310–321 (2002)
7. Combi, C., Lavarini, N., Oliboni, B.: Querying semistructured temporal data. In: Grust, T., Höpfner, H., Illarramendi, A., Jablonski, S., Fischer, F., Müller, S., Patranjan, P.-L., Sattler, K.-U., Spiliopoulou, M., Wijsen, J. (eds.) EDBT 2006. LNCS, vol. 4254, pp. 625–636. Springer, Heidelberg (2006)
8. Gao, D., Snodgrass, R.T.: Temporal slicing in the evaluation of XML queries. In: VLDB, pp. 632–643 (2003)
9. Gou, G., Chirkova, R.: Efficiently querying large XML data repositories: a survey. IEEE Trans. Knowl. Data Eng. 19(10), 1381–1403 (2007)
10. Kimelfeld, B., Kosharovsky, Y., SagivMatching, Y.: Query efficiency in probabilistic XML models. In: SIGMOD, pp. 701–714 (2008)
11. Kimelfeld, B., SagivMatching, Y.: Matching twigs in probabilistic XML. In: VLDB, pp. 27–38 (2007)
12. Li, J., Liu, C., Zhou, R., Wang, W.: Top-k keyword search over probabilistic XML data. In: ICDE, pp. 673–684 (2011)
13. Lu, J., Chen, T., Ling, T.W.: Efficient processing of XML twig patterns with parent child edges: a look-ahead approach. In: CIKM, pp. 533–542 (2004)
14. Mendelzon, A.O., Rizzolo, F., Vaisman, A.: Indexing temporal XML documents. In: VLDB, pp. 216–227 (2004)
15. Nierman, A., Jagadish, H.V.: ProTDB: Probabilistic Data in XML. In: VLDB, pp. 646–657 (2002)
16. Nicola, M.: Temporal data management with IBM DB2 Time Travel Query. IBM Software White Paper
17. Rizzolo, F., Vaisman, A.A.: Temporal XML: modeling, indexing, and query processing. VLDB J. 17(5), 1179–1212 (2008)
18. Tatarinov, I., Viglas, S., Beyer, K.S., Shanmugasundaram, J., Shekita, E.J., Zhang, C.: Storing and querying ordered XML using a relational database system. In: SIGMOD, pp. 204–215 (2002)
19. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest LCAs in XML databases. In: SIGMOD, pp. 527–538 (2005)
20. Zhang, C., Naughton, J.F., DeWitt, D.J., Luo, Q., Lohman, G.M.: On supporting containment queries in relational database management systems. In: SIGMOD, pp. 425–436 (2001)

XReason: A Semantic Approach That Reasons with Patterns to Answer XML Keyword Queries

Cem Aksoy[1], Aggeliki Dimitriou[2], Dimitri Theodoratos[1], and Xiaoying Wu[3]

[1] New Jersey Institute of Technology, Newark, NJ, USA
[2] National Technical University of Athens, Athens, Greece
[3] Wuhan University, Wuhan, China

Abstract. Keyword search is a popular technique which allows querying multiple data sources on the web without having full knowledge of their structure. This flexibility comes with a drawback: usually, even though a large number of results match the user's request only few of them are relevant to her intent. Since data on the web are often in tree-structured form, several approaches have been suggested in the past which attempt to exploit the structural properties of the data in order to filter out irrelevant results and return meaningful answers. This is certainly a difficult task, and depending on the type of dataset, these approaches show low precision and/or recall.

In this paper, we introduce an original approach for answering keyword queries called XReason. XReason identifies structural patterns in the keyword matches and reasons with them in order to return meaningful results and to rank them with respect to their relevance. Our semantics shows a non-monotonic behavior and in the presence of additional patterns, it is able to better converge to the users intent. We design an efficient stack-based algorithm for evaluating keyword queries on tree structured data, and we run experiments to evaluate its efficiency and the effectiveness of our semantics as a filtering and ranking system. Our results show that our approach shows better performance than the other approaches in many cases of real and benchmark datasets.

1 Introduction

Keyword search on tree-structured data has attracted a lot of attention in recent years. One of the reasons is that tree-structured data (e.g., XML, JSON) has been established as the standard format for exporting and exchanging data on the web. Another reason is that keyword search is by far the most popular technique for searching the web. Its popularity is due to the fact that the users do not need to know a complex query language (like XQuery) in order to retrieve information from the web. In addition, they can issue queries without having full or even partial knowledge of the schema (structure) and the same query can be issued against multiple, differently structured data sources on the web.

The candidate results of a keyword query on an XML tree are defined usually as the minimum connecting trees (MCTs) in the XML tree that contain an

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 299–314, 2013.

instance of all the keywords. The roots of the MCTs are the lowest common ancestors (LCA) of the included keyword instances and are often used to identify the candidate results. A drawback of keyword search is that keyword queries usually return a very large number of results most of which are irrelevant to the user's intent. Many recent works focus on addressing this problem by appropriately assigning semantics to keyword queries on tree data. A number of these semantics, characterized as *filtering*, aim at filtering out a subset of the candidate LCAs that are irrelevant. Some filtering semantics prune LCAs based exclusively on *structural* information (e.g., Smallest LCA [7, 14, 19] or Exclusive LCA [6, 20] semantics) while others take also into account *semantic* information, that is, the labels of the nodes in the XML tree (e.g., Valuable LCA [4, 8] or Meaningful LCA [11] semantics). Other recent works assign *ranking* semantics to keyword queries, that is, they rank the results aiming at placing on top those that are more relevant [1, 3, 4, 6, 12, 15, 17]. Ranking the results improves the usability of the system. In order to perform the ranking these works exploit: (a) structural characteristics of the results, and/or (b) statistical information or information theory metrics adapted to the tree structure of the data. All the ranking approaches rank the results based on some scoring function which assigns scores to the results.

Although filtering approaches are intuitively reasonable for specific cases of data, they are ad-hoc and they are frequently violated in practice resulting in low precision and/or recall [17]. This weakness is due to the fact that these semantics depend on the local properties of the LCAs in the XML tree. For instance, the Exclusive LCA semantics filters out candidate LCAs whose keyword instances are descendants of other descendant candidate LCAs, while the Smallest LCA semantics prunes candidate LCAs that are ancestors of other candidate LCAs. Most ranking approaches are combined with filtering approaches, that is, they rank only the LCAs accepted by the respective filtering semantics, this way inheriting the low recall of the filtering semantics.

Contribution. In this paper, we claim that a meaningful semantics for keyword queries should not depend on the local properties of the LCAs in the XML tree but on the patterns the keyword instances define on the XML tree. Further, a ranking for the results should not be obtained based merely on scores assigned through a scoring function, but by directly comparing the structural and semantic properties of the patterns, that is, by reasoning on them. In the present work this comparison is realized based on homomorphisms between patterns.

The main contributions of this paper are the following:

- We define the concept of pattern of a keyword query on an XML tree. A pattern is a tree that records the structural relationships between node labels in the tree defined by an instance of the query. Every pattern represents the set of query instances in the XML tree that comply with this pattern. We introduce homomorphisms between patterns and we use them to define two relations *all_path_homomorphism (aph)* and *partial_path_homomorphism (pph)* on patterns (Sections 3.1 and 3.2).

- Based on the *aph* and *pph* relations, we organize the patterns of a query into a graph of patterns which we leverage to define filtering and ranking semantics for queries. We name this approach and the semantics *XReason*. When the size of the data increases, new patterns may be detected and the reasoning process might retain them as more relevant to the query. Since these patterns might represent fewer query instances, *XReason* has a non-monotonic behavior (Section 3.3).
- We design an efficient stack-based algorithm for computing query answers that implements *XReason*. Contrary to the previous ranking algorithms that rely on auxiliary structures and require a preprocessing of the datasets, our algorithm uses only the inverted lists of the keywords in order to compute and rank the query answers (Section 4).
- We run experiments on real and benchmark datasets and we compared with previous approaches in order to assess the effectiveness of *XReason* and the efficiency of our algorithm. Our results show that *XReason* filtering and ranking semantics outperform previous approaches with respect to various metrics and our algorithm is fast and scales well when the number of query results increases (Section 5).

2 Definitions

Data Model. As is usual, we view XML documents as ordered node labeled trees. Nodes represent elements and attributes. Edges represent element to element and element to attribute relationships. A function *value* is defined on every node and returns the content of the element or the value of the attribute represented by the node. Since we want to allow keywords to match also element and attribute names, we assume that function *value* returns also the label of a node on which it is applied. If the value $value(n)$ of a node n includes a keyword k we say that n *contains* keyword k and that node n is an *instance* of k. XML tree nodes are enumerated using the Dewey encoding scheme.

Fig. 1 shows an XML tree. Dewey codes are omitted for clarity. Plain numbers are used instead to identify the nodes.

Keyword Queries. A *(keyword) query* Q is a set of keywords $\{k_1, k_2, \ldots, k_n\}$. Keyword queries are embedded to XML trees.

Definition 1. *Let Q be a query and T be an XML tree. An* instance *of Q on T is a function from Q to the nodes of T that maps every keyword k in Q to an instance of k in T.*

We use the term "query instance" to refer to the function that maps the query keywords to the tree nodes and also, to the set of the nodes in the tree which are the images of the query keywords under this function. Note that two query keywords can be mapped to the same tree node.

Definition 2. *Let Q be a query, T be an XML tree, and I be an instance of Q on T. The* instance tree (IT) *of I is the minimum subtree S of T such that: (a)*

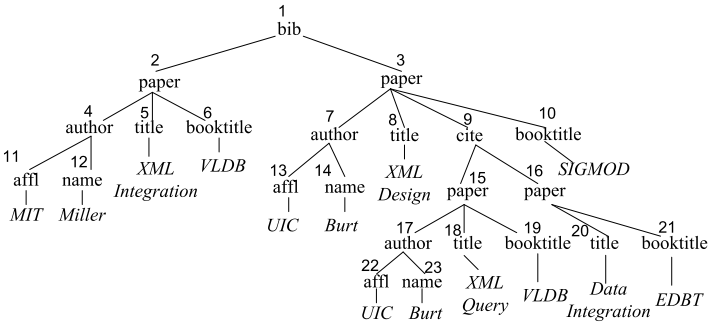


Fig. 1. An XML tree T

S is rooted at the root of T and comprises all the nodes of I, and (b) every node n in S is annotated by the keywords which are mapped by I to n. The annotation of node n is denoted as $ann(n)$. The minimum connected tree (MCT) *of I is the minimum subtree of S that comprises the nodes of I.*

Clearly, the root of the MCT is the *Lowest Common Ancestor* (LCA) of the nodes of I in T.

Consider the XML tree of Fig. 1 borrowed from [17] and the keyword query $Q = \{XML, Integration, VLDB\}$. Figures 2(a) and (b) show the IT and the MCT, respectively, of the instance $\{(XML, 18), (VLDB, 19), (Integration, 20)\}$ of Q on T. Non-empty annotations are shown between square brackets by the nodes.

Given a keyword query Q and an XML tree T, the set $\mathcal{C}$ of the ITs of all the instances of Q on T is the set of the candidate results of Q on T. The answer of a keyword query Q on an XML tree T is a *subset* of $\mathcal{C}$.

3 Query Semantics

In order to define semantics for queries we introduce patterns of ITs and homomorphisms between patterns and study their properties.

3.1 IT Patterns and Pattern Homomorphisms

Definition 3 (IT pattern). *A* pattern *P of a query Q on an XML tree T is a tree which is isomorphic (including the annotations) to an IT of Q on T.*

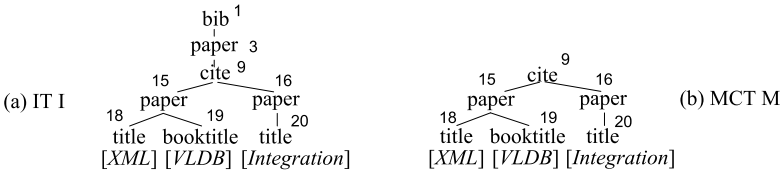


Fig. 2. (a) An IT and (b) its MCT

The MCT of a pattern P refers to P without the path that links the LCA of the nodes with non-empty annotations to the root of P. Function $size(P)$ returns the number of edges of P.

Multiple ITs of Q on T can share the same pattern. Fig. 3 shows eight patterns (out of 12 in total) of the keyword query $Q = \{XML, Integration, VLDB\}$ on the XML tree T of Fig. 1. The size of pattern P_3 is 3 and that of its MCT is 2. The size of pattern P_6 is 7 and that of its MCT is 6.

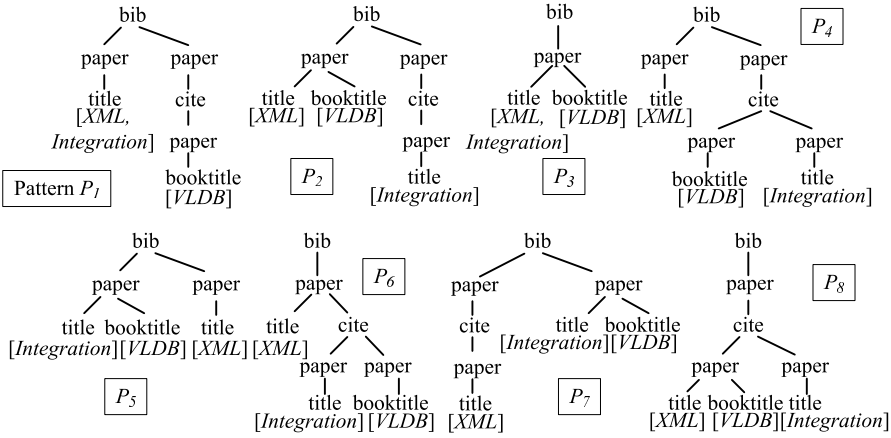


Fig. 3. Some patterns for $Q = \{XML, Integration, VLDB\}$ on the tree of Fig.1

Definition 4 (Pattern homomorphism). *Let S and S' be two subtrees of patterns of a query on an XML tree. A* homomorphism *from S to S' is a function h from the nodes of S to the nodes of S' such that:*
(a) for every node n in S, n and $h(n)$ have the same labels.
(b) if n_2 is a child of n_1 in S, $h(n_2)$ is a child of $h(n_1)$ in S', and
(c) for every node n in S, $ann(n) \subseteq ann(h(n))$.

Fig. 4 shows the MCTs M, M', and M'' of three patterns of the query $Q = \{Informatics, John, Smith\}$ on an XML tree. As we can see in this figure there are homomorphisms from M'' to M' and from M' to M but not from M to M' or from M' to M''.

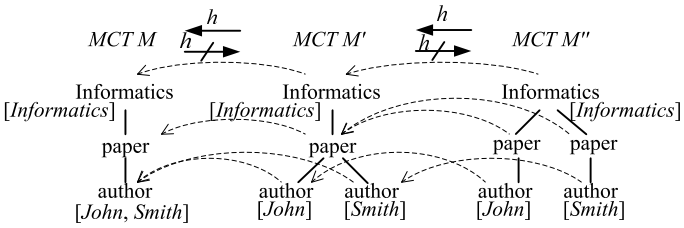


Fig. 4. Pattern MCTs M, M' and M'' and homomorphisms between them

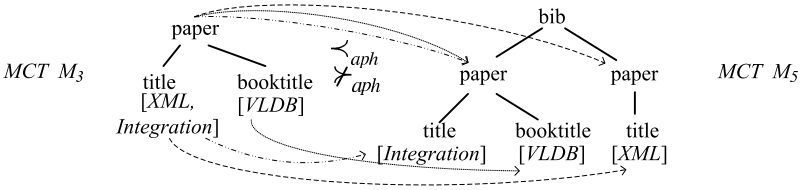


Fig. 5. Pattern MCTs M_3 and M_5 and a path homomorphism from M_3 to M_5

3.2 Path Homomorphisms and PH Relations

Definition 5 (Path homomorphism). *Let S and S' be two subtrees of patterns of a query on an XML tree. We say that there is a* path homomorphism *from S to S' if for every path p from the root of S to a node n in S annotated by a keyword k (every path is considered separately), there is a function ph from the nodes of p to the nodes of a path p' of S' such that:*
(a) for every node n_1 in p, n_1 and $ph(n_1)$ have the same labels.
(b) if n_2 is a child of n_1 in p, $ph(n_2)$ is a child of $ph(n_1)$ in p', and
(c) $k \in ann(ph(n)) \cup label(ph(n))$.

Fig. 5 shows the MCTs M_3 and M_5 of the corresponding patterns (shown in Fig. 3) for the query $\{XML, Integration, VLDB\}$ on the XML tree of Fig. 1. There is a path homomorphism from M_3 to M_5 (but not vice versa). The different types of dashed lines indicate the different mappings of the paths of M_3 to paths of M_5 according to this path homomorphism.

In Fig. 4, one can see that there are path homomorphisms between any two pattern MCTs M, M' and M''. This is expected due to the following proposition.

Proposition 1. *Let M and M' be two pattern MCTs of a query on an XML tree. If there is a homomorphism from M to M', there is also a path homomorphism from M to M'. The opposite is not necessarily true. Further, if there is a homomorphism from M to M', there is also a path homomorphism from M' to M.*

We now use the concept of homomorphism to define a relation named *all_path_homomorphism* (denoted $\prec_{aph}$), on patterns.

Definition 6 ($\prec_{aph}$ relation). *Let P and P' be two patterns of a query Q on an XML tree T. $P \prec_{aph} P'$ iff one of the following conditions holds:*
(a) There is a homomorphism from the MCT of P' to the MCT of P but not vice versa.
(b) There is a path homomorphism from the MCT of P to the MCT of P' but not vice versa.

As an example, observe that for the pattern MCTs M_3 and M_5 of Fig. 5, $P_3 \prec_{aph} P_5$ by virtue of condition (b). For the patterns P, P' and P'' whose MCTs are shown in Fig. 4, $P \prec_{aph} P' \prec_{aph} P''$ by virtue of condition (a). We now show a property of relation $\prec_{aph}$.

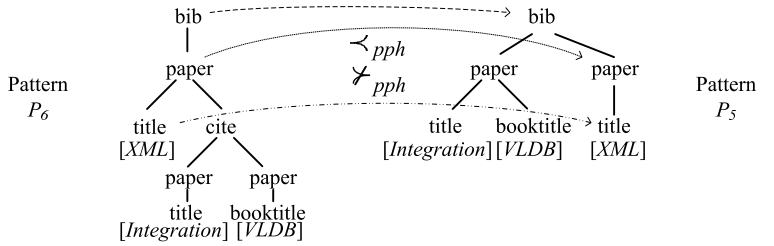


Fig. 6. A path homomorphism from a path of pattern P_6 to a path of P_5

Proposition 2. *The relation $\prec_{aph}$ on the set of patterns of a query on an XML tree is a strict partial order.*

In order to avoid returning results that come from different unrelated parts of an XML tree, we define below a relation, named *partial_path_homomorphism*, on query patterns.

Definition 7 ($\prec_{pph}$ relation). *Let P and P' be two patterns of a query Q on an XML tree T, and p be a path from the root of P to an annotated node of P. $P \prec_{pph} P'$ iff there is a path homomorphism ph of p to a path in P' such that:*
(a) the root of P is mapped by ph to the root of P'.
(b) the root of the MCT of P is mapped by ph to a node which is a descendant (not self) of the root of the MCT of P', and
(c) $P' \not\prec_{aph} P$.

Fig. 6 shows the patterns P_5 and P_6 of the query $\{XML, Integration, VLDB\}$ on the XML tree of Fig. 1. As shown in the figure, there is a path homomorphism from the path `bib/paper/title[XML]` of P_6 to the same path of P_5 and the image of root of the MCT of P_6 (`paper`) under this homomorphism is a descendant of the root of the MCT of P_5 (`bib`). Therefore, $P_6 \prec_{pph} P_5$. It is not difficult to see that the relation $\prec_{pph}$ is acyclic.

3.3 *XReason* Semantics

We use path homomorphisms to define filtering and ranking semantics to keyword queries called *XReason* semantics. We first define a precedence relation, $\prec$, on patterns.

Definition 8. *Let P and P' be two patterns of a query Q in an XML tree T. $P \prec P'$ iff $P \prec_{pph} P'$ or $P \prec_{aph} P'$.*

Based on the previous discussion one can see that the following property holds for the precedence relation on patterns.

Proposition 3. *The relation $\prec$ on the set of patterns of a query on an XML tree is acyclic.*

Given the relation $\prec$ on the set of patterns of a query Q on an XML tree T, consider a directed graph $G_{\prec}$ such that: (a) the nodes of $G_{\prec}$ are the patterns of Q on T, and (b) there is an edge in $G_{\prec}$ from node P_1 to node P_2 iff $P_1 \prec P_2$. Clearly, because of Proposition 3, $G_{\prec}$ is *acyclic*. Fig. 7(a) shows the graph $G_{\prec}$ for the relation $\prec$ on the set of patterns of query $Q = \{XML, Integration, VLDB\}$ on the XML tree of Fig. 1. There are 12 such patterns and eight of them (patterns P_1 - P_8) are shown in detail in Fig. 3. The edges are labeled by letters a and/or p to indicate which of the relations $\prec_{aph}$ and $\prec_{pph}$ relate its nodes. Transitive a-edges which are not p-edges are omitted to reduce the clutter. In general, a graph $G_{\prec}$ can have multiple source nodes (i.e., nodes without incoming edges). The one of Fig. 7 has only one source node (pattern P_3).

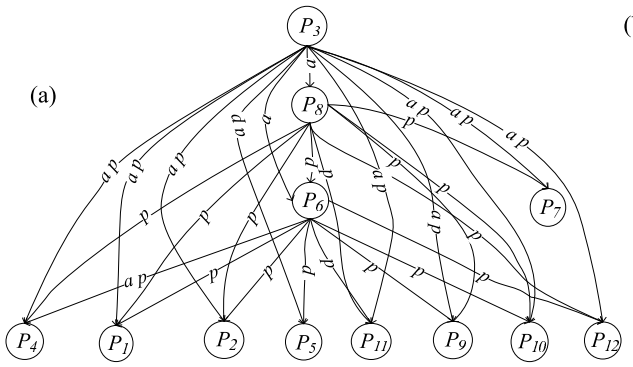


Fig. 7. (a) The graph $G_{\prec}$, (b) Pattern order $\mathcal{O}$

Definition 9 (Filtering *XReason* semantics). *According to the filtering* XReason *semantics the answer of Q on T is the set of ITs of Q on T whose patterns are source nodes in the $G_{\prec}$ graph. These ITs are called* results *of Q on T.*

Based on the previous definition the answer of query $Q = \{XML, Integration, VLDB\}$ on the XML tree of Fig. 1 is the set of ITs which share pattern P_3 (the only source node in graph $G_{\prec}$ of Fig. 7(a)). There is only one such IT which corresponds to the subtree rooted at `bib` whose leaves are nodes 5 and 6 in Fig. 1.

In the graph of Fig. 7(a), observe that all the nodes (patterns) can be partitioned in levels based on their maximum distance ($GLevel$) from a source node. For instance, in level 3 there are patterns P_6 and P_7. The patterns in one level can also be further distinguished based on the depth of their MCT root in the pattern ($MCTDepth$) and the size of their MCT ($MCTSize$). We create an order $\mathcal{O}$ for the patterns in $G_{\prec}$ which ranks them in: (a) ascending order of $GLevel$, (b) descending order of $MCTDepth$, and (c) ascending order of $MCTSize$. Note that two patterns might be placed at the same level in $\mathcal{O}$. Order $\mathcal{O}$ does not distinguish between these patterns. In our running example, one can see that the twelve patterns of Fig. 7(a) are ordered with respect to $\mathcal{O}$ as shown in Fig. 7(b).

Definition 10 (Ranking *XReason* semantics). *According to the ranking XReason semantics the answer of a query Q on an XML tree T is the list of ITs of Q on T ranked in an order which complies with the order $\mathcal{O}$ of their patterns.*

4 The Algorithm

In order to implement *XReason* we designed a stack-based algorithm named *PatternStack* which is outlined in Algorithm 1. PatternStack takes as input the keyword query and the inverted lists of the nodes (Dewey codes) for these keywords in the dataset. It returns the patterns of the query answers associated with their ITs. PatternStack processes nodes from the keyword inverted lists in document order. Each stack entry corresponds to a node and is associated with a set of patterns. In order for a node n to be pushed into a stack, the top stack node should be the parent of n (lines 4-7). This is guaranteed by appropriate pops of non-ancestor nodes and pushes of all the ancestors of n. For each new node, a partial pattern MCT (i.e., an MCT of a subset of the query) is constructed (line 21). Then, it is combined with all existing patterns of the top stack node and the resulting pattern set is unioned with that of the top stack node (lines 22-23). During a pop action, all complete patterns are removed from the top stack entry (lines 13-14). The remaining patterns are extended with an edge from their root to the parent of the top node (line 15-16). The top entry is popped (line 17) and its extended patterns are combined with the patterns

Algorithm 1: PatternStack algorithm

```
1  PatternStack(k_1, ..., k_n: keyword query, invL: inverted lists)
2      s = new Stack()
3      while currentNode = getNextNodeFromInvertedLists() do
4          while s.topNode is not ancestor of currentNode do
5              pop(s)
6          while s.topNode is not parent of currentNode do
7              push(s, ancestor of currentNode at s.topNode.depth+1, "")
8          push(s, currentNode, keyword)
9      while s is not empty do
10         pop(s)
11 pop(Stack s)
12     for temp = s.top.patterns.next() do
13         if temp is complete then
14             s.top.removePattern(temp)
15         else
16             childPatterns.add(extendToParent(temp))
17     s.pop()
18     newPatterns = produceNewPatterns(s.top.patterns, childPatterns)
19     add(s.top, newPatterns)
20 push(Stack s, Node n, String keyword)
21     newPatternId = partialPattern(n.labelId,keyword)
22     newPatternMCTs = produceNewPatterns(s.top.patterns, newPatternId)
23     add(s.top, newPatternMCTs)
```

of the parent stack entry to produce new ones (line 18). Finally, the extended patterns together with newly combined ones are added to the parent stack entry.

Algorithm PatternStack constructs every pattern once and keeps only their ids in the stack entries. Every pattern has a unique string representation comprising the labels and keywords contained by the nodes. This representation is used for comparing patterns between them. Every time two patterns are combined to produce a new one, the ids of the combined patterns and that of the new one are kept in a table. Another table maps each pattern MCT to the pattern produced when the first is extended with an edge to the parent of its root (line 16). These two tables are consulted before any pattern is constructed (lines 16, 18, 21 22), in order to avoid reconstructing any patterns already created.

Once the patterns are computed the graph $G_{\prec}$ is constructed and the order $\mathcal{O}$ of patterns is produced.

5 Experimental Evaluation

We performed experiments to measure the efficiency and effectiveness of *XReason* as a filtering and ranking system. We compared the quality of our results to that of previous approaches.

For the experimentation, we use Mondial (1.7 MB), SIGMOD (467 KB), EBAY (34 KB) and NASA (23 MB) datasets obtained from the UW XML Data Repository[1]. We also use the benchmark dataset, XMark (150 MB) for scalability experiments. The experiments were conducted on a 2.9 GHz Intel Core i7 machine with 3 GB memory running Windows 7.

We first introduce the metrics we use for the experimental evaluation; then, we present our results on effectiveness for both filtering and ranking semantics, and finally we present our efficiency experiments.

5.1 Metrics

Since the ranking approaches we consider may view a number of results as equivalent (i.e., having the same rank) we extend below the metrics that are usually used to measure the quality of ranking. For the experiments, relevant results were determined by the experts after examining all the results.

Filtering Experiments. Since *XReason* works with patterns, if a pattern is minimal with respect to $\prec$, all candidate results that conform to that pattern are regarded as relevant and are returned to the user. We use *precision* and *recall* to measure the effectiveness of filtering semantics. *Precision* is the ratio of the number of relevant results in the result set of the system to the total number of results returned by the system. *Recall* is the ratio of the number relevant results in the result set of the system to the total number of relevant results.

Ranking Experiments. For the ranking experiments, we employ two metrics: Mean Average Precision (MAP) and R-Rank.

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

Table 1. Definitions of SLCA, ELCA and XReal semantics in terms of *IT*s

Approach	**Definition of the answer of Q on an XML tree T**
$SLCA$	$\{t \mid t \in ITset(Q,T), v = root(MCT(t))$, and $\nexists\, t',\ v'\ (t' \in ITset(Q,T),$ $v' = root(MCT(t'))$ and v' is a descendant of $v)\}$
$ELCA$	$\{t \mid t \in ITset(Q,T), v = root(MCT(t))$, and $\nexists\, v'(v'$ is a node in $MCT(t),$ $v \neq v'$ and $v' \in LCAset(Q,T)\}$
$ITReal$	$\{t \mid t \in ITset(Q,T), v = root(MCT(t))$, and $\exists\, v'\ (v' \in XReal_{Nodes}$ and v' is an ancestor of $v)\}$

MAP is the mean average precision of a set of queries with average precision of a query being the average of precision scores after each relevant result of the query is retrieved. As a ranking effectiveness metric, MAP takes the order of the results into account. *R-Rank* of a query is the reciprocal of the rank of the first correct result of the query. If no relevant results are retrieved, R-Rank is 0.

We extend both MAP and R-Rank so that they take into account equivalence classes of results. An equivalence class in a ranked list is a set of all the results which have the same rank. Different orderings of these results in the ranked list would affect the value of ranking metrics MAP and R-Rank. For this reason, we define and compute *worst* and *best* versions for MAP and R-Rank. In the *worst* (resp. *best*) version, the ranked list is assumed to have the correct results ranked at the end (resp. beginning) of each equivalence class. This extension allows us to compute upper and lower bounds for the ranking metrics between which the scores of all the possible rankings lie. We denote these metrics as MAP_{worst}, MAP_{best} and $R\text{-}Rank_{worst}$, $R\text{-}Rank_{best}$.

In order to assess the effect of answer set size on precision in ranking experiments, we also measure precision with a cutoff point for the number N of results which is called precision@N (P@N). Similarly to MAP and R-Rank we consider two versions of P@N: $P@N_{worst}$ and $P@N_{best}$.

5.2 Effectiveness of Filtering Semantics

For the filtering experiments, we compare *XReason* with two well-known baseline approaches SLCA [3, 7, 19] and ELCA [6, 20]. We also compare with an adaptation of a recent state-of-the-art approach, XReal [1].

In order to allow the comparison of *XReason*, which returns *IT*s and not simply LCAs with the other approaches, we provide definitions for a query answer according to SLCA and ELCA semantics in terms of *IT*s in Table 1. For a query Q on an XML tree T, $ITset(Q,T)$ denotes the set of *IT*s of the instances of Q on T. XReal infers promising result node types (label paths from the root) and ranks and returns the nodes that match these node types. In order to compare XReal with *XReason* we adjusted XReal in Table 1 so that it returns *IT*s and we named this new approach *ITReal*. $XReal_{Nodes}$ denotes the set of nodes in T that match the node type inferred by XReal.

Table 2. Queries used in the experiments

Dataset	Query ID	Keywords
Mondial	M1	*international, monetary, fund, established*
	M2	*government, democracy, muslim*
	M3	*jewish, percentage*
	M4	*new, york*
	M5	*north, atlantic, treaty, organization*
	M6	*bosnia, herzegovina, government*
SIGMOD	S1	*divesh, srivastava, database*
	S2	*michael, stonebraker, postgres*
	S3	*query, optimization*
	S4	*database, systems, security*
	S5	*christos, faloutsos, signature, files*
	S6	*efficient, maintenance, materialized, views, subrahmanian*
EBAY	E1	*shipping, 10, days*
	E2	*maxtor, hard, drive*
	E3	*cpu, 933, mhz*
	E4	*2, days*
	E5	*1, year, warranty*
	E6	*logitech, mouse, multimedia, keyboard*

We run 6 queries on each of the datasets shown in Table 2. Precision scores of systems are presented in Fig. 8. Since all the systems have perfect (100%) recall, recall scores are not displayed.

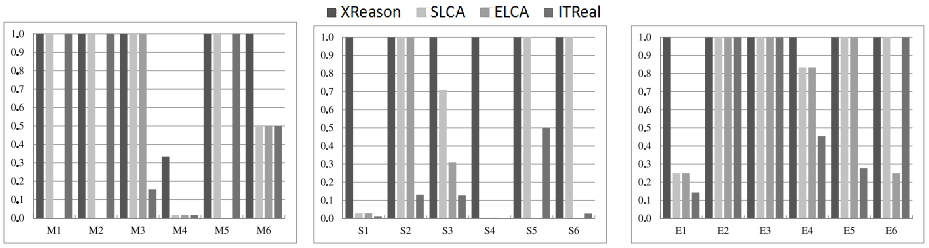
ELCA has the worst precision scores among the four semantics. ITReal has better scores on the Mondial dataset. Its precision is reduced if there are many irrelevant results under the nodes that match the node types returned as in the SIGMOD dataset. Both ITReal and *XReason* are also ranking systems and they can use their ranking capacity to reduce the negative effect of a large size answer set on precision. For this reason, in the next section we also measure P@N. As we can see, *XReason* outperforms the other approaches and shows very good precision and perfect recall on all cases.

5.3 Effectiveness of Ranking Semantics

In order to evaluate the effectiveness of the ranking semantics of *XReason* we computed the queries of Table 2 under *XReason* and ITReal semantics on the datasets and we measured best and worst bounds for MAP, R-Rank and P@N.

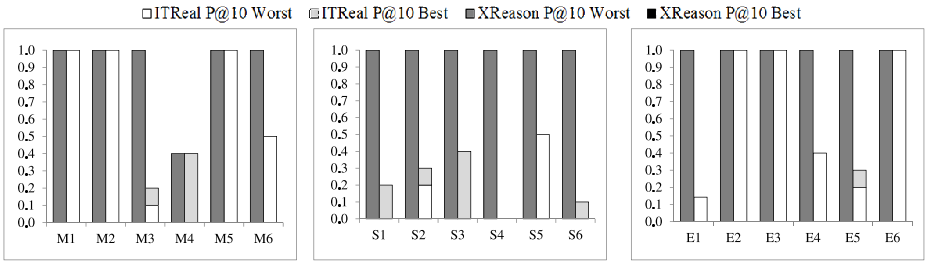
Table 3 shows the average scores for MAP and R-Rank. *XReason* outperforms ITReal with respect to both MAP and R-Rank. Since *XReason* ranks the correct answers almost always higher than the incorrect answers, it has almost perfect MAP and R-Rank scores.

Best and worst P@N scores are shown in Fig. 9. For a given query and a given approach, best and worst scores are shown on the same column with worst scores superimposing best scores, i.e., if the scores are the same, only worst scores are



(a) Mondial (b) SIGMOD (c) EBAY

Fig. 8. Precision scores for the queries of Table 2



(a) Mondial (b) SIGMOD (c) EBAY

Fig. 9. Best and worst P@10 scores for the queries of Table 2

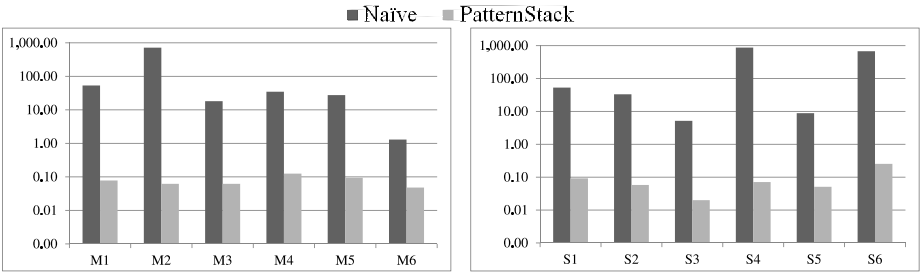
visible. $N = 10$ for all datasets because most of the queries have few correct results. For ITReal, limiting the result set size did not have a significant effect on the precision for most of the queries, which means that some incorrect results are ranked high in the result list. *XReason* has almost perfect P@10 scores in almost all cases.

5.4 Efficiency

In order to evaluate the efficiency of PatternStack algorithm we compared its performance with that of a naïve algorithm and we also ran scaling experiments. The naïve algorithm generates all the ITs of the query using the inverted lists of the keywords and iterates over them to extract the patterns.

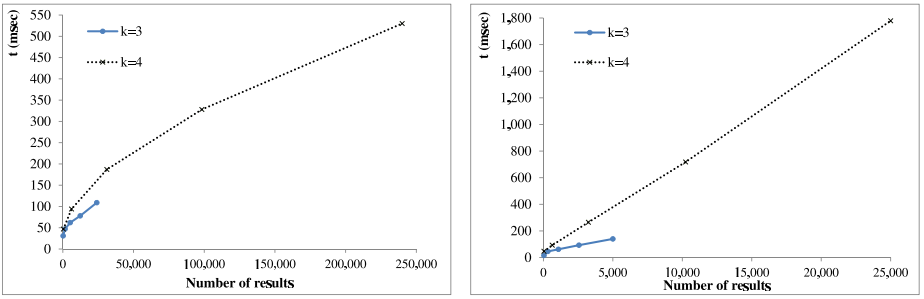
Table 3. Best and worst MAP and average R-Rank scores for the queries of Table 2

Dataset	Semantics	MAP_{worst}	MAP_{best}	$R\text{-}Rank_{worst}$	$R\text{-}Rank_{best}$
Mondial	*XReason*	0.95	0.95	1.00	1.00
	ITReal	0.60	0.87	0.59	1.00
SIGMOD	*XReason*	1.00	1.00	1.00	1.00
	ITReal	0.19	0.69	0.26	0.83
EBAY	*XReason*	1.00	1.00	1.00	1.00
	ITReal	0.60	0.80	0.53	1.00



(a) Mondial (b) SIGMOD

Fig. 10. Algorithm execution times (in secs) for the queries of Table 2



(a) NASA (b) XMark

Fig. 11. Response time for PatternStack for queries with 3 and 4 keywords

Fig. 10 shows the response times of the queries of Table 2 on the Mondial and SIGMOD datasets. Note that the y-axis is in logarithmic scale. PatternStack is at least two orders of magnitude faster than the naïve algorithm in all the cases. The response time of PatternStack is reasonable for real-time search systems even without optimizations.

We ran scalability experiments for PatternStack on NASA (a real dataset) and XMark (a benchmark dataset). We used queries with the first three and all four of the keywords in the sets $\{define(40), dates(30), boxes(20), bubble(10)\}$ and $\{conserve(25), preservings(20), dreamers(10), almond(5)\}$ for NASA and XMark respectively. The numbers in parenthesis indicate the number of instances of the respective keyword. For each query, we truncated the keyword inverted lists to the 20%, 40%, 60% and 80% of their total sizes.

The measured response times in relation to the total number of results are shown in Fig. 11. We can see that PatternStack scales smoothly and the response time is almost linear. Even though the number of results on XMark is smaller than that on NASA, the response times are larger. This is due to the fact that XMark is deeper (max. depth 11) than NASA (max. depth 7) which results in a much larger number of patterns in XMark than in NASA.

6 Related Work

Several papers elaborate on filtering and ranking semantics for keyword queries on tree-structured data. A number of them compute LCAs using structural [6, 16, 19] and/or semantic information [4, 8, 11, 18]. XMean [12] defines a conceptual relation among keyword matches using entities, and clusters the results according to their patterns.

XRank [6] uses a variation of PageRank algorithm to rank the results. XSearch [4] ranks the results using a tf-idf-like function. XReal [1] introduces a similarity function to rank nodes with respect to their similarity to the query. Termehchy and Winslett [17] exploit mutual information to calculate coherency ranking measures for ranking the query answer. Nguyen and Cao [15] use mutual information to compare results and to define a dominance relationship between results for ranking. SAIL [9] introduces the concept of minimal-cost trees and identifies the top-k answers by using link analysis and keyword-pair relevancy.

XReal [1, 2] defines the type of a node as the prefix path from the root to the node which includes labels of all the nodes on the path. A variation of tf-idf is used to find the candidate result type of a query. XBridge [10] uses a scoring measure as well but it also takes into account the structure of the results while scoring the type. XSeek [13] utilizes entity, attribute or connection nodes to decide upon the nodes to be returned in the results. MaxMatch [14] introduces the concepts of consistency and monotonicity for evaluating semantics assigning approaches.

Algorithms for finding SLCAs and ELCAs for a keyword query are presented in [19–22]. Hristidis et al. [7] develop efficient algorithms for finding a compact representation of the result subtrees. In [5], a multi-stack algorithm to return a size-ranked result list to a keyword query is presented. Chen and Papakonstantinou [3] introduce algorithms to support top-k SLCA and ELCA calculation.

7 Conclusion

The goal of this paper is to show that high quality semantics for keyword queries on tree-structured data can be obtained by extracting patterns of the query instances on the data tree and by reasoning on them, but also that this is computationally feasible. To this end we introduced *XReason*, an adaptive approach that does not use a scoring function to rank the results but does so by using homomorphisms between patterns. Our experimental results show that *XReason* outperforms many previous approaches in terms of quality of answers and can efficiently compute keyword queries on large data trees without requiring a preprocessing of the datasets.

An interesting feature of the "reasoning with patterns" aspect of our approach is that it can easily consider additional information (e.g, contextual, keyword instance statistical, or ontological information) during the reasoning process to filter out patterns and to further improve their ranking and the quality of the results. We are currently working towards this direction.

References

1. Bao, Z., Ling, T.W., Chen, B., Lu, J.: Effective XML keyword search with relevance oriented ranking. In: ICDE, pp. 517–528 (2009)
2. Bao, Z., Lu, J., Ling, T.W., Chen, B.: Towards an effective XML keyword search. IEEE Trans. Knowl. Data Eng. 22(8), 1077–1092 (2010)
3. Chen, L.J., Papakonstantinou, Y.: Supporting top-K keyword search in XML databases. In: ICDE, pp. 689–700 (2010)
4. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSEarch: A semantic search engine for XML. In: VLDB, pp. 45–56 (2003)
5. Dimitriou, A., Theodoratos, D.: Efficient keyword search on large tree structured datasets. In: KEYS, pp. 63–74 (2012)
6. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK: Ranked keyword search over XML documents. In: SIGMOD, pp. 16–27 (2003)
7. Hristidis, V., Koudas, N., Papakonstantinou, Y., Srivastava, D.: Keyword proximity search in XML trees. IEEE Trans. Knowl. Data Eng. 18(4), 525–539 (2006)
8. Li, G., Feng, J., Wang, J., Zhou, L.: Effective keyword search for valuable LCAs over XML documents. In: CIKM, pp. 31–40 (2007)
9. Li, G., Li, C., Feng, J., Zhou, L.: SAIL: Structure-aware indexing for effective and progressive top-k keyword search over XML documents. Inf. Sci. 179(21), 3745–3762 (2009)
10. Li, J., Liu, C., Zhou, R., Wang, W.: Suggestion of promising result types for XML keyword search. In: EDBT, pp. 561–572 (2010)
11. Li, Y., Yu, C., Jagadish, H.V.: Schema-free XQuery. In: VLDB, pp. 72–83 (2004)
12. Liu, X., Wan, C., Chen, L.: Returning clustered results for keyword search on XML documents. IEEE Trans. Knowl. Data Eng. 23(12), 1811–1825 (2011)
13. Liu, Z., Chen, Y.: Identifying meaningful return information for XML keyword search. In: SIGMOD, pp. 329–340 (2007)
14. Liu, Z., Chen, Y.: Reasoning and identifying relevant matches for XML keyword search. In: PVLDB, vol. 1(1), pp. 921–932 (2008)
15. Nguyen, K., Cao, J.: Top-k answers for XML keyword queries. World Wide Web 15(5-6), 485–515 (2012)
16. Sun, C., Chan, C.Y., Goenka, A.K.: Multiway SLCA-based keyword search in XML data. In: WWW, pp. 1043–1052 (2007)
17. Termehchy, A., Winslett, M.: Using structural information in XML keyword search effectively. ACM Trans. Database Syst. 36(1), 4 (2011)
18. Theodoratos, D., Wu, X.: An original semantics to keyword queries for XML using structural patterns. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 727–739. Springer, Heidelberg (2007)
19. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest LCAs in XML databases. In: SIGMOD, pp. 537–538 (2005)
20. Xu, Y., Papakonstantinou, Y.: Efficient LCA based keyword search in XML data. In: EDBT, pp. 535–546 (2008)
21. Zhou, J., Bao, Z., Wang, W., Ling, T.W., Chen, Z., Lin, X., Guo, J.: Fast SLCA and ELCA computation for XML keyword queries based on set intersection. In: ICDE, pp. 905–916 (2012)
22. Zhou, R., Liu, C., Li, J.: Fast ELCA computation for keyword queries on XML data. In: EDBT, pp. 549–560 (2010)

History-Offset Implementation Scheme of XML Documents and Its Evaluations

Tatsuo Tsuji, Keita Amaki, Hiroomi Nishino, and Ken Higuchi

Information Science Department, Faculty of Engineering, University of Fukui
Bunkyo 3-9-1, Fukui City, 910-8507, Japan
{tsuji,amaki,h_nisino,higuchi}@pear.fuis.u-fukui.ac.jp

Abstract. This paper presents a novel implementation scheme of XML documents. First, we describe a labeling scheme for dynamic XML trees, in which no relabeling is necessary against the structural update of trees by the help of small auxiliary data structure. Second, two kinds of encoding/decoding data structure are proposed for implementing XML documents based on *history-offset* encoding, which is designed for encoding multidimensional datasets. One is for XML tree structure and the other is for path expressions from the root node. By cross-referencing the encoded/decoded results obtained by using these data structures, the structural retrieval using both axis specification and path expressions can be performed very efficiently. Finally, using the constructed prototype system, the performance of our implementation scheme is evaluated and compared with eXist-db, a native XMLDB system.

1 Introduction

To handle XML document structure, it is important to provide a labeling scheme for XML nodes that can effectively capture the underlying XML tree structure. There are some sophisticated labeling schemes such as [1], [2], [3] with their encoding methods that support dynamic structural updates such as node insertions and deletions. The advantage of such schemes is that they can preserve the document order without relabeling any existing nodes even if dynamic nodes are inserted dynamically. However, the encoded label size may become very large with the dynamic insertions. [7] takes an approach that embeds an XML tree to a complete *k-ary* tree, and an improved scheme using this approach can be found in [8]. In these schemes, a query along an axis such as *parent* or *sibling* can be quickly answered through simple arithmetic operation on label values. However, they are basically for static XML trees; when new nodes are dynamically inserted, many node label values need to be recomputed to preserve the document order. In addition, they are not efficient in the usage of label value space and this space can be quickly saturated by node insertions.

The *history-offset* labeling scheme we are proposing in [4] embeds XML tree structure into a dynamically extendible multidimensional array [5] and takes advantage of the array structure in both space and time costs. The following are significant advantages of this scheme over the above-mentioned labeling schemes. The storage cost for label values is smaller irrespective of the node insertion order and position in the XML tree. In addition, structural retrieval along an *axis* such as parent, siblings, or

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 315–330, 2013.

descendents is faster. In general a node labeling scheme can handle XML tree structure, but the representation of XML documents including element or attribute names are not considered. Therefore, a *path expression* of a node from the root node cannot be handled using the element names on the path.

This paper presents a new implementation scheme of XML documents. Two types of encoding/decoding data structure are constructed based on the history-offset encoding/decoding methods. One is for XML tree structure and the other is for path expressions from the root node. By cross-referencing the encoded/decoded results obtained by using these data structures, the structural retrieval by both axis specification and path expressions can be performed very efficiently. Using the constructed prototype system, the performance of our proposed scheme is evaluated. In the evaluation, our system is compared with eXist-db(Release-1.4.1) [6], a native XMLDB system. The evaluation proves that in most retrieval conditions using XPath our system exhibits higher performance than eXist-db and the dynamic structural updates of XML trees outperforms eXist-db.

2 History-Offset Encoding

History-offset encoding[17] is for encoding multidimensional datasets on the basis of the concept of *extendible arrays.* HOMD is an implementation scheme of multidimensional datasets using history-offset encoding.

2.1 Extendible Arrays

An n dimensional extendible array A has a history counter h and three kinds of auxiliary table for each extendible dimension $i(i = 1,...,n)$. See Fig.1. These tables are *history table* H_i, *address table* L_i, and *coefficient table* C_i. The history tables memorize extension history of A. When the current size of A is $[s_1, s_2, \ldots, s_n]$, for an extension of A along dimension i, contiguous memory block that forms an n - 1 dimensional subarray S of size $[s_1, s_2,...,s_{i-1}, s_{i+1},..., s_{n-1}, s_n]$ is dynamically allocated and attached to A. Then the history counter h is incremented by one, and the value is memorized at $H_i[s_i+1]$, while the first address of S is held at $L_i[s_i+1]$. Note that an extended subarray S is uniquely identified by the history value $H_i[s_i+1]$.

As is well known, element $(i_1, i_2, ..., i_{n-1})$ in an n-1 dimensional fixed size array of size $[s_1, s_2, ..., s_{n-1}]$ is allocated on memory using addressing function as follows:

$$f(i_1, ..., i_{n-1}) = s_2 s_3 \ldots s_{n-1} i_1 + s_3 s_4 \ldots s_{n-1} i_2 + \ldots + s_{n-1} i_{n-2} + i_{n-1} \quad (1)$$

We call these n-2 coefficients $(s_2 s_3 \ldots s_{n-1}, s_3 s_4 \ldots s_{n-1}, ..., s_{n-1})$ in (1) as a *coefficient vector.* The vector is computed at array extension and is held in the coefficient table $C_i[s_i+1]$. Note that if n is less than 3 the table can be void. Using these three kinds of auxiliary table, the address of an array element $(i_1, i_2, ..., i_n)$ can be computed as follows:

(a) Determine the largest history value in $\{H_1[i_1], H_2[i_2], \ldots, H_{n-1}[i_n]\}$. If it is $H_k[i_k]$, the subarray identified by $H_k[i_k]$ includes the element.

(b) Using the coefficient vector at $C_k[i_k]$, the *offset* of the element $(i_1, ..., i_{k-1}, i_{k+1}, ..., i_n)$ in the subarray corresponding to $H_k[i_k]$ is computed according to its addressing function in (1).

(c) The sum of $L_k[i_k]$ and the *offset* computed in (b) is the address of the element.

For example, consider element (2,2) in Fig. 1. We compare the history values $H_1[2]=3$ and $H_2[2]=4$. Since $H_2[2]>H_1[2]$, (2,2) is involved in the subarray S corresponding to the history value 4 in the first dimension and the first address of S is found out in $L_2[2]=56$. The offset of (2,2) from the first address of S is 2, therefore the element address is determined as $L_2[2]+2=56+2=58$.

Note that we can use such a simple computational scheme to access an extendible array element only at the cost of small auxiliary tables.

2.2 History-Offset Encoding

Fig. 2 shows a realization of a two attribute relational table using a two dimensional extendible array. In general, for an n dimensional dataset R, each attribute of R can be mapped to a dimension of an n dimensional extendible array A and each attribute value of a tuple in R can be uniquely mapped to a subscript of the dimension of A. Hence, each tuple in R can be uniquely mapped to an n dimensional coordinate of an element in A. Furthermore each coordinate of an element in A can be mapped to a pair of scalar values, one of which is the history value of the subarray S including the element and the other is the offset value in S. In consequence, each tuple in R can be encoded to its corresponding pair of <*history value, offset value*>. We call this encoding of the tuples in R as *history-offset* encoding of R. A will be often called as a *logical extendible array*.

In the coordinate method, the length of the coordinate is directly proportional to the dimension of the extendible array. But in the history-offset encoding, even if the dimension is high, the size of the encoded tuple is fixed as short; i.e. only two kinds of scalar value. This encoded tuple can be easily decoded to the set of attribute values of the tuple.

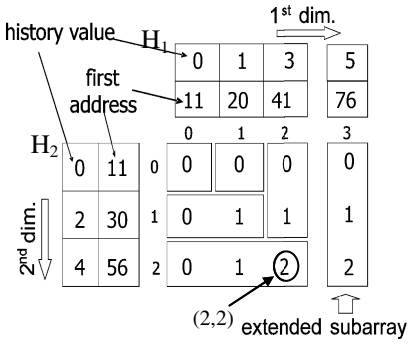


Fig. 1. An extendible array

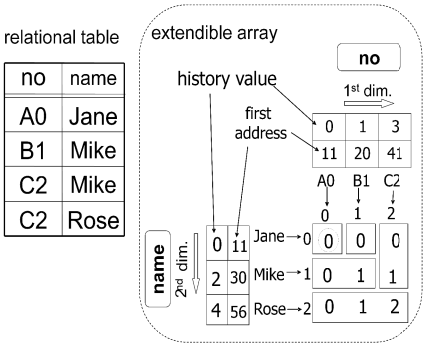


Fig. 2. Relational table implementation using an extendible array

2.3 HOMD

Fig. 3 shows the implementation of the relational table in Fig. 2 by using HOMD(History Offset implementation for Multidimensional Datasets). For an n attribute multidimensional dataset R, HOMD consists of the following five kinds of data structure.

CVT: Attribute value to subscript ConVersion Tree; CVT_k for the k-th attribute of R is a B^+ tree with each distinct attribute value v as a key and its associated data value is subscript i of the k-th dimension of the logical extendible array A. Hence the entry of the *sequence set* of the B^+ tree is the pair (v, i). Subscript i references to the corresponding entry of the HOMD table below.

HT: HOMD Table; HT_k corresponds to the auxiliary table of the k-th dimension explained in Section 2.1. It includes the history table and the coefficient table. Note that the address table can be void in our HOMD physical implementation.

AT: Attribute value Table; AT_k stores attribute values of the k-th dimension for converting a subscript of the k-th dimension to its corresponding attribute value.

RDT: The set of pairs <*history value, offset value*> for all of the effective elements in the extendible array are housed as the keys in a B^+ tree called RDT(Real Data Tree). Here, "effective elements" mean corresponding to the tuples in the multidimensional dataset.

HOMD: HOMD implementation of R is the set of n CVTs, n HTs, nATs and RDT.

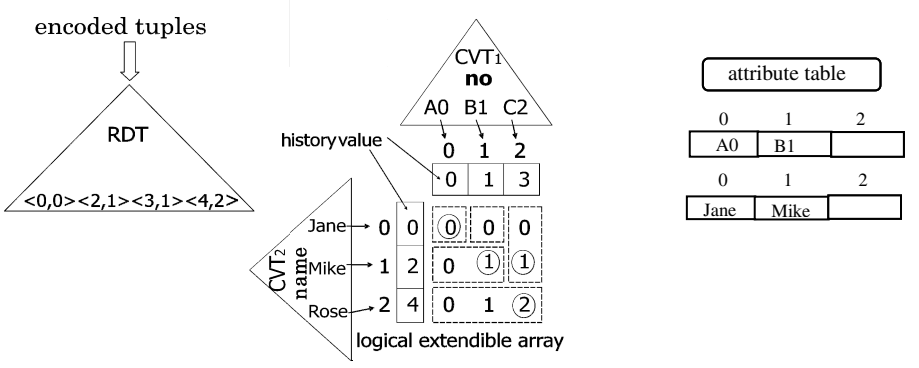


Fig. 3. HOMD implementation of Fig.2

3 History-Offset Labeling Scheme for Dynamic XML Trees

In this section, we present our labeling scheme for XML tree nodes, using the *history-offset* encoding described in Section 2.2. Our scheme takes advantage of the history-offset method by embedding an XML tree into HOMD data structures mentioned in Section 2.3.

3.1 Node Labeling Using History-Offset Encoding

An XML tree can be embedded into HOMD data structures described in Section 2.3. See Fig. 4. Each depth level of an XML tree is mapped to a dimension of HOMD, and the relative position of each node among its siblings is uniquely mapped to a subscript of the dimension. Thus, the coordinate of a node can be uniquely specified in HOMD. Using our *history-offset* method, a node can be labeled with the pair of history value

and offset value, which will be stored in RDT of HOMD. A coefficient vector is prepared for each subarray in HOMD, and it can be used for computing the labels of the parent, children and sibling nodes of the *context* node. In Fig. 4, the extension of subarrays and the computed <history value, offset value> labels of the tree nodes are shown assuming that the XML tree grows according to the *pre-order* node insertions.

Note that in our *history-offset* based labeling scheme, dynamic extendibility of HOMD makes the relabeling of the existing nodes unnecessary when any new nodes are dynamically inserted. In addition, even if an XML tree grows, namely the dimensionality of HOMD increases, the size of a node label value can be fixed as short. The benefits of our history-offset labeling scheme for dynamic XML trees rely on this extendibility of HOMD.

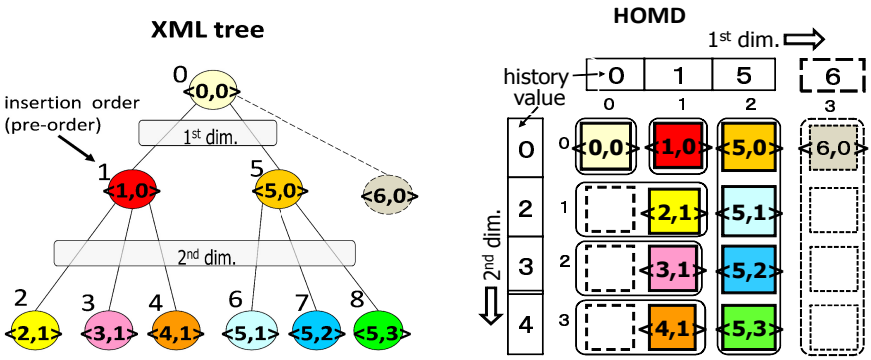


Fig. 4. Labeling XML tree using history-offset method

3.2 Handling Insertion and Deletion of an XML Tree Node

XML tree node insertion may possibly cause extension of the logical extendible array in HOMD. This extension would occur when a new node insertion causes the maximum fan out of the level to increase. For example, we consider the XML tree shown in Fig. 4. If a new node is inserted on the first level of the XML tree, the first dimension of HOMD would be extended and a subarray of history value 6 is dynamically allocated and attached along the first dimension. The node of label <6, 0> can be placed at the position of coordinate (3, 0). However, node deletion may possibly cause the reduction of the logical extendible array. For example, deletion of all nodes in the subtree rooted at node labeled <5, 0> would cause the subarray of history value 5 to be deleted and the first dimension of HOMD would shrink by one.

3.3 Handling Dynamic Changes of XML Tree Height

The HOMD dimension is directly proportional to the XML tree height, namely the maximum nested level of the XML document. Dimensional extension of HOMD can be handled very efficiently. Fig. 5 shows addition of a new dimension against a two dimensional HOMD. When a new dimension is added to an n dimensional HOMD, HOMD table HT(See Section 2.3) of the new dimension is created and

initialized. All existing elements in the original HOMD becomes n+1 dimensional and the subscript of the newly added dimension becomes 0. Note that relabeling of the nodes in the original HOMD is not necessary, since <history value, offset value> labels of any nodes are not necessary to be modified. Conversely, in the case that the height of an XML tree decreases by the deletion of all nodes on the maximum level of the XML tree, the dimension of HOMD decreases by one. Note also that relabeling of the nodes in the original HOMD is not necessary. However, when *drop column* or *add column* SQL command for a relational table is performed, a large overhead arises since reorganization of all existing tuples would be necessary.

3.4 Preserving Order among Siblings in Dynamic Environment

HOMD cannot preserve the logical order (i.e., document order) among siblings in dynamic environment, because the subscript value for each child node of the same parent is assigned in the order of insertion irrespective of the logical order among them. In order to preserve the logical order among sibling nodes, an additional table called *OS* (Order of Siblings) *table* is maintained in each parent node. The OS table is an array and serves to rearrange the subscripts in the sibling order as shown in Fig. 6.

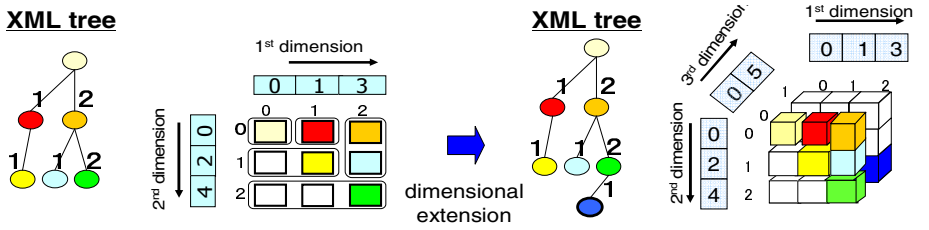


Fig. 5. Dimensional extension in HOMD

3.5 Axis Computation

This section describes the computation of *axes* in XPath language[9] using the labeling scheme presented in Section 3.1. In our history-offset encoding method, the label can be calculated on the basis of its corresponding coordinate in the logical extendible array in HOMD. In Fig. 7, assume the context node is (2, 2, 0, 0). The coordinate of its parent node can be obtained by replacing the subscript of the second dimension (i.e., the level of the context node) with 0. Therefore the coordinate of the parent node is (2, 0, 0, 0), which is encoded to the label <*history value*, *offset value*>.

Decoding of the label <*history value*, *offset value*> of a node to its corresponding coordinate $(i_1, i_2, \ldots, i_n)$ can be performed as follows:

(a) The dimension p and subscript i_p corresponding to the subarray that includes the node can be known from the *history value* by inspecting the history tables.
(b) Let $o(o > 0)$ be the *offset value* of the label and let the coefficient vector $C_p[i_p]$ of the subarray known in step (i) be $(c_1, c_2, \ldots, c_{n-2})$. Then o can be computed as follows:

$$o = c_1 i_1 + c_2 i_2 + \ldots + c_{p-1} * i_{p-1} + c_p * i_{p+1} + \ldots + c_{n-2} * i_{n-1} + i_n$$

Therefore, the coordinate i_k can be simply computed as:

$$i_k = (o\%c_{k-1})/c_k \quad (\text{if } k < p), \qquad i_k = (o\%c_{k-2})/c_{k-1} \quad (\text{if } k > p).$$

Here, % denotes the residue of the division.

Conversely, encoding of the coordinate of a node to <*history value*, *offset value*> can be performed as in the element address computation of an extendible array explained in Section 2.1; the largest history value decided in step (a) gives *history value* and the offset in step (b) gives *offset value.*

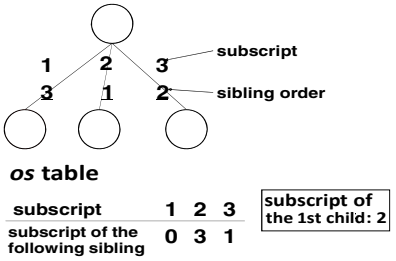


Fig. 6. OS table

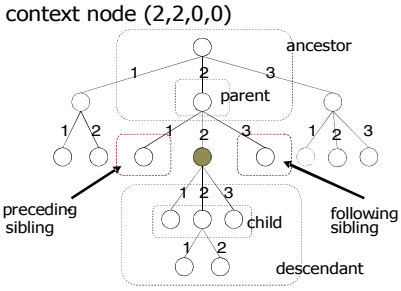


Fig. 7. XML tree axes

4 History-Offset Implementation Scheme of XML Documents

In Section 3, the structural representation of an XML tree was described, but the XML primitives for representing XML documents, such as *element names*(tag names), *path expressions*(concatenation of element names) or *attribute names* have not been considered. In this section an implementation scheme of XML documents is proposed incorporating such primitives for representing XML documents.

4.1 nHOMD and pHOMD

The HOMD for representing XML tree structure described in Section 3 is called an nHOMD(*node* HOMD). It only represents the XML tree structure. The label <history value, offset value> of a node is called nID and the RDT of an nHOMD is called nRDT. A set of nIDs is stored in nRDT. It should be noted that among the nHOMD components, the CVT of each dimension does not exist, since the content information such as element or attribute names are not involved in nHOMD.

In XML documents, more than two child nodes of the same element name may exist, and more than two distinct routes from the root node can be represented by the same path expression. Here, in order to solve this non-uniqueness problem, *structural summary* of an XML document (See Fig. 8), in which sibling nodes of the same element name are unified to a single node, is used in combination with nHOMD. For implementing the structural summary, we also employ HOMD separately from nHOMD. This HOMD is called *path* HOMD (pHOMD). By using pHOMD, a path expression from the root node can be encoded into the corresponding <history value,

offset value> label. This label value is called pID and the RDT of the pHOMD is called pRDT. Unlike nHOMD, pHOMD handles document information such as element or attribute names, so it requires all kinds of HOMD component including CVTs.

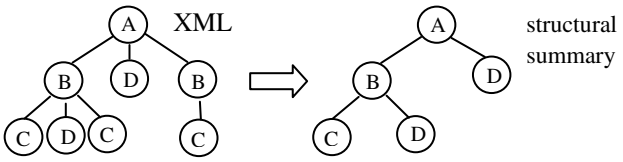


Fig. 8. Structural summary of an XML tree

4.2 Storing Path Expressions Using pHOMD

As in nHOMD, each level of the structural summary of an XML tree corresponds to a dimension of the logical extendible array of pHOMD and each element name of a node (simply referred to as *node name* in the following) on the same level of the summary tree corresponds to a subscript of its dimension (See Fig. 9). If more than two nodes of the same name exist, they are assigned to the same subscript. Each node name is converted to its subscript of the dimension by using the corresponding CVT. A path expression on the summary tree is converted into a coordinate of the corresponding array element of pHOMD and the coordinate can be encoded to <history value, offset value> as pID, and stored in pRDT as a key value. Note that the nodes reached by the same path expression in an XML tree can be encoded to the same pID.

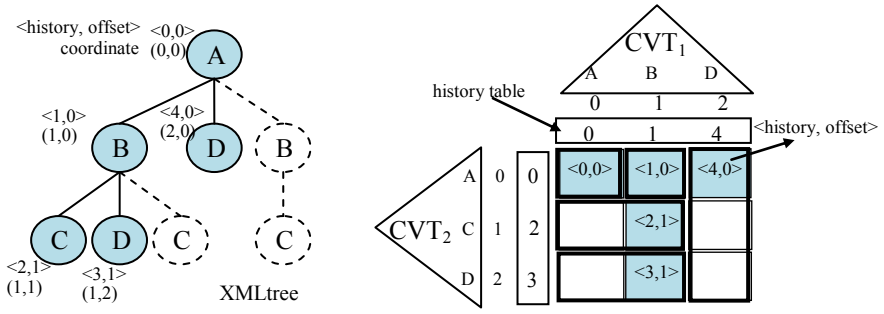


Fig. 9. HOMD for storing path expressions (pHOMD)

4.3 Data Structures for Cross-Referencing pID and nID

Fig. 10 shows the relationship between nHOMD and pHOMD. For each node in the XML tree, its nID in nHOMD corresponds to the unique pID of the path expression from the root node to it. For each path expression in the pHOMD tree, its pID corresponds to the set of nIDs, each of whose nodes in the XML tree is reached by the path identified by the pID. In order to establish these relationships in nHOMD and

pHOMD, in the *sequence set* of nRDT, for a key value nID, its pID is stored as a data value. Conversely, in the sequence set of pRDT, for a key value pID, the reference to its corresponding list of nIDs is stored as a data value. Note that by cross-referencing pID and nID using pHOMD and nHOMD, both the path expression retrieval and structural retrieval along the specified axis can be performed quickly.

4.4 Conversion between Node Label and Path Expression

For a node label (nID), its corresponding path expression *pe* can be known as follows. First, by searching nRDT with nID as a key, the pID of *pe* can be known, then the pID is decoded to the coordinate of the subscripts using the coefficient vector table of pHOMD. Since each subscript of the coordinate corresponds to its node name, by referring to the attribute tables of pHOMD mentioned in Section 2.3, each node name in *pe* can be obtained. Conversely, for a path expression *pe*, its corresponding set of node labels can be known as follows. First, using pHOMD including CVTs, *pe* is encoded to its pID, then the pID is searched in pRDT to get the reference to the set of node labels (nIDs).

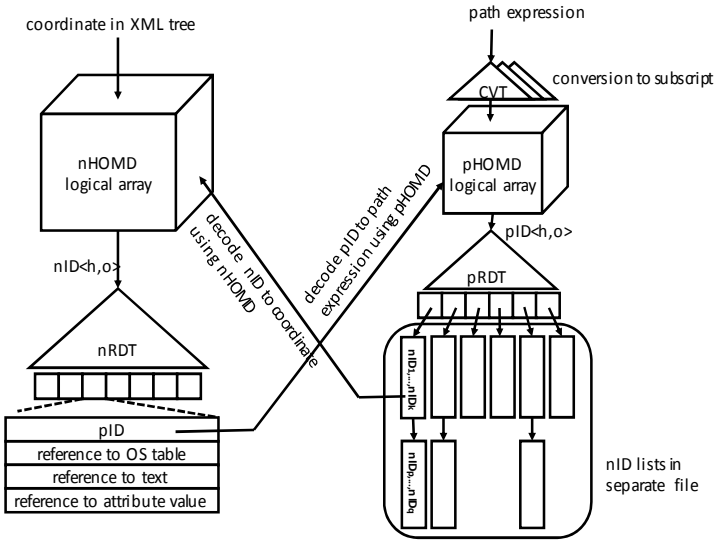


Fig. 10. Cross-referencing of pID and nID using nHOMD and pHOMD

5 Structural Retrieval Using nHOMD and pHOMD

nHOMD plays a important role in structural retrieval along an axis on an XML tree. In this section we describe the procedures of structural retrieval along some axes based on the example illustrated in Fig. 11. Fig. 11 shows an XML tree and the logical extendible array in nHOMD. Here, the nID of the c*ontext node* is assumed to be <5,0> and its coordinate of the extendible array to be (3,0).

(1) Retrieval of the child nodes
The OS table of the context node <5,0> is accessed in nRDT, by which the subscript of each child node is known in the document order. In Fig. 11, the subscripts 1 and 2 of the second dimension are known in this order. Therefore, from the coordinate (3,0) of the context node, the coordinates of the child nodes are known to be (3,1), (3,2). By using nHOMD, these coordinates are encoded into the child node nIDs <5,1>, <5,2> respectively. The path expressions corresponding to the obtained nIDs can be known by using the procedure mentioned in Section 4.4.

(2) Retrieval of the parent node
Let *dm* be the largest dimension of the context node's coordinate whose subscript value is non-zero. The coordinate of the parent node can be determined by replacing the non-zero subscript of the dimension *dm* of the context node with 0. Since the context node's coordinate is (3,0) and *dm* is the first dimension, the coordinate of the parent node is determined to be (0, 0), which is encoded to nID <0,0>.

(3) Retrieval of the sibling nodes
To retrieve the sibling nodes, first the parent node of the context node is determined. The nID of the parent node is <0,0>, which is searched in nRDT and its OS table is obtained. The subscripts arranged in the document order are read from the OS table; in the order 1, 2, 3 in this example. These subscripts are combined with the parent's coordinate (0, 0). Therefore the coordinate of the first child is (1,0) and this is encoded to nID <1,0>. In the same way, the following siblings (2,0) (nID <4,0>) and (3,0) (nID<5,0>) can be obtained. Since the latter is the context node itself, the elder siblings nID<1,0> and nID<4,0> are obtained. No younger siblings exist. The path expressions corresponding to the obtained nIDs can be known by using the procedure mentioned in Section 4.4.

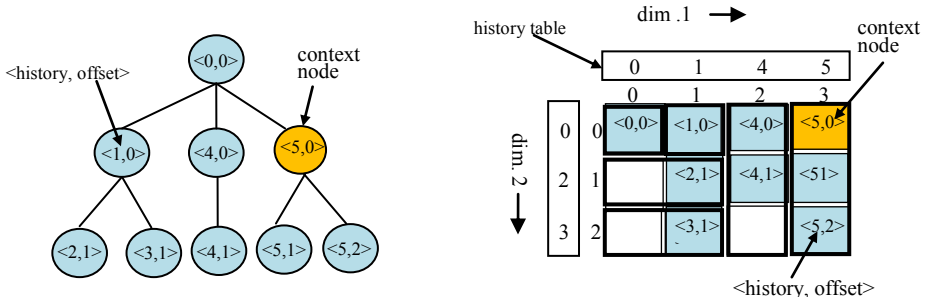


Fig. 11. An XML tree and its logical array of nHOMD

6 Related Work

As is well known, the implementation schemes of XMLDB include mapping XML documents into relational tables and constructing native XMLDB for storing XML documents. The most significant advantage of the former is that the database functions, such as query processing or transaction processing, provided by RDBMS can be

utilized, but both storage cost and retrieval cost increase because of the required mapping. Many commercial RDBMSs, such as Oracle XDK[10], provide XML documents processing capabilities as their extensions, and so many XMLDB systems such as [11] are designed for research purpose. On the other hand, in the latter implementation scheme both storage and retrieval costs are smaller, but the system development cost is rather high. Many native XMLDB systems such as eXist-db[6], Xindiche[12], DB2[13] are developed, .

Along with these XMLDB system developments, various kinds of XML node labeling scheme reflecting the underlying XML tree structures have been proposed and analyzed. These include range-based labeling scheme [14], prefix scheme [15], and the scheme for mapping XML trees to *k-ary* complete trees [7][8]. These labeling schemes make it possible to traverse structural relationships such as *parent, children, siblings, ancestors,* or *descendents* efficiently by only inspecting node labels, but they are static labeling schemes and for the dynamic new node insertions, a part of or all of existing node labels should be recalculated.

On the other hand, node label encoding schemes for dynamic XML trees such as ORDPATH [1], QED [2], DLN [3] and prime number labeling scheme [16] have been proposed. eXist-db[6] employs DLN labeling scheme. The main benefit of these labeling schemes is that they can preserve the document order without relabeling even if structural updates are made against the XML tree. However, in these labeling schemes, label sizes can be greatly increased if nodes are concentratedly inserted around the close position. In contrast, our *history-offset* labeling scheme does not share this deficiency and the label size does not increase irrespective of the insertion positions. The *history-offset* encoding scheme has been originally designed for encoding multidimensional datasets [17]. Some works on the storage schemes for data warehousing based on history-offset encoding can be found in [18] or [19].

7 Experiments and Evaluations

Using a prototype system incorporating our proposed implementation scheme, experiments have been conducted. Its storage cost and retrieval performance are measured in the following computing environment. Retrieval performance is measured on some typical query path expressions using XPath.

CPU:Intel Xeon(R) X7560(2.27GHz), Memory: 132GB, OS: Linux 2.6.18

In the experiment, our constructed system was compared with eXist-db (version 1.4.1-rev15155), which is one of the native XMLDB systems. The reason for adopting eXist-db is that it is well maintained and its node labeling scheme DLN is similar to our scheme, in which no relabeling is necessary against dynamic structural updates of an XML tree. We show the results of two types of experiments. For eXist-db, one is the most advantageous node insertion order (Section 7.1) and the other is nearly the most disadvantageous node insertion order (Section 7.2). Essentially our scheme is not influenced by the node insertion order. Our prototype system is written in C language, while eXist-db is written in Java.

7.1 Comparisons When an XML Document Is Loaded in Preorder

In eXist-db, the instruction was issued to load an XML document into DB in preorder, while in our system the same XML document was scanned through and also stored in preorder into nHOMD and pHOMD data structures. Using XMark benchmark data [20], the employed XML document was generated as follows:

Total number of nodes 370,311
(Document file size: 9.8502MB)
(Element nodes: 200,336, attribute nodes: 53,512, text nodes: 116,463)

(1) Comparison of storage costs
The storage cost of our system was the total size of nHOMD, pHOMD, OS tables and the external file containing the content of the document. The storage cost of eXist-db is the total size of the produced files in the DB directory after storing XML document into its DB. Table 1 shows the results. The major files used in eXist-db will be explained in Section 7.3. The cost of our system was 1.16 times larger than that of eXist-db. The greater part of our system was occupied by nHOMD, in which the size of nRDT storing nIDs was dominant. Size of nRDT is the size of the file storing the set of pairs nID and its data (pID, references to OS table, attribute values, and texts (See Fig. 10)). The B^+ tree for nRDT was constructed from this file at opening DB. The data structures of nHOMD other than nRDT, such as history tables or coefficient vector tables described in Section 2 were very small and negligible. On the other hand, data structures of pHOMD are very small compared with those of nHOMD and they are placed in the main memory. They are frequently accessed in the path expression retrieval.

(2) Comparison of retrieval costs
The retrieval costs were measured using XPath queries on the XML document specified above. The results are shown in Table 2. In any queries, the performance of our system overweighed that of eXist-db. In particular, for the queries that consist of only element names, our system far outweighed eXist-db. Conversely, for the queries that specify attribute values or texts, the performance degraded in the case of a large number of retrieval results, because much time was spent in seeking the position of data stored in the external files.

Table 1. Storage costs

our scheme		eXist-db	
data structure	storage size [MB]	file	storage size [MB]
nHOMD	10.140	dom.dbx	16.4727
- nRDT in nHOMD	9.8884	elements.dbx	2.1406
pHOMD	0.03528	collections.dbx	0.01172
-p RDT in pHOM	0.01543	values.dbx	0.007813
nID list	5.0938	ngram.dbx	0.007813
OS tables	0.03020	symbols.dbx	0.0008850
text data	5.7738		
attribute values	0.5857		
total	21.6588	total	18.6415

Table 2. Retrieval costs for the XMLDB constructed in preorder

query path expression	our scheme[msec]	eXist-db[msec]	# of hits
/site/regions/africa/item/description/parlist/listitem	4.8993	171.1	65
/site/closed_auctions/closed_auction/price	81.0875	260.8	1365
/site/people/person/name	119.182	401.4	3570
/site/regions/*/item/description	212.9675	402.7	3045
/site/regions/africa/item/location[text()='United States']	5.2991	185.8	62
/site/regions/africa/item[@featured='yes']	4.3994	152.9	7
/site/regions/*/item	17.4974	366.8	3045
/site/regions/*/item/location[text()='United States']	258.1609	616.8	2284
/site/regions/*/item[@featured='yes']	178.773	250.1	303

Table 3 shows the retrieval times to get the nodes that meet the XPath queries by using only pHOMD. In this experiment, since the structural data in nHOMD was not used, the retrieved results could not be sorted in the document order. In addition, queries that specify attribute values or texts could not be issued since only nHOMD stores these data as shown in Fig. 10. Table 3 shows the total retrieval times measured 1000 times for each query. It is negligible compared with that of nHOMD. The retrievals of such simple path expressions are very fast if the results do not need to be output in the document order.

Table 3. Retrieval costs by using only pHOMD

query path expression	retrieval time x1000 times [msec]
/site/regions/item/description/parlist/listitem	3.7993
/site/closed_auctions/closed_auction/price	22.0967
/site/people/person/name	52.5920
/site/regions/*/item/description	64.1902
/site/regions/*/item	63.5904

7.2 Comparison When Dynamic Node Insertions Are Performed Concentratedly

In this experiment, as shown in Fig. 12, XML nodes were appended between the context node and last inserted node; hence, concentratedly inserted around the same position. These insertions of the new nodes cannot be performed in a *batch manner* mentioned in Section 7.1, but require dynamic insertions. In eXist-db, XUpdate statement can be used for dynamic insertions. However the statement was very time consuming in determining the insert position in the DOM; therefore a smaller XML file generated by XMark was used as input.

Total number of nodes 112,340
(Document file size: 9.8502MB)
(Element nodes: 200,336, attribute nodes: 53,512, text nodes: 16,463)

The storage and retrieval costs were measured in storing the XML file in both preorder and the concentrated insertions stated above.

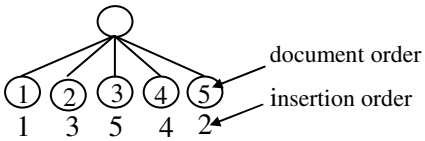


Fig. 12. Document order and insertion order in concentrated insertions

(1) Comparison of storage costs

Table 4 summarizes the total storage costs in preorder and concentrated node insertions. In eXist-db, we can observe that the cost of concentrated insertions was 1.8 times larger than that of preorder insertions. This is because of the property that the labeling scheme of DLN allows the label size to become very large by the concentrated insertions. Note that this property is shared by other labeling schemes including ORDPATH [1] and QED [2]. Conversely, the storage cost of our scheme was nearly constant irrespective of the node insertion order.

(2) Comparison of retrieval costs

Table 5 shows the retrieval costs in preorder and concentrated node insertions. In eXist-db, in some queries, the retrieval time increased. This is because since the label size tends to become large in DLN, the retrieval of nodes with many sibling nodes is time consuming in decoding long labels.

7.3 Used Files and Indexing in eXist-db

The major files used in eXist-db are shown in Table 1. collection.dbx stores the meta information for the XML document. dom.dbx stores persistent DOM and stores node data. symbols.dbx stores the names of nodes, attributes, namespaces. eXist-db provides

Table 4. Total storage costs for small-sized XML documents

	our scheme [MB]	eXist-db [MB]
preorder insertions	12..4047	10.6728
concentrated insertions	12.5114	19.1728

Table 5. Retrieval costs for small-size XMLDB

	preorder insertions[msec]		concentrated insertions [msec]	
query path expression	our scheme	eXist-db	our scheme	eXist-db
/site/regions/africa/item/description/parlist/listitem	2.8995	153.5	2.9991	176.5
/site/closed_auctions/closed_auction/price	47.6927	210.4	37.2945	266.4
/site/people/person/name	65.7902	319.7	85.6871	326.6
/site/regions/*/item/description	121.0816	312.7	129.8802	437.5
/site/regions/africa/item/location[text()='United States']	2.9994	157.7	3.2994	157.5
/site/regions/africa/item[@featured='yes']	2.7997	142	2.7994	142.9
/site/regions/*/item	10.0985	276	14.7979	282.7
/site/regions/*/item/location[text()='United States']	145.9775	416.5	157.576	420.9
/site/regions/*/item[@featured='yes']	102.3845	208.5	107.0839	215.9

some kinds of index. elements.dbx is the structural index for XML elements and attributes. The other indices are for string search and full text search using Apache Lucene[21], which were not used in our experiment.

8 Conclusion

In this paper, first we described labeling scheme for dynamic XML trees by using *history-offset* encoding, which has been designed for encoding multidimensional data. Second, on the basis of history-offset encoding, we proposed a scheme of implementing XML documents. Two kinds of encoding/decoding data structure were provided; one for XML tree structure and the other for path expressions. Finally, using the constructed prototype system, the performance of our scheme was evaluated and compared with the native XMLDB system eXist-db. It proved that the storage cost of our scheme was more than that of eXist-db, but the retrieval cost of our scheme outperformed that of eXist-db in almost every kind of XPath query. We also verified the advantage of our scheme that the storage cost is not dependent on the node insertion order and places.

References

1. O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Schaller, G., Westbury, N.: ORDPATHs: insert-friendly XML node labels. In: Proceedings of 2004 ACM SIGMOD Conference, pp. 903–908 (2004)
2. Li, C.Q., Ling, T.W.: QED: a new quaternary encoding to completely avoid re-labeling in XML updates. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 501–508 (2005)
3. Bohme, T., Rahm, E.: Supporting Efficient Streaming and Insertion of XML Data in RDBMS. In: Proceedings of Third International Workshop on Data Integration over the Web (DIWeb), pp. 70–81 (2004)
4. Li, B., Kawaguchi, K., Tsuji, T., Higuchi, K.: A Labeling Scheme for Dynamic XML Trees Based on History-offset Encoding. IPSJ Transactions on Databases 3(1), 1–17 (2010)
5. Otoo, E.J., Rotem, D.: Efficient Storage Allocation of Large-Scale Extendible Multi-dimensional Scientific Datasets. In: Proc. of SSDBM, pp. 179–183 (2006)
6. eXist-db Open Source Native XML Database, `http://eXist-db.sourceforge.net/`
7. Meier, W.: eXist: An Open Source Native XML Database. In: Chaudhri, A.B., Jeckle, M., Rahm, E., Unland, R. (eds.) Web Database and Web Services 2002. LNCS, vol. 2593, pp. 169–183. Springer, Heidelberg (2003)
8. Sato, T., Satomoto, T., Kobata, K., Pan, T.H.: Numbering Scheme which Identifies Tree Structures. In: Proceeding of DEWS 2002 Conference, A4-8 (2002)
9. XML Path Language (XPath), `http://www.w3.org/TR/xpath`
10. Oracle XML Developer Kit, `http://www.oracle.com/technetwork/developer-tools/xmldevkit/index.html`

11. Yoshikawa, M., Amagasa, T., Shimura, T., Uemura, S.: XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases. ACM Transactions on Internet Technology 1(1), 110–141 (2001)
12. The Apache Foundation: Xindice 1.1 Developer Guide, http://xml.apache.org/xindice/guide-developer.html
13. DB2 pureXML – Intelligent XML database management ALT, http://www-01.ibm.com/software/data/db2/xml/
14. Grust, T.: Accelerating XPath location steps. In: Proceedings of ACM SIGMOD Conference, pp. 109–120 (2002)
15. Tatarinov, I., Viglas, S., Beyer, K., Shanmugasundaram, J., Shekita, E., Zhang, C.: Storing and Querying Ordered XML Using a Relational Database System. In: Proceedings of 2002 ACM SIGMOD Conference, pp. 204–215 (2002)
16. Wu, X.D., Lee, M.L., Hsu, W.: A Prime Number Labeling Scheme for Dynamic Ordered XML Trees. In: Proceedings of the 20th ICDE Conference, pp. 66–78 (2004)
17. Hasan, K.M.A., Tsuji, T., Higuchi, K.: An Efficient Implementation for MOLAP Basic Data Structure and Its Evaluation. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 288–299. Springer, Heidelberg (2007)
18. Jin, D., Tsuji, T., Tsuchida, T., Higuchi, K.: An Incremental Maintenance Scheme of Data Cubes. In: Haritsa, J.R., Kotagiri, R., Pudi, V. (eds.) DASFAA 2008. LNCS, vol. 4947, pp. 172–187. Springer, Heidelberg (2008)
19. Tsuchida, T., Tsuji, T., Higuchi, K.: Implementing Vertical Splitting for Large Scale Multidimensional Datasets and Its Evaluations. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 208–223. Springer, Heidelberg (2011)
20. XMark – An XML Benchmark Project, http://monetdb.cwi.nl/xml/
21. Apache Lucene, http://lucene.apache.org/

On the Complexity of t-Closeness Anonymization and Related Problems*

Hongyu Liang[1] and Hao Yuan[2]

[1] Institute for Interdisciplinary Information Sciences (IIIS)
Tsinghua University, Beijing 100084, China
lianghy08@mails.tsinghua.edu.cn
[2] Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong
haoyuan@cityu.edu.hk

Abstract. An important issue in releasing individual data is to protect the sensitive information from being leaked and maliciously utilized. Famous privacy preserving principles that aim to ensure both data privacy and data integrity, such as k-anonymity and l-diversity, have been extensively studied both theoretically and empirically. Nonetheless, these widely-adopted principles are still insufficient to prevent attribute disclosure if the attacker has partial knowledge about the overall sensitive data distribution. The t-closeness principle has been proposed to fix this, which also has the benefit of supporting numerical sensitive attributes. However, in contrast to k-anonymity and l-diversity, the theoretical aspect of t-closeness has not yet been well investigated.

We initiate the first systematic theoretical study on the t-closeness principle under the commonly-used attribute suppression model. We prove that for every constant t such that $0 \leq t < 1$, it is NP-hard to find an optimal t-closeness generalization of a given table. The proof consists of several reductions each of which works for different values of t, which together cover the full range. To complement this negative result, we also provide exact and fixed-parameter algorithms. Finally, we answer some open questions regarding the complexity of k-anonymity and l-diversity left in the literature.

1 Introduction

Privacy-preserving data publication is an important and active topic in the database area. Nowadays many organizations need to publish microdata that contain certain information, e.g., medical condition, salary, or census data, of a collection of individuals, which are very useful for research and other purposes. Such microdata are usually released as a table, in which each record (i.e., row) corresponds to a particular individual and each column represents an attribute of the individuals. The released data usually contain *sensitive attributes*, such as Disease and Salary, which, once leaked to unauthorized

* This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174. The research of the second author was supported by the Research Grants Council of Hong Kong under grant 9041688 (CityU 124411).

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 331–345, 2013.

Table 1. The raw microdata table

	Quasi-identifiers			Sensitive
	Zipcode	Age	Education	Disease
1	98765	38	Bachelor	Viral Infection
2	98654	39	Doctorate	Heart Disease
3	98543	32	Master	Heart Disease
4	97654	65	Bachelor	Cancer
5	96689	45	Bachelor	Viral Infection
6	97427	33	Bachelor	Viral Infection
7	96552	54	Bachelor	Heart Disease
8	97017	69	Doctorate	Cancer
9	97023	55	Master	Cancer
10	97009	62	Bachelor	Cancer

Table 2. A 3-anonymous partition

	Quasi-identifiers			Sensitive
	Zipcode	Age	Education	Disease
1	98⋆⋆⋆	3⋆	⋆	Viral Infection
2	98⋆⋆⋆	3⋆	⋆	Heart Disease
3	98⋆⋆⋆	3⋆	⋆	Heart Disease
4	9⋆⋆⋆⋆	⋆⋆	Bachelor	Cancer
5	9⋆⋆⋆⋆	⋆⋆	Bachelor	Viral Infection
6	9⋆⋆⋆⋆	⋆⋆	Bachelor	Viral Infection
7	9⋆⋆⋆⋆	⋆⋆	Bachelor	Heart Disease
8	970⋆⋆	⋆⋆	⋆	Cancer
9	970⋆⋆	⋆⋆	⋆	Cancer
10	970⋆⋆	⋆⋆	⋆	Cancer

Table 3. A 2-diverse partition

	Quasi-identifiers			Sensitive
	Zipcode	Age	Education	Disease
1	98⋆⋆⋆	3⋆	⋆	Viral Infection
2	98⋆⋆⋆	3⋆	⋆	Heart Disease
3	9⋆⋆⋆⋆	⋆⋆	⋆	Heart Disease
5	9⋆⋆⋆⋆	⋆⋆	⋆	Viral Infection
8	9⋆⋆⋆⋆	⋆⋆	⋆	Cancer
9	9⋆⋆⋆⋆	⋆⋆	⋆	Cancer
4	97⋆⋆⋆	⋆⋆	Bachelor	Cancer
6	97⋆⋆⋆	⋆⋆	Bachelor	Viral Infection
7	9⋆⋆⋆⋆	⋆⋆	Bachelor	Heart Disease
10	9⋆⋆⋆⋆	⋆⋆	Bachelor	Cancer

parties, could be maliciously utilized and harm the individuals. Therefore, those features that can directly identify individuals, e.g., Name and Social Security Number, should be removed from the released table. See Table 1 for example of an (imagined) microdata table that a hospital prepares to release for medical research. (Note that the IDs in the first column are only for simplicity of reference, but not part of the table.)

Nonetheless, even with unique identifiers removed from the table, sensitive personal information can still be disclosed due to the *linking attacks* [27,28], which try to identify individuals from the combination of *quasi-identifiers*. The quasi-identifiers are those attributes that can reveal partial information of the individual, such as Gender, Age, and Hometown. For instance, consider an adversary who knows that one of the records in Table 1 corresponds to Bob. In addition he knows that Bob is around thirty years old and has a Master's Degree. Then he can easily identify the third record as Bob's and thus learns that Bob has a heart disease.

A widely-adopted approach for protecting privacy against such attacks is *generalization*, which partitions the records into disjoint groups and then transforms the quasi-identifier values in each group to the same form. (The sensitive attribute values are not generalized because they are usually the most important data for research.) Such

generalization needs to satisfy some *anonymization principles*, which are designed to guarantee data privacy to a certain extent.

The earliest (and probably most famous) anonymization principle is the k-*anonymity* principle proposed by Samarati [27] and Sweeney [28], which requires each group in the partition to have size at least k for some pre-specified value of k; such a partition is called k-*anonymous*. Intuitively, this principle ensures that every combination of quasi-identifier values appeared in the table is indistinguishable from at least $k-1$ other records, and hence protects the individuals from being uniquely recognized by linking attacks. The k-anonymity principle has been extensively studied, partly due to the simplicity of its statement. Table 2 is an example of a 3-anonymous partition of Table 1, which applies the commonly-used *suppression method* to generalize the values in the same group, i.e., *suppresses* the conflicting values with a new symbol '$\star$'.

A potential issue with the k-anonymity principle is that it is totally independent of the sensitive attribute values. This issue was formally raised by Machanavajjhala et al. [19] who showed that k-anonymity is insufficient to prevent disclosure of sensitive values against the *homogeneity attack*. For example, assume that an attacker knows that one record of Table 2 corresponds to Danny, who is an elder with a Doctorate Degree. From Table 2 he can easily conclude that Danny's record must belong to the third group, and hence knows Danny has a cancer since all people in the third group have the same disease. To forestall such attacks, Machanavajjhala et al. [19] proposed the l-*diversity* principle, which demands that at most a $1/l$ fraction of the records can have the same sensitive value in each group; such a partition is called l-*diverse*. Table 3 is an example of a 2-diverse partition of Table 1. (There are some other formulations of l-diversity, e.g., one requiring that each group comprises at least l different sensitive values.)

Li et al. [16] observed that the l-diversity principle is still insufficient to protect sensitive information disclosure against the *skewness attack*, in which the attacker has partial knowledge of the *overall* sensitive value distribution. Moreover, since l-diversity only cares whether two sensitive values are distinct or not, it fails to well support sensitive attributes with semantic similarities, such as numerical attributes (e.g., the salary).

To fix these drawbacks, Li et al. [16] introduced the t-*closeness* principle, which requires that the sensitive value distribution in any group differs from the overall sensitive value distribution by at most a threshold t. There is a metric space defined on the set of possible sensitive values, in which the maximum distance of two points (i.e., sensitive values) in the space is normalized to 1. The distance between two probability distributions of sensitive values are then measured by the *Earth-Mover Distance* (EMD) [26], which is widely used in many areas of computer science. Intuitively, the EMD measures the minimum amount of work needed to transform one probability distribution to another by means of moving distribution mass between points in the probability space. The EMD between two distributions in the (normalized) space is always between 0 and 1. We will give an example of a t-closeness partition of Table 1 for some threshold t later in Section 2, after the related notation and definitions are formally introduced.

The t-closeness principle has been widely acknowledged as an enhanced principle that fixes the main drawbacks of previous approaches like k-anonymity and l-diversity. There are also many other principles proposed to deal with different attacks or for use of ad-hoc applications; see, e.g., [3,20,22,23,29,30,32,33] and the references therein.

1.1 Theoretical Models of Anonymization

It is always assumed that the released table itself satisfies the considered principle (k-anonymity, l-diversity, or t-closeness), since otherwise there exists no feasible solution at all. Therefore, the trivial partition that puts all records in a single group always guarantees the principle to be met. However, such a solution is useless in real-world scenarios, since it will most probably produce a table full of '$\star$'s, which is undesirable in most applications. This extreme example demonstrates the importance of finding a balance between data privacy and *data integrity*.

Meyerson and Williams [21] proposed a framework for theoretically measuring the data integrity, which aims to find a partition (under certain constraints such as k-anonymity) that minimizes the number of suppressed cells (i.e., '$\star$'s) in the table. This model has been widely adopted for theoretical investigations of anonymization principles. Under this model, k-anonymity and l-diversity have been extensively studied; more detailed literature reviews will be given later.

However, in contrast to k-anonymity and l-diversity, the theoretical aspects of the t-closeness principle have not been well explored before. There are only a handful of algorithms designed for achieving t-closeness [16,17,8,25]. The algorithms given by Li et al. [16,17] incorporate t-closeness into k-anonymization frameworks (Incognito [14] and Mondrian [15]). Cao et al. [8] proposed the SABRE algorithm, which is the first framework tailored for t-closeness. The information-theoretic approach in [25] works for an "average" version of t-closeness. None of these algorithms is guaranteed to have good worst-case performance. Furthermore, to the best of our knowledge, no computational complexity results of t-closeness have been reported in the literature.

1.2 Our Contributions

In this paper, we initiate the first systematic theoretical study on the t-closeness principle under the commonly-used suppression framework. First, we prove that for every constant t such that $0 \leq t < 1$, it is NP-hard to find an optimal t-closeness generalization of a given table. Notice that the problem becomes trivial when $t = 1$, since the EMD between any two sensitive value distributions is at most 1, and hence putting each record in a distinct group provides a feasible solution that does not need to suppress any value at all, which is of course optimal. Our result shows that the problem immediately becomes hard even if the threshold is relaxed to, say, 0.999. At the other extreme, a 0-closeness partition demands that the sensitive value distribution in every group must be the same with the overall distribution. This seems to restrict the sets of feasible solutions in a very strong sense, and thus one might imagine whether there exists an efficient algorithm for this special case. Our result dashes this hope. The proof of our hardness result consists of several different reductions. Interestingly, each of these reductions only work for a set of special values of t, but altogether they cover the full range $[0, 1)$. We note that the hardness of t_1-closeness does not directly imply that of t_2-closeness for $t_1 \neq t_2$ since they may have very different optimal objective values.

As a by-product of our proof, we establish the NP-hardness of k-anonymity when $k = cn$, where n is the number of records and c is any constant in $(0, 1/2]$. To the best of our knowledge, this is the first hardness result for k-anonymity that works for $k = \Omega(n)$. The existing approaches for proving hardness of k-anonymity all fail to

generalize to this range of k due to inherent limits of the reductions. Note that $k = n/2$ is the largest possible value for which k-anonymity can be hard, since when $k > n/2$ any k-anonymous partition can only contain one group, namely the table itself.

To complement our negative results, we also provide exact and fixed-parameter algorithms for obtaining the optimal t-closeness partition. Our exact algorithm for t-closeness runs in time $2^{O(n)} \cdot O(m)$, where n and m are respectively the number of rows and columns in the input table. Together with a reduction that we derive (Lemma 1), this gives a $2^{O(n)} \cdot O(m)$ time algorithm for k-anonymity for *all* values of k, thus generalizing the result in [4] which only works for constant k. We then prove that the problem is fixed-parameter tractable when parameterized by m and the alphabet size of the input table. This implies that an optimal t-closeness partition can be found in polynomial time if the number of quasi-identifiers and that of distinct attribute values are both small (say, constants), which is true in many real-world applications. (Parameterized complexity has become a very active research area. For standard notation and definitions in parameterized complexity, we refer the reader to [10].) We obtain our fixed-parameter algorithm by reducing t-closeness to a special *mixed integer linear program* in which some variables are required to take integer values while others are not. The integer linear program we derived for characterizing t-closeness may have its own interest in future applications. Both of our algorithms work for all values of t.

Last but not least, we review the problems of finding optimal k-anonymous and l-diverse partitions, and answer two open questions left in the literature.

- We prove that the 2-diversity problem can be solved in polynomial time, which complements the NP-hardness results for $l \geq 3$ given in [31]. (We notice that the authors of [9] claimed that 2-diversity was proved to be polynomial by [31]. However what [31] actually proved is that the special 2-diversity instances in which there are only two distinct sensitive values can be solved in polynomial time. They do not have results for general 2-diversity. To the best of our knowledge, ours is the first work to demonstrate the tractability of 2-diversity.)
- We then present an m-approximation algorithm for k-anonymity that runs in polynomial time for all values of k. (Recall that m is the number of quasi-identifiers.) This improves the $O(k)$ and $O(\log k)$ ratios in [1,24] when k is relatively large compared to m. We note that the performance guarantee of their algorithms cannot be reduced even for small values of m, due to some intrinsic limitations (for example, [24] uses the tight $\Theta(\log k)$ approximation for k-set cover).

Finally, we note that our contributions in this paper are mainly towards building theoretical foundations of the privacy-preserving principles, which is desirable for the whole system to stand on. It is very interesting to see whether good implementations of our algorithms can be found that perform well on real databases.

1.3 Related Work

It is known that finding an optimal k-anonymous partition of a given table is NP-hard for every fixed integer $k \geq 3$ [21], while it can be solved optimally in polynomial time when $k \leq 2$ [4]. The NP-hardness result holds even for very restricted cases, e.g., when $k = 3$ and there are only three quasi-identifiers [5,6]. On the other hand, Blocki and Williams

[4] gave a $2^{O(n)} \cdot O(m)$ time algorithm that finds an optimal k-anonymous partition when $k = O(1)$, where n and m are the number of rows and columns of the input table respectively. They also showed this problem to be fixed-parameter tractable when m and $|\Sigma|$ (the alphabet size of the table) are parameters. The parameterized complexity of k-anonymity has also been studied in [6,7,11] with respect to other parameters.

Meyerson and Williams [21] gave an $O(k \log k)$ approximation algorithm for k-anonymity, i.e., it finds a k-anonymous partition in which the number of suppressed cells is at most $O(k \log k)$ times the optimum. The ratio was later improved to $O(k)$ by Aggarwal et al. [1] and to $O(\log k)$ by Park and Shim [24]. We note that the algorithms in [21,24] run in time $n^{O(k)}$, and hence are polynomial only if $k = O(1)$, while the algorithm in [1] runs in polynomial running time for all k. There are also a number of heuristic algorithms for k-anonymity (e.g., Incognito [14]), which work well in many real datasets but fail to give good worst-case performance guarantee.

Xiao et al. [31] are the first to establish a systematic theoretical study on l-diversity. They showed that finding an optimal l-diverse partition is NP-hard for every fixed integer $l \geq 3$ even if m, the number of quasi-identifiers, is any fixed integer not smaller than l. They also provided an $(l \cdot m)$-approximation algorithm. Dondi et al. [9] proved an inapproximability factor of $c \ln(l)$ for l-diversity where $c > 0$ is some constant, and showed that the problem remains APX-hard even if $l = 4$ and the table consists of only three columns. They also presented an m-approximation when the number of distinct sensitive values is constant, and gave parameterized hardness results and algorithms.

1.4 Paper Organization

The rest of this paper is organized as follows. Section 2 introduces notation and definitions used throughout the paper, and then formally defines the problems. Section 3 is devoted to proving the hardness of finding the optimal t-closeness partition, while Section 4 provides exact and parameterized algorithms. Sections 5 and 6 present our results for k-anonymity and 2-diversity, respectively. Finally, the paper is concluded in Section 7 with some discussions and future research directions.

Due to space limitations, we omit most rigorous proofs of our theorems. All the missing proofs can be found in the full version of this paper [18].

2 Preliminaries

We consider a raw database that contains m quasi-identifiers (QIs) and a sensitive attribute (SA).[1] Each record t in the database is an $(m+1)$-dimensional vector drawn from Σ^{m+1}, where Σ is the alphabet of possible values of the attributes. For $1 \leq i \leq m$, $t[i]$ is the value of the i-th QI of t, and $t[m+1]$ is the value of the SA of t. Let $\Sigma_s \subseteq \Sigma$ be the alphabet of possible SA values. A microdata table (or table, for short) $\mathcal{T}$ is a multiset of vectors (or rows) chosen from Σ^{m+1}, and we denote by $|\mathcal{T}|$ the size of $\mathcal{T}$, i.e., the number of vectors contained in $\mathcal{T}$. We will let $n = |\mathcal{T}|$ when the table $\mathcal{T}$ is clear in the context. Note that $\mathcal{T}$ may contain identical vectors since it can be a multiset.

[1] Following previous approaches, we only consider instances with one sensitive attribute. Our hardness result indicates that one SA already makes the problem NP-hard. Meanwhile, it is easy to verify that our algorithms also work for the multiple SA case.

Table 4. The first three records in Table 1

	Quasi-identifiers			Sensitive
	Zipcode	Age	Education	Disease
1	98765	38	Bachelor	Viral Infection
2	98654	39	Doctorate	Heart Disease
3	98543	32	Master	Heart Disease
After generalization:				
1	98⋆⋆⋆	3⋆	⋆	Viral Infection
2	98⋆⋆⋆	3⋆	⋆	Heart Disease
3	98⋆⋆⋆	3⋆	⋆	Heart Disease

Table 5. A 0.3-closeness partition

	Quasi-identifiers			Sensitive
	Zipcode	Age	Education	Disease
1	9⋆⋆⋆⋆	⋆⋆	⋆	Viral Infection
2	9⋆⋆⋆⋆	⋆⋆	⋆	Heart Disease
4	9⋆⋆⋆⋆	⋆⋆	⋆	Cancer
3	9⋆⋆⋆⋆	⋆⋆	⋆	Heart Disease
5	9⋆⋆⋆⋆	⋆⋆	⋆	Viral Infection
8	9⋆⋆⋆⋆	⋆⋆	⋆	Cancer
9	9⋆⋆⋆⋆	⋆⋆	⋆	Cancer
6	9⋆⋆⋆⋆	⋆⋆	Bachelor	Viral Infection
7	9⋆⋆⋆⋆	⋆⋆	Bachelor	Heart Disease
10	9⋆⋆⋆⋆	⋆⋆	Bachelor	Cancer

We also use $\mathcal{T}[j]$ to denote the j-th vector in $\mathcal{T}$ under some ordering, e.g., $\mathcal{T}[3][m+1]$ is the SA value of the third vector of $\mathcal{T}$. Let $\star$ be a fresh character not in Σ. For each vector $t \in \mathcal{T}$, let t^* be the *suppressor* of t (inside $\mathcal{T}$) defined as follows:

- $t^*[m+1] = t[m+1]$;
- for $1 \le i \le m$, $t^*[i] = t[i]$ if $t[i] = t'[i]$ for all $t' \in \mathcal{T}$, and $t^*[i] = \star$ otherwise.

The *cost* of a suppressor t^* is $cost(t^*) = |\{1 \le i \le m \mid t^*[i] = \star\}|$, i.e., the number of '$\star$'s in t^*. It is easy to see that all vectors in $\mathcal{T}$ have the same suppressor if we only consider the quasi-identifiers. The *generalization* of $\mathcal{T}$ is defined as $Gen(\mathcal{T}) = \{t^* \mid t \in \mathcal{T}\}$. (Note that $Gen(\mathcal{T})$ is also a multiset.) The *cost* of the generalization of $\mathcal{T}$ is $cost(\mathcal{T}) = \sum_{t^* \in Gen(\mathcal{T})} cost(t^*)$, i.e., the sum of costs of all the suppressors. Since all suppressors in $\mathcal{T}$ have the same cost, we can equivalently write $cost(\mathcal{T}) = |\mathcal{T}| \cdot cost(t^*)$ for any $t^* \in Gen(\mathcal{T})$.

As an illustrative example, Table 4 consists of the first three record of Table 1, which contains eight QIs (we regard each digit of Zip-code and Age as a separate QI) and one SA. The generalization of Table 4 is also shown. In this case all suppressors have cost 5, and the cost of this generalization is $5 \cdot 3 = 15$.

A *partition* $\mathcal{P}$ of table $\mathcal{T}$ is a collection of pairwise disjoint non-empty subsets of $\mathcal{T}$ whose union equals $\mathcal{T}$. Each subset in the partition is called a *group* or a *sub-table*. The cost of the partition $\mathcal{P}$, denoted by $cost(\mathcal{P})$, is the sum of costs of all its groups. For example, the partition of Table 1 given by Table 2 has cost $5 \cdot 3 + 6 \cdot 4 + 5 \cdot 3 = 54$.

2.1 t-Closeness Principle

We formally define the t-closeness principle introduced in [16] for protecting data privacy. Let $\mathcal{T}$ be a table, and assume without loss of generality that $\Sigma_s = \{1, 2, \ldots, |\Sigma_s|\}$. The *sensitive attribute value space* (SA space) is a normalized metric space (Σ_s, d), where $d(\cdot, \cdot)$ is a distance function defined on $\Sigma_s \times \Sigma_s$ satisfying that (1)$d(i, i) = 0$ for any $i \in \Sigma_s$; (2)$d(i, j) = d(j, i)$ for all $i, j \in \Sigma_s$; (3)$d(i, j) + d(j, k) \ge d(i, k)$ for $i, j, k \in \Sigma_s$ (this is called the triangle inequality); and (4)$\max_{i,j \in \Sigma_s} d(i, j) = 1$ (this is called the normalized condition).

For a sub-table $M \subseteq \mathcal{T}$ and $i \in \Sigma_s$, denote by $n(M, i)$ the number of vectors whose SA value equals i. Clearly $|M| = \sum_{i \in \Sigma_s} n(M, i)$. The *sensitive attribute value distribution* (SA distribution) of M, denoted by $\mathbf{P}(M)$, is a $|\Sigma_s|$-dimensional vector whose i-th coordinate is $\mathbf{P}(M)[i] = n(M, i)/|M|$ for $1 \leq i \leq |\Sigma_s|$. Thus $\mathbf{P}(M)$ can be seen as the probability distribution of the SA values in M, assuming that each vector in M appears with the same probability. For a threshold $0 \leq t \leq 1$, we say M have t-*closeness* (with $\mathcal{T}$) if $\mathrm{EMD}(\mathbf{P}(M), \mathbf{P}(\mathcal{T})) \leq t$, where $\mathrm{EMD}(\mathbf{X}, \mathbf{Y})$ is the *Earth-Mover Distance* (EMD) between distributions $\mathbf{X}$ and $\mathbf{Y}$ [26]. A t-*closeness partition* of $\mathcal{T}$ is one in which every group has t-closeness with $\mathcal{T}$.

Intuitively, the EMD measures the minimum amount of work needed to transform one probability distribution to another by means of moving distribution mass between points in the probability space; here a unit of work corresponds to moving a unit amount of probability mass by a unit of ground distance. The EMD between two SA distributions $\mathbf{X}$ and $\mathbf{Y}$ can be formally defined as the optimal objective value of the following linear program [26,16]:

$$\text{Minimize } \sum_{i=1}^{|\Sigma_s|}\sum_{j=1}^{|\Sigma_s|} d(i,j)f(i,j) \text{ subject to:}$$

$$\sum_{j=1}^{|\Sigma_s|} f(i,j) = \mathbf{X}[i],\ \forall 1 \leq i \leq |\Sigma_s| \text{ and } \sum_{i=1}^{|\Sigma_s|} f(i,j) = \mathbf{Y}[j], \forall 1 \leq j \leq |\Sigma_s|$$

$$f(i,j) \geq 0,\ \forall 1 \leq i, j \leq |\Sigma_s|.$$

The above constraints are a little different from those in [16]; however they can be proved equivalent using the triangle inequality condition of the SA space. It is also easy to see that $\mathrm{EMD}(\mathbf{X}, \mathbf{Y}) = \mathrm{EMD}(\mathbf{Y}, \mathbf{X})$. By the normalized condition of the SA space, we have $0 \leq \mathrm{EMD}(\mathbf{X}, \mathbf{Y}) \leq 1$ for any SA distributions $\mathbf{X}$ and $\mathbf{Y}$.

The *equal-distance space* refers to a special SA space in which each pair of distinct sensitive values have distance exactly 1. There is a concise formula for computing the EMD between two SA distributions in this space.

Fact 1 ([16]). *For any SA distributions* $\mathbf{X}$ *and* $\mathbf{Y}$ *in the equal-distance space, we have*

$$\mathrm{EMD}(\mathbf{X}, \mathbf{Y}) = \frac{1}{2}\sum_{i=1}^{|\Sigma_s|} |\mathbf{X}[i] - \mathbf{Y}[i]| = \sum_{1 \leq i \leq |\Sigma_s| : \mathbf{X}[i] \geq \mathbf{Y}[i]} (\mathbf{X}[i] - \mathbf{Y}[i]).$$

Therefore, in the equal-distance space, the EMD coincides with the *total variation distance* between two distributions.

Let us go back to Table 1 for an example. We let 1,2,and 3 denote the sensitive values "Viral Inspection", "Heart Disease", and "Cancer", respectively. Let the SA space be the equal-distance space. The SA distribution of the whole table is then $(0.3, 0.3, 0.4)$. Suppose we set the threshold $t = 0.3$. It can be verified that Table 3, although being a 2-diverse partition, is not a t-closeness partition of Table 1. In fact, the SA distribution of the first group is $(0.5, 0.5, 0)$, and hence the EMD between it and the overall distribution is 0.4. (This example also reflects some property of the skewness attack that l-diversity suffers from. If an attacker can locate the record of Alice in the first group of Table 3,

then he knows that Alice does not have a cancer. If he in addition knows that Alice comes from some district where people have a very low chance to have heart disease, then he would be confident that Alice has a viral infection.) We instead give a 0.3-closeness partition in Table 5. We can actually verify that it achieves 0.1-closeness.

Now we are ready to define the main problem studied in this paper.

Problem 1. Given an input table $\mathcal{T}$, an SA space (Σ_s, d), and a threshold $t \in [0, 1]$, the t-CLOSENESS problem asks to find a t-closeness partition of $\mathcal{T}$ with minimum cost.

Finally we review another two widely-used principles for privacy preserving, namely k-anonymity and l-diversity, and the combinatorial problems associated with them. A partition is called *k-anonymous* if all its groups have size at least k. A (sub-)table $\mathcal{M}$ is called *l-diverse* if at most $|\mathcal{M}|/l$ of the vectors in $\mathcal{M}$ have an identical SA value. A partition is called l-diverse if all its groups are l-diverse.

Problem 2. Let $\mathcal{T}$ be a table given as input. The k-ANONYMITY (l-DIVERSITY) problem requires to find a k-anonymous (l-diverse) partition of $\mathcal{T}$ with minimum cost.

3 NP-Hardness Results

In this section we study the complexity of the t-CLOSENESS problem. The problem is trivial if the given threshold is $t = 1$, since putting each vector in a distinct group produces a 1-closeness partition with cost 0, which is obviously optimal. Our main theorem stated below indicates that this is in fact the only easy case.

Theorem 1. *For any constant t such that $0 \leq t < 1$, t-CLOSENESS is NP-hard.*

We will prove Theorem 1 via several reductions each covering a particular range of t. We first present a result that relates t-CLOSENESS to k-ANONYMITY.

Lemma 1. *There is a polynomial-time reduction from k-ANONYMITY to t-CLOSENESS with equal-distance space and $t = 1 - k/n$.*

Proof. Let $\mathcal{T}$ be an input table of k-ANONYMITY. We properly change the SA values of vectors in T to ensure that all their SA values are distinct; this can be done because the SA values are irrelevant to the objective of the k-ANONYMITY problem. Assume w.l.o.g. that the SA values are $\{1, 2, \ldots, n\}$. Consider an instance of t-CLOSENESS with the same input table $\mathcal{T}$, in which $t = 1-k/n$ and the SA space is the equal-distance space. The SA distribution of $\mathcal{T}$ is $(1/n, 1/n, \ldots, 1/n)$. In the SA distribution of each size-r group $\mathcal{T}_r$, there are exactly r coordinates equal to $1/r$ and $n-r$ coordinates equal to 0. It is easy to see that $\mathrm{EMD}(\mathbf{P}(\mathcal{T}), \mathbf{P}(\mathcal{T}_r)) = (n - r)(1/n) = 1 - r/n$. Hence, a group has t-closeness if and only if it is of size at least k. Therefore, each k-anonymous partition of $\mathcal{T}$ is also a t-closeness partition, and vice versa. The lemma follows. □

By Lemma 1 we can directly deduce the NP-hardness of t-CLOSENESS when the threshold t is given as input, using e.g. the NP-hardness of 3-ANONYMITY [21]. However, to show hardness for constant t that is bounded away from 1, we need $k/n = \Omega(1)$ and thus $k = \Omega(n)$. Unfortunately, the existing hardness results for k-ANONYMITY

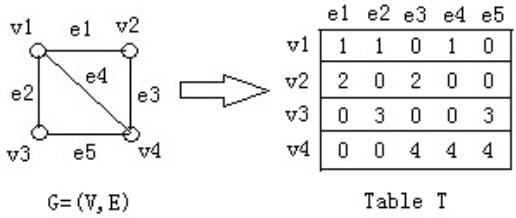


Fig. 1. An example of the reduction from MINBISECTION to $(n/2)$-ANONYMITY

only work for $k = O(1)$ and cannot be generalized to large values of k. For example, most hardness proofs use reductions from the k-dimensional matching problem, but this problem can be solved in polynomial time when $k = \Omega(n)$. Below we show the NP-hardness of k-ANONYMITY for $k = \Omega(n)$ via different approaches.

Theorem 2. *For any constant c such that* $0 < c \leq 1/2$, (cn)-ANONYMITY *is NP-hard.*

To the best of our knowledge, Theorem 2 is the first hardness result for k-ANONYMITY when $k = \Omega(n)$. We note that the constant $1/2$ is the best possible, since for any $k > n/2$, a k-anonymous partition can only contain one group, namely the table itself. We first prove the following result as a starting point in further reductions.

Theorem 3. $(n/2)$-ANONYMITY *is NP-hard.*

Proof. We will present a polynomial-time reduction from the minimum graph bisection (MINBISECTION) problem to $(n/2)$-ANONYMITY. MINBISECTION is a well-known NP-hard problem [12,13] defined as follows: given an undirected graph, find a partition of its vertices into two equal-sized halves so as to minimize the number of edges with exactly one endpoint in each half.

Let $G = (V, E)$ be an input graph of MINBISECTION, where $|V| \geq 4$ is even. Suppose $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$. In what follows we construct a table $\mathcal{T}$ of size $n = |V|$ that contains $m = |E|$ quasi-identifiers. (The sensitive attributes are useless in k-ANONYMITY so they will not appear.) This table will serve as the input to the k-ANONYMITY problem with $k = n/2$. Intuitively each row (or vector) of $\mathcal{T}$ corresponds to a vertex in V, while each column (or QI) of $\mathcal{T}$ corresponds to an edge in E. The alphabet Σ is $\{1, 2, \ldots, n\}$. For each $i \in [n]$ and $j \in [m]$,[2] let $\mathcal{T}[i][j] = i$ if $v_i \in e_j$, and $\mathcal{T}[i][j] = 0$ if $v_i \notin e_j$. Thus each column contains exactly two non-zero elements corresponding to the two endpoints of the associated edge. See Figure 1 for a toy example. Obviously $\mathcal{T}$ can be constructed in polynomial time.

Before delving into the reduction, we first prove a result concerning the partition cost of $\mathcal{T}$. Any $(n/2)$-anonymous partition of $\mathcal{T}$ contains at most two groups. For the trivial partition that only contains $\mathcal{T}$ itself, the cost is $n \cdot m$ because all elements in $\mathcal{T}$ should be suppressed. Thus an $(n/2)$-anonymous partition with minimum cost should consist of exactly two groups. Suppose $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2\}$ is an $(n/2)$-anonymous partition of $\mathcal{T}$ where $|\mathcal{T}_1| = |\mathcal{T}_2| = n/2$. Let $\{V_1, V_2\}$ be the corresponding partition of V (recall that each vector in $\mathcal{T}$ corresponds to a vertex in V). Consider $Gen(\mathcal{T}_1)$, the generalization

[2] We use $[q]$ to interchangeably denote $\{1, 2, \ldots, q\}$.

of $\mathcal{T}_1$. For any column $j \in [m]$, if some endpoint of e_j, say v_i, belongs to V_1, then $\mathcal{T}[i][j] = i$. By our construction of $\mathcal{T}$, any other element in the j-th column does not equal to i. Since $|\mathcal{T}_1| \geq n/2 \geq 2$, column j of $\mathcal{T}_1$ must be suppressed to $\star$. On the other hand, if none of e_j's endpoints belongs to V_1, then column j of $\mathcal{T}_1$ contains only zeros and thus can stay unsuppressed. Therefore, we obtain

$$cost(\mathcal{T}_1) = |\mathcal{T}_1| \cdot (|E_{11}| + |E_{12}|) = n(|E_{11}| + |E_{12}|)/2, \tag{1}$$

where E_{pq} denotes the set of edges with one endpoint in V_p and another in V_q, for $p, q \in \{1, 2\}$. Similarly we have $cost(\mathcal{T}_2) = n(|E_{22}| + |E_{12}|)/2$. Hence,

$$cost(\mathcal{P}) = \sum_{p=1}^{2} cost(\mathcal{T}_p) = n(|E| + |E_{12}|)/2, \tag{2}$$

noting that $|E| = |E_{11}| + |E_{12}| + |E_{22}|$.

We now prove the correctness of the reduction. Let OPT be the minimum size of any cut $\{V_1, V_2\}$ of G with $|V_1| = |V_2|$, and OPT' be the minimum cost of any $(n/2)$-anonymous partition of $\mathcal{T}$. We prove that $OPT' = n(|E| + OPT)/2$, which will complete the reduction from MINBISECTION to $(n/2)$-ANONYMITY. Let $\{V_1, V_2\}$ be the cut of G achieving the optimal cut size OPT, where $|V_1| = |V_2| = n/2$. Using notation introduced before, we have $OPT = |E_{12}|$. Let $\mathcal{P} = \{\mathcal{T}_1, \mathcal{T}_2\}$ where $\mathcal{T}_p = \{\mathcal{T}[i] \mid v_i \in V_p\}$ for $p \in \{1, 2\}$. Clearly $\mathcal{P}$ is an $(n/2)$-anonymous partition of $\mathcal{T}$. By Equation (2) we have $OPT' \leq cost(\mathcal{P}) = n(|E| + OPT)/2$.

On the other hand, let $\mathcal{P}' = \{\mathcal{T}'_1, \mathcal{T}'_2\}$ be an $(n/2)$-anonymous partition with $cost(\mathcal{P}') = OPT'$. We have $|\mathcal{T}'_1| = |\mathcal{T}'_2| = n/2$. Consider the partition $\{V'_1, V'_2\}$ of V with $V'_p = \{v_i \mid \mathcal{T}[i] \in \mathcal{T}'_p\}$ for $p \in \{1, 2\}$. Since $|V'_1| = |V'_2| = n/2$, we have $OPT \leq |E'_{12}|$ where E'_{12} denotes the set of edges with one endpoint in V'_1 and another in V'_2. By Equation (2) we have $OPT' = n(|E| + |E'_{12}|)/2 \geq n(|E| + OPT)/2$. Combined with the previously obtained inequality $OPT' \leq n(|E| + OPT)/2$, we have shown that $OPT' = n(|E| + OPT)/2$. By the analyses we also know that an optimal $(n/2)$-anonymous partition of $\mathcal{T}$ can easily be transformed to an optimal equal-sized cut of G. This finishes the reduction from MINBISECTION to $(n/2)$-ANONYMITY, and completes the proof of Theorem 3. □

Using Theorem 3 we can design a reduction to show the NP-hardness of (cn)-ANONYMITY when $0 < c \leq 1/3$. Roughly speaking, the idea is to add a large subgraph that is too expensive to split and therefore the optimal bisection of the old graph used in the proof of Theorem 3 does not change too much. This can be done only if c is small since the number of nodes in an old group is required to be more than c times the final size. We can find that $c = 1/3$ is the largest possible value for this reduction to work. Thus, the hardness of (cn)-ANONYMITY for $c \in (0, 1/3] \cup \{1/2\}$ is established. For the remaining case $c \in (1/3, 1/2)$, we need a totally different reduction, starting from a variant of the famous maximum clique problem [12]. The proof is harder and more tedious than previous ones. Due to space limitations, the proofs of these two reductions are omitted and can be found in the full version of this paper [18].

Combining Theorem 2 with Lemma 1, we obtain:

Corollary 1. *For any constant t such that $1/2 \leq t < 1$, t-CLOSENESS is NP-hard even with equal-distance space.*

The proof for the remaining case $0 \leq t < 1/2$ is done by two reductions from the 3-dimensional matching problem [12]. We only present the theorem due to space constraints, and the proof can be found in the full version of this paper [18].

Theorem 4. *For any constant $t \in [0, 1/2)$, t-CLOSENESS is NP-hard even if $|\Sigma_s| = 4$.*

4 Exact and Fixed-Parameter Algorithms

In this section we design exact algorithms for solving t-CLOSENESS. Notice that the size of an instance of t-CLOSENESS is polynomial in n and $m + 1$. The brute-force approach that examines each possible partition of the table to find the optimal solution takes $n^{O(n)}m^{O(1)} = 2^{O(n \log n)}m^{O(1)}$ time. We first improve this bound to single exponential in n. (Note that it cannot be improved to polynomial unless P = NP.) The proof can be found in the full version of this paper [18].

Theorem 5. *The t-CLOSENESS problem can be solved in $2^{O(n)} \cdot O(m)$ time.*

In many real applications, there are usually only a small number of attributes and distinct attribute values. Thus it is interesting to see whether t-CLOSENESS can be solved more efficiently when m and $|\Sigma|$ is small. We answer this question affirmatively in terms of fixed-parameter tractability.

Theorem 6. *t-CLOSENESS is fixed-parameter tractable when parameterized by m and $|\Sigma|$. Thus we can solve t-CLOSENESS optimally in polynomial time when m and $|\Sigma|$ are constants.*

We prove this theorem by reducing the task to an mixed integer linear program that precisely characterizes the objective value of our problem. The details can be found in the full version of this paper [18].

5 Approximation Algorithm for k-Anonymity

In this section we give a polynomial-time m-approximation algorithm for k-ANONYMITY, which improves the previous best ratio $O(k)$ [1] and $O(\log k)$ [24] when k is relatively large compared with m. Compared to theirs, our algorithm is very easy to implement.

Theorem 7. *k-ANONYMITY can be approximated within factor m in polynomial time.*

Proof. Consider a table $\mathcal{T}$ with n rows and m QI columns. Denote by OPT the minimum cost of any k-anonymous partition of $\mathcal{T}$. Partition $\mathcal{T}$ into "equivalence classes" $C_1, \ldots, C_R$ in the following sense: any two vectors in the same class are identical, i.e., they have the same value on each attribute, while any two vectors from different classes differ on at least one attribute. Assume $|C_1| \leq |C_2| \leq \ldots \leq |C_R|$. If $|C_1| \geq k$, then these classes form a k-anonymous partition with cost 0, which is surely optimal. Thus we assume $|C_1| < k$, and let $L \in [R]$ be the maximum integer for which $|C_L| < k$. Then $|C_{L'}| \geq k$ for all $L < L' \leq R$. It is clear that each vector in $C_1 \cup \ldots \cup C_L$ contributes at least one to the cost of any partition of $\mathcal{T}$. Thus $OPT \geq \sum_{i=1}^{L} |C_i|$.

Case 1: $\sum_{i=1}^{L} |C_i| \geq k$. In this case we partition $\mathcal{T}$ into $R - L + 1$ groups: $\{C_1 \cup \ldots \cup C_L, C_{L+1}, C_{L+2}, \ldots, C_R\}$. This is a k-anonymous partition of cost at most $m \cdot \sum_{i=1}^{L} |C_i| \leq m \cdot OPT$.

Case 2: $\sum_{i=1}^{L} |C_i| < k$ and $\sum_{i=1}^{L} |C_i| + \sum_{i=L+1}^{R} (|C_i| - k) \geq k$. We choose $C_i' \subseteq C_i$ for $L+1 \leq i \leq R$ satisfying that $|C_i \setminus C_i'| \geq k$ and $\sum_{i=1}^{L} |C_i| + \sum_{i=L+1}^{R} |C_i'| = k$. This can be done because of the second condition of this case. We partition $\mathcal{T}$ into $R-L+1$ groups: $\{\bigcup_{i=1}^{L} C_i \cup \bigcup_{i=L+1}^{R} C_i', C_{L+1} \setminus C_{L+1}', \ldots, C_R \setminus C_R'\}$. This is a k-anonymous partition of cost at most $m \cdot k \leq m \cdot OPT$, since $OPT \geq k$.

Case 3: $\sum_{i=1}^{L} |C_i| + \sum_{i=L+1}^{R} (|C_i| - k) < k$. We claim that there exists $i \in \{L + 1, \ldots, R\}$ such that any vector in C_i contributes at least one to the cost of any k-anonymous partition. Assume the contrary. Then there exists a k-anonymous partition such that, for every $L + 1 \leq i \leq R$, there is a vector $v \in C_i$ whose suppression cost is 0, which means that v belongs to a group that only contains vectors in C_i; denote this group by C_i'. We also know that there is at least one group in the partition that has positive cost. However, by removing all C_i', $L + 1 \leq i \leq R$, from $\mathcal{T}$, the number of vectors left is at most $n - k(R - L) = \sum_{i=1}^{R} |C_i| - k(R - L) = \sum_{i=1}^{L} |C_i| + \sum_{i=L+1}^{R} (|C_i| - k) < k$, due to the condition of this case. This contradicts with the property of k-anonymous partitions. Therefore the claim holds, i.e., there exists $j \in \{L + 1, \ldots, R\}$ such that any vector in C_j contributes at least one to the partition cost. Thus we have $OPT \geq \sum_{i=1}^{L} |C_i| + |C_j| \geq \sum_{i=1}^{L+1} |C_i|$. We partition $\mathcal{T}$ into $R - L$ groups: $\{\bigcup_{i=1}^{L+1} C_i, C_{L+2}, \ldots, C_R\}$. This is a k-anonymous partition with cost at most $m \cdot \sum_{i=1}^{L+1} |C_i| \leq m \cdot OPT$.

By the above case analyses, we can always find in polynomial time a k-anonymous partition of $\mathcal{T}$ with cost at most $m \cdot OPT$. This completes the proof of Theorem 7. □

We note that Theorem 7 implies that k-ANONYMITY can be solved optimally in polynomial time when $m = 1$. This is in contrast to l-DIVERSITY, which remains NP-hard when $m = 1$ (with unbounded l) [9].

6 Algorithm for 2-Diversity

In this part we give the first polynomial time algorithm for solving 2-DIVERSITY. Let $\mathcal{T}$ be an input table of 2-DIVERSITY. The following lemma is crucial to our algorithm, whose proof can be found in the full version of this paper [18].

Lemma 2. *There is an optimal 2-diverse partition of $\mathcal{T}$ in which every group consists of 2 or 3 vectors with distinct SA values.*

We reduce 2-DIVERSITY to a combinatorial problem called SIMPLEX MATCHING introduced in [2], which admits a polynomial algorithm [2]. The input of SIMPLEX MATCHING is a hypergraph $H = (V, E)$ containing edges of sizes 2 and 3 with non-negative edge costs $c(e)$ for all edges $e \in E$. In addition H is guaranteed to satisfy the following *simplex condition*: if $\{v_1, v_2, v_3\} \in E$, then $\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}$ are also in E, and $c(\{v_1, v_2\}) + c(\{v_2, v_3\}) + c(\{v_1, v_3\}) \leq 2 \cdot c(\{v_1, v_2, v_3\})$. The goal

is to find a perfect matching of H (i.e., a set of edges that cover every vertex $v \in V$ exactly once) with minimum cost (which is the sum of costs of all chosen edges).

Let $\mathcal{T}$ be an input table of 2-DIVERSITY. We construct a hypergraph $H = (V, E)$ as follows. Let $V = \{v_1, v_2, \ldots, v_n\}$ where v_i corresponds to the vector $\mathcal{T}[i]$. For every two vectors $\mathcal{T}[i], \mathcal{T}[j]$ (or three vectors $\mathcal{T}[i], \mathcal{T}[j], \mathcal{T}[k]$) with distinct SA values, there is an edge $e = \{v_i, v_j\}$ (or $e = \{v_i, v_j, v_k\}$) with cost equal to $cost(\{\mathcal{T}[i], \mathcal{T}[j]\})$ (or $cost(\{\mathcal{T}[i], \mathcal{T}[j], \mathcal{T}[k]\})$). Consider any 3D edge $e = \{v_i, v_j, v_k\}$. Since each column that needs to be suppressed in $\{\mathcal{T}[i], \mathcal{T}[j]\}$ must also be suppressed in $\{\mathcal{T}[i], \mathcal{T}[j], \mathcal{T}[k]\}$, we have $c(e)/3 \geq c(\{v_i, v_j\})/2$. Similarly, $c(e)/3 \geq c(\{v_i, v_k\})/2$ and $c(e)/3 \geq c(\{v_j, v_k\})/2$. Summing the inequalities up gives $2c(e) \geq c(\{v_i, v_j\}) + c(\{v_i, v_k\}) + c(\{v_j, v_k\})$. Therefore H satisfies the simplex condition, and it clearly can be constructed in polynomial time. Call a 2-diverse partition of $\mathcal{T}$ *good* if every group in it consists of 2 or 3 vectors with distinct SA values. Lemma 2 shows that there is an optimal 2-diverse partition that is good. By the construction of H, each good 2-diverse partition of $\mathcal{T}$ can be easily transformed to a perfect matching of H with the same cost, and vice versa. Hence, we can find an optimal 2-diverse partition of $\mathcal{T}$ by using the polynomial time algorithm for SIMPLEX MATCHING [2]. We thus have:

Theorem 8. *2-DIVERSITY is solvable in polynomial time.*

7 Conclusions

This paper presents the first theoretical study on the t-closeness principle for privacy preserving. We prove the NP-hardness of the t-CLOSENESS problem for every constant $t \in [0, 1)$, and give exact and fixed-parameter algorithms for the problem. We also provide conditionally improved approximation algorithm for k-ANONYMITY, and give the first polynomial time exact algorithm for 2-DIVERSITY. There are still many related problems that deserve further explorations, amongst which the most interesting one to the authors is designing polynomial time approximation algorithms for t-CLOSENESS with provable performance guarantees. The parameterized complexity of t-CLOSENESS with respect to other sets of parameters (see, e.g., [6,7,11]) are also of interest.

References

1. Aggarwal, G., Feder, T., Motwani, K.K.R., Panigrahy, R., Thomas, D., Zhu, A.: Anonymizing tables. In: ICDT, pp. 246–258 (2005)
2. Anshelevich, E., Karagiozova, A.: Terminal backup, 3D matching, and covering cubic graphs. SIAM Journal on Computing 40(3), 678–708 (2011)
3. Baig, M.M., Li, J., Liu, J., Wang, H.: Cloning for privacy protection in multiple independent data publications. In: CIKM, pp. 885–894 (2011)
4. Blocki, J., Williams, R.: Resolving the complexity of some data privacy problems. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part II. LNCS, vol. 6199, pp. 393–404. Springer, Heidelberg (2010)
5. Bonizzoni, P., Vedova, G.D., Dondi, R.: Anonymizing binary and small tables is hard to approximate. Journal of Combinatorial Optimization 22(1), 97–119 (2011)
6. Bonizzoni, P., Vedova, G.D., Dondi, R., Pirola, Y.: Parameterized complexity of k-anonymity: hardness and tractability. Journal of Combinatorial Optimization (in press)

7. Bredereck, R., Nichterlein, A., Niedermeier, R., Philip, G.: The effect of homogeneity on the complexity of k-anonymity. In: Owe, O., Steffen, M., Telle, J.A. (eds.) FCT 2011. LNCS, vol. 6914, pp. 53–64. Springer, Heidelberg (2011)
8. Cao, J., Karras, P., Kalnis, P., Tan, K.-L.: SABRE: a sensitive attribute bucketization and redistribution framework for t-closeness. The VLDB Journal 20(1), 59–81 (2011)
9. Dondi, R., Mauri, G., Zoppis, I.: The l-diversity problem: Tractability and approximability. Theoretical Computer Science (2012) (in press), doi:10.1016/j.tcs.2012.05.024
10. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1999)
11. Evans, P.A., Wareham, T., Chaytor, R.: Fixed-parameter tractability of anonymizing data by suppressing entries. Journal of Combinatorial Optimization 18(4), 362–375 (2009)
12. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
13. Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified NP-complete problems. In: STOC, pp. 47–63 (1974)
14. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: SIGMOD, pp. 49–60 (2005)
15. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: ICDE (2006)
16. Li, N., Li, T., Venkatasubramanian, S.: t-Closeness: Privacy beyond k-Anonymity and l-Diversity. In: ICDE, pp. 106–115 (2007)
17. Li, N., Li, T., Venkatasubramanian, S.: Closeness: A new privacy measure for data publishing. IEEE Transactions on Knowledge and Data Engineering 22(7), 943–956 (2010)
18. Liang, H., Yuan, H.: On the complexity of t-closeness anonymization and related problems. Technical report (2012), `http://arxiv.org/abs/1301.1751`
19. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data (TKDD) 1(1) (2007)
20. Martin, D.J., Kifer, D., Machanavajjhala, A., Gehrke, J., Halpern, J.Y.: Worst-case background knowledge for privacy preserving data publishing. In: ICDE, pp. 126–135 (2007)
21. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: PODS (2004)
22. Mohammed, N., Chen, R., Fung, B.C.M., Yu, P.S.: Differentially private data release for data mining. In: KDD (2011)
23. Nergiz, M.E., Atzori, M., Clifton, C.: Hiding the presence of individuals from shared databases. In: SIGMOD, pp. 665–676 (2007)
24. Park, H., Shim, K.: Approximate algorithms for k-anonymity. In: SIGMOD (2007)
25. Rebollo-Monedero, D., Forné, J., Domingo-Ferrer, J.: From t-closeness-like privacy to postrandomization via information theory. IEEE Transactions on Knowledge and Data Engineering 22(11), 1623–1636 (2010)
26. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision 40(2), 99–121 (2000)
27. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)
28. Sweeney, L.: k-anonymity: a model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5), 557–570 (2002)
29. Xiao, X., Tao, Y.: Personalized privacy preservation. In: SIGMOD, pp. 229–240 (2006)
30. Xiao, X., Tao, Y.: m-invariance: Towards privacy preserving re-publication of dynamic datasets. In: SIGMOD, pp. 689–700 (2007)
31. Xiao, X., Yi, K., Tao, Y.: The hardness and approximation algorithms for l-diversity. In: EDBT, pp. 135–146 (2010)
32. Xue, M., Karras, P., Raissi, C., Pung, H.K.: Utility-driven anonymization in data publishing. In: CIKM, pp. 2277–2280 (2011)
33. Zhang, Q., Koudas, N., Srivastava, D., Yu, T.: Aggregate query answering on anonymized tables. In: ICDE, pp. 116–125 (2007)

Distributed Anonymization for Multiple Data Providers in a Cloud System

Xiaofeng Ding[1,2], Qing Yu[2], Jiuyong Li[1], Jixue Liu[1], and Hai Jin[2]

[1] School of Information Technology & Mathematical Sciences
University of South Australia, Adelaide
{xiaofeng.ding,jiuyong.li,jixue.liu}@unisa.edu.au
[2] SCTS & CGCL, School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, China
{xfding,arik,hjin}@hust.edu.cn

Abstract. With the proliferation of cloud computing, there is an increasing need for sharing data repositories containing personal information across multiple distributed databases, and such data sharing is subject to different privacy constraints of multiple individuals. Most of the existing methods focus on single database anonymization, although the concept of distributed anonymization was discussed in some literatures, it only provides an uniform approach that exerts the same amount of preservation for all data providers, without catering for user's specific privacy requirements. The consequence is that we may offer insufficient protection to a subset of people, while applying excessive privacy budget to the others. Motivated by this, we present a new distributed anonymization protocol based on the concept of personalized privacy preservation. Our technique performs a personalized anonymization to satisfy multiple data provider's privacy requirements, and then publish their global anonymization view without any privacy breaches. Extensive experiments have been conducted to verify that our proposed protocol and anonymization method are efficient and effective.

1 Introduction

Cloud computing is a long dreamed vision of computing as a utility, where cloud consumers can remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources[1]. Third-party computing clouds, such as Amazon's EC^2 and Microsoft's Azure, provide support for computation, data management, and internet services. For its great flexibility and economic savings, more and more data providers outsource their data to the cloud platform. Government and organizations increasingly recognize the critical value and opportunities in sharing such a wealth of information across multiple distributed databases. Examples of success cases on EC^2 include Nimbus Health [2], which manages patient medical records, and ShareThis [3], a social content-sharing network that has shared 430 million items across 30,000 web sites. Unfortunately, such data sharing is subject

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 346–360, 2013.

to constraints imposed by privacy of individuals, which is consistent with related works on cloud security [4][6][7][8]. In fact, researchers have showed that attackers could effectively target and observe information from specific cloud instances on third party clouds [9]. To protect data privacy, the sensitive information of individuals should be preserved before outsourcing to the commercial public cloud [10]. Partition-based privacy preserving data publishing technique resolves this problem by anonymizing the original data. It has been widely discussed in the literature such as *k-anonymity* [11][19], *(a,k)-anonymity* [12], *l-diversity* [13], *t-closeness* [30], *m-invarance* [14], etc. Given that many data providers need to release their original data to the cloud and each of them has a different need for privacy requirement, the problem becomes a challenging task.

In the most related literature, privacy preserving data publishing for a single dataset has been extensively studied. A lot of work contributes to algorithms that transform a dataset to meet a certain privacy principle by using techniques such as generalization, suppression, perturbation, and swapping [15][16][17][18][21]. Although Xiong et al. [5] studied a problem of data anonymization for horizontally partitioned datasets and proposed a distributed anonymization protocol, it only gave a uniform approach that exerts the same level of protection for all data providers, without catering for their d needs. How to design a new distributed anonymization protocol over cloud servers, which can satisfy different data provider's personal needs and maximize the utility of the anonymous data, still remains a challenging problem.

In this paper, we design an algorithm which inserts data objects into an R-tree for anonymization on top of the k-anonymity and l-diversity principle. For the first time, we propose a new distributed anonymization protocol that allows multiple data providers to publish datasets with personalized needs to cloud servers for building a virtual anonymized database, which is in turn based on the integration of all the local generalized data. As the output of the protocol, each private dataset produces a local anonymized dataset that satisfies each data provider's privacy constraints and their union forms a global virtual database that meets a global anonymization principle.

The remainder of this paper is organized as follows. Section 2 briefly reviews works related to our research. Section 3 introduces the problem definition and discusses the privacy model we used. Section 4 presents our designed distributed annymization protocol, and Section 5 gives a generalization algorithm to achieve the protocol, Section 6 presents a set of experimental evaluations and Section 7 concludes the paper with future work.

2 Related Work

Our work is inspired by work in a number of research areas. We briefly review the work in closely related areas below and discuss how our work leverages and advances the current state-of-the-art techniques.

Privacy Preserving Data Publishing. Privacy preserving data publishing for centralized databases has been studied extensively in recent years [25].

One aspect of work aims at devising privacy principles, such as *k-anonymity* [11], *(a,k)-anonymity* [12], *l-diversity* [13], *t-closeness* [30], and *m-invarance* [14], that designed as criteria for judging whether a published dataset provides a certain privacy preservation. Another large part of work contributes to proposing algorithms that transform a dataset to meet one of the above privacy principles. In this study, our distributed anonymization protocol is built on top of the k-anonymity and l-diversity principles. We propose a new anonymization algorithm by inserting all the data objects into an R-tree to achieve high quality generalization.

Distributed Anonymization Solutions. There are a number of potential approaches can be applied to enable data anonymization for distributed databases. One naive solution is for each data provider to implement data anonymization independently. Data analysts or users can then query the individual anonymized database or an integrated view over all of them. Since the data is anonymized before integration, thus the main drawback of this solution is that it will cause low data utility. An alternative approach assumes the existence of a third party that can be trusted by all data providers. In this case, data providers upload their data to the trusted third party where data integration and anonymization are performed, clients then can query the centralized database in the third party. However, finding such a trusted third party is not always feasible. Compromise of the server by attackers could lead to a complete privacy loss for all the participating parties and data subjects [5].

Jiang et al. [26] presented a two-party framework along with an application that generates k-anonymous data from two vertically partitioned sources without disclosing data from one site to the other. Zhong et al. [27] proposed provably private solutions for k-anonymous generalization in the distributed scenario by maintaining end-to-end privacy from the original customer data to the final k-anonymous results. Xiong et al. [5] presented a distributed anonymization protocol aims to achieve anonymity for both data subjects and data providers.

In contrast to the above work, our work is aimed at outsourcing data provider's private dataset to cloud servers for data sharing. More importantly, our anonymization protocol aims at satisfying each data provider's individual need of privacy budget and producing a virtual database based on all the local anonymized datasets. We propose a distributed data anonymization approach that enables data providers to publish their private data into cloud servers. It is important to note that the anonymous data still resides at individual databases in servers. Integration of all local anonymized datasets is performed through the secure distributed protocols. In this case, each individual database can execute the query on its local data, and then engage in distributed protocols to assemble the results that are guaranteed to be k-anonymous.

3 Problem Statement

3.1 Preliminaries and Problem Definition

We assume that the microdata is distributed stored among s ($s > 2$) sites (nodes), and each site owns a private local database d_i ($1 \leq i \leq s$) published

Site 1

Age	Zip Code	Disease
20 − 30	5000-5100	AIDS
20 − 30	5000-5100	Flu
20 − 30	5000-5100	Cold
40 − 50	5000-5100	Cold
40 − 50	5000-5100	Flu

Site 2

Age	Zip Code	Disease
20 − 30	5000-5100	Cold
20 − 30	5000-5100	Flu
20 − 30	5000-5100	AIDS
40 − 50	5000-5100	Flu
40 − 50	5000-5100	Cold
40 − 50	5000-5100	Flu

Site 3

Age	Zip Code	Disease
30 − 40	5200-5300	AIDS
30 − 40	5200-5300	Flu
30 − 40	5200-5300	Cold
30 − 40	5200-5300	Flu
30 − 40	5200-5300	Cold

Fig. 1. Different anonymous tables within 3 sites

by one of the data providers. The union of all the local databases, denoted as microdata set D as given in Definition 1, gives a complete view of all the data ($D = \cup d_i$). In particular, the *quasi-identifier* attributes of each record within each local database are the same. To preserve the privacy of data, our work based on the rationale of k-anonymity to achieve anonymity for data subjects and this algorithm can be easily extended to support l-diversity [13]. Following a distributed anonymization protocol, each site produces a local anonymized database d_i^* that meets its own privacy principle k_i since data providers have different privacy requirements for publishing. The union of all local anonymized databases forms a virtual database that is guaranteed to meet a global privacy requirement K. Note that $max(k_1, k_2, k_3, ..., k_s) \leq K$. As illustrated in Fig.1, the data is partitioned horizontally among 3 sites. The anonymized dataset d_1^* of d_1 is required to be 2-anonymity, d_2^* of d_2 is required to be 3-anonymity, and d_3^* of d_3 is required to be 5-anonymity. The global anonymized database $D^* = (d_1 \cup d_2 \cup d_3)^*$ is required to be 5-anonymity. Note that, simply combine all anonymized local datasets cannot always satisfy the global privacy budget K, that is to say, $(d_1 \cup d_2 \cup d_3)^* = (d_1^* \cup d_2^* \cup d_3^*)$ does not always hold.

Definition 1 (Microdata Set). *A microdata set D (or table T) is of the following form: $D = \{r_1, r_2, ..., r_n\}$, where $r_1, r_2, ..., r_n$ are n distinct records, and r_i $(1 \leq i \leq n)$ represents the information of an individual with identifier attribute $A_{id} = \{A_0\}$, quasi-identifier (QI) attributes $A_{QI} = \{A_1, A_2, ..., A_d\}$ and sensitive attributes $A_S = \{A_{d+1}\}$.*

Definition 2 (Anonymized Dataset). *An anonymized dataset D^* is of the following form: $D^* = \{G_1, G_2, ..., G_m\}$, where $G_1, G_2, ..., G_m$ are m disjoint groups, denoted as Equivalence Class or QI-group, and G_i $(1 \leq i \leq m)$ represents the information of a group of individuals with the same generalized quasi-identifier attributes $A_{QI}^* = \{A_1^*, A_2^*, ..., A_d^*\}$ and all occurrences of sensitive attribute $\{A_{d+1}\}$ of individuals within group G_i.*

In general, given a microdata set $D = \cup d_i$ $(1 \leq i \leq s)$ and the privacy budget k_i, K, our objective is to generate an anonymized data set D^* such that (1) d_i^* satisfies k_i-anonymity, (2) D^* satisfies K-anonymity, and (3) the anonymized dataset preserves as much detailed information as possible for query analysis.

3.2 Privacy Preservation Goals

We use k-anonymity and l-diversity as objectives of privacy preservation. Although they have relatively weak privacy guarantee compared to principles such as *differential privacy* [16], we chose them in this paper because they are intuitive and have been justified to be useful in many practical applications such as privacy-preserving location services. Under the protocol in this paper, the data providers and users can only see the result of the designed functions and can not learn anything more.

Privacy for Data Objects Based on Anonymity. k-anonymity [11] [19] was regarded as an important privacy principle that protects against individual identifiability, it requires a set of k records to be indistinguishable from each other based on a quasi-identifier group. As mentioned above, given a microdata set D, attributes are characterized into: *unique identifiers* attribute A_0 which identifies individuals (e.g. names); *quasi-identifier* which is a minimal set of attributes $(A_1, ..., A_d)$ that can be combined with an external database to recover the personal identities; and sensitive attribute that should be protected. The set of all tuples containing identical values for the QI set is recorded as an equivalence class. An improved principle l-diversity [13], requires each equivalence class contains at least l diverse sensitive values. Given our research goal of extending the anonymization techniques and integrating them with secure computation techniques, we use the principle of k-anonymity and l-diversity to achieve privacy protection.

Privacy between Data Providers. Given that multiple data providers need to publish their private datasets in the cloud for data sharing, certain background knowledge may enable attacks for individual identifiability by composing several local anonymized datasets in servers (see composition attack [31]). Our second privacy goal is to avoid the attack between data providers, in which individual dataset reveal nothing about their data to the other data providers apart from the virtual anonymized database. It resembles the goal of secure multi-party computation (SMC) [28]. In SMC, a protocol is secure if no participant can learn anything more than the result of the function (or what can be derived from the result).

Similar to the method [5] using the multi-dimensional top-down $Mondrain$ algorithm for distributed anonymization, we propose a distributed anonymization protocol using R-tree [23] [24]. We assume that the initial local k_i-anonymous tables have been built. We use a distributed anonymization algorithm to build a virtual K-anonymous database and ensure the locally anonymized table d_i^* to be k_i-anonymous. When users query the virtual database, each individual database executes the query on d_i^* and then engage in a distributed querying protocol to assemble the results that are guaranteed to be K-anonymous.

3.3 Anonymization Algorithm

Given our privacy model, one important thing is to design an anonymization algorithm with checking mechanism for cloud servers and implement the algorithm

using distributed protocols. Generally speaking, we can decompose a centralized anonymization algorithm and utilize distributed protocols for communication which are proven to be secure in order to generate a secure and distributed anonymization among cloud servers. However, even every step is secure, using the results obtained from one secure computation to perform another computation at another server, may still reveal intermediate information that is not part of the final results. Therefore, minimizing the disclosure of intermediate information is an important consideration when designing such protocols.

It has mentioned above that we plan to generalize against the initial local k_i-anonymous tables using a distributed anonymization algorithm, and aim at building a virtual database that guaranteed to be K-anonymous and maintaining the local anonymized datasets to be k_i-anonymous from the beginning to the end. There are plenty of algorithms proposed to achieve k-anonymity, in this paper, we choose the bottom-up R-tree generalization method instead of the top-down *Mondrain* algorithm to achieve k-anonymity for better generalization efficiency (See Section 6.2).

Based on the multi-dimensional R-tree algorithm, our distributed anonymization protocol uses a greedy bottom-up approach to recursively insert the data objects into the minimum overlapping quasi-identifer domain space, and when overflow occurs, the quasi-identifer domain space will be split into two parts. It recursively chooses the best branch to insert the data objects for grouping the closest data objects. The split of the R-tree leaf node is repeated until it can not be divided, meaning that the data objects in a particular region cannot be divided without violating an anonymity constraints or generalization hierarchies.

4 Distributed Anonymization

4.1 Protocol

The main idea of the distributed anonymization protocol is to use secure multi-servers computation protocols to realize the R-tree generalization method for the cloud setting, so that each database produces a local anonymized dataset to meet the data provider's need, and the union of all anonymized data sets forms a virtual database that is K-anonymous.

We assume that a master node is selected from cloud server for the main protocol, and all the other local databases are consider as slave nodes. The protocols for the master node and other slave nodes are presented in Algorithm as illustrated in Fig.2. Clearly, the procedure performed at the master node is similar to the centralized generalization method. Considering the possibility of too much workload imposed to the master node, slave nodes only need to send summary information of each *equivalence class* to the master node for generalization in form (I, num), where I denotes a d-dimensional rectangle which is the bounding box of the *equivalence class*'s spacial QI values $[l_1, u_1], ..., [l_d, u_d]$ and num is the total number of data objects in the *equivalence class*. In addition, the sender slave node's address S_{id} is need to record because the master can use

Algorithm Master node
Phase 1: Read data from each Slave node
1. read data (I, num) from each slave node into a set S
Phase 2: Generalize data to be *K*-anonymous table
2. For each rectangle $I \in S$ do
3. Insert I into an R-tree
4. if split is possible, do *Split*(see the process of split)
5. end For
Phase 3: Modify the local k_i-anonymous database
6. For each *equivalence class* E_i of the K-anonymous table, do
7. get rectangles set $\mathcal{G}$ contained in R of E_i
8. For each rectangle $I \in \mathcal{G}$
9. Send I and R to Slave node through S_{id}
10. end For
11. end For

Algorithm Slave node i $(i > 0)$
Phase 1: Send *equivalence classes* to the master
1. For each *equivalence class* E_j of k_i-anonymous table, do
2. get summary information of E_j in form (I, num)
3. send data (I, num) and address S_{id} to the master
4. end For
Phase 2: Receive modifying commands from the master
5. Read the data I and R from the Master
6. For each *equivalence class* E_j of k_i-anonymous table, do
7. if rectangle of E_j equals I
8. Enlarge the rectangle of E_j to R
9. end if
10. end For

Fig. 2. Distributed anonymization protocol

it to send message back. Note that, before the distributed generalization, the initial local k_i-anonymous tables have been built among cloud servers.

In Phase 1, the master node reads data (I, num) of each *equivalence class* from each slave node into a set S. In Phase 2, the algorithm generalizes the data to be K-anonymous, where K is set by the cloud platform from a global view. The algorithm inserts each rectangle I from S into an R-tree. If split is possible, do *split*. Note that, the *equivalence class* read from slave nodes can not be split into two parts in the process of *split*. Later we will give the detail of the *split* process in Section 4.3. In Phase 3, the master node modifies all the initial local k_i-anonymous databases. The algorithm traverses each *equivalence class* E_i of the K-anonymous table, to find the rectangles set $\mathcal{G}$ contained in it. Note that, the rectangle is the data I that read from slave nodes at the initial stage. For each rectangle $I \in \mathcal{G}$ that contained in the bounding box R of E_i, we

Node 0

ID	Age	Zip Code
1	11 − 23	5200 − 5300
2	11 − 23	5200 − 5300

Node 1

ID	Age	Zip Code
3	73 − 80	5400 − 5500
4	73 − 80	5400 − 5500

Node 2

ID	Age	Zip Code
5	23 − 30	5200 − 5300
6	23 − 30	5200 − 5300
7	23 − 30	5200 − 5300
8	65 − 76	5400 − 5500
9	65 − 76	5400 − 5500
10	65 − 76	5400 − 5500

Node 3

ID	Age	Zip Code
11	45 − 60	5300 − 5400
12	45 − 60	5300 − 5400
13	45 − 60	5300 − 5400
14	45 − 60	5300 − 5400
15	45 − 60	5300 − 5400
16	45 − 60	5300 − 5400

Fig. 3. Initial local k_i-anonymous tables

Node 0

ID	Age	Zip Code
1	11 − 30	5200 − 5300
2	11 − 30	5200 − 5300

Node 1

ID	Age	Zip Code
3	65 − 80	5400 − 5500
4	65 − 80	5400 − 5500

Node 2

ID	Age	Zip Code
5	11 − 30	5200 − 5300
6	11 − 30	5200 − 5300
7	11 − 30	5200 − 5300
8	65 − 80	5400 − 5500
9	65 − 80	5400 − 5500
10	65 − 80	5400 − 5500

Node 3

ID	Age	Zip Code
11	45 − 60	5300 − 5400
12	45 − 60	5300 − 5400
13	45 − 60	5300 − 5400
14	45 − 60	5300 − 5400
15	45 − 60	5300 − 5400
16	45 − 60	5300 − 5400

Fig. 4. Modified local k_i-anonymous tables

send data I and R back to a corresponding slave node. When receive the data I and R from the master, the slave node finds out the *equivalence class* whose rectangle equals I and then enlarges its rectangle to R. Now our distributed anonymization algorithm completes.

4.2 Example

We illustrate the overall protocol with an example scenario as shown in Fig. 3 and Fig. 4, where four data providers publish their private databases across four slave nodes with their personalized privacy budget as $k_0 = 2, k_1 = 2, k_2 = 3, k_3 = 5$ and the global K=5. Before the distributed anonymization algorithm performs, four initial local k_i-anonymous tables have been built shown in Fig. 3,Node 0 is 2-anonymous, Node 1 is 2-anonymous, Node 2 is 3-anonymous and Node 3 is 5-anonymous. From Fig. 3 to Fig. 4, the distributed algorithm only modifies

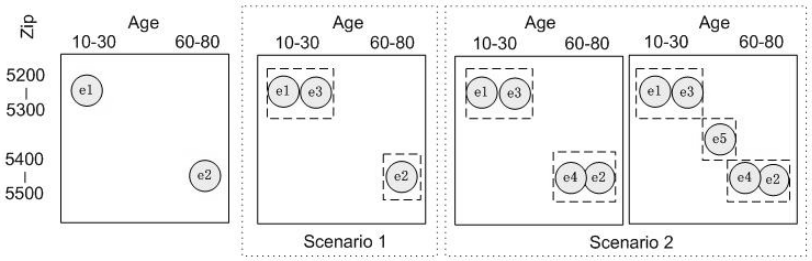


Fig. 5. Two scenarios in split process

the rectangle of the *equivalence class* of Node 0 from $[11-23][5200-5300]$ to $[11-30][5200-5300]$, Node 1 from $[73-80][5400-5500]$ to $[65-80][5400-5500]$ and the rectangle of two *equivalence class* of Node 2 from $[23-30][5200-5300]$ to $[11-30][5200-5300]$, $[65-76][5400-5500]$ to $[65-80][5400-5500]$ respectively. The distributed generalization algorithm first read all the *equivalence classes* from each 4 slave nodes and then insert the rectangle of each *equivalence class* into an R-tree for generalization. At last it modifies each local k_i-anonymous database and the union forms a virtual 5-anonymous database.

4.3 Split Process

The *split process* performed for a global R-tree in the master node is different from the *Node splitting* phase of a local R-tree. The algorithm focuses on the split of overflowed leaf nodes, which consists of rectangles of *equivalence classes* that read from other initially k_i-anonymous databases. As to the split of internal R-tree nodes, we can follow the split phase of R-tree in [23]. Our goal is to split the data as much as possible while satisfying the privacy constraints so as to maximize the utility of the anonymized data. The split algorithm first picks two seeds from the entries (rectangles of *equivalence classes*) that would get the largest area enlargement when covered by a single rectangle. That is, the distance between the two chose entries is the biggest. Then, for the others, one at a time is chosen to be added into one of the two groups. The one chosen is the one with the greatest difference in area expansion. If one group gets too full (more would force the other group to violate min fill requirement) then another group gets the rest. Finally, the entries are successfully split into two groups. We say the split is successful only if the two parts after splitting both satisfy the privacy constraints. e.g. assume the generalized table need to be k-anonymous and the total number of data objects in two groups is l and r respectively, so $l \geq k$ and $r \geq k$. Here we give two scenarios: First, the split is not successful because the two groups are not both meet the privacy constraints; Second, the splitting node is divided into two groups while both satisfy the privacy constraints.

We show the two scenarios in Fig. 5, where we denote the *equivalence classes* of Node 0 to Node 3 (See Fig. 3) as $e_1, e_2, ..., e_5$. Initially, *equivalence classes* e_1

and e_2 are inserted into the R-tree. When e_3 is inserted, the R-tree node splits into two groups, e_1 and e_3 into one group and e_2 into the other. But the number of tuples in e_2 is only 3 (less than 5), it is the scenario 1, the split is unsuccessful and the process will quit. When the *equivalence classes* e_4 comes, the scenario becomes different, e_1 and e_3 will be split into one group, e_2 and e_4 into the other. The two groups both satisfy the privacy constraints (the number of tuples in two groups both equal 5). At last, e_5 comes, following the insert algorithm, it will be inserted into the group that contains e_2 and e_4, and then the group will be split to two parts again, that is e_2 and e_4 in one group and e_5 the other, as illustrated in scenario 2 of Fig. 5.

5 R-Tree Generalization

Similar to the multi-dimensional top-down Mondrian algorithm [20] that generalizes data by using a Kd-tree, our method uses an R-tree. In particular, our algorithm uses a greedy bottom-up approach to recursively insert the data objects into the minimum overlapping quasi-identifer domain space, and when overflow occurs, the quasi-identifer domain space will be split into two parts. It recursively chooses the best branch to inset the data objects for gathering the closest data objects. When all the data tuples are inserted into the R-tree, the generalization table was built. Our method is experimentally verified to have better quality in anoymized dat than the Mondrian algorithm.

5.1 Index Structure

It is simple to generalize datasets to k-anonymous by mapping the tuples of each R-tree leaf node to an *equivalence class* and setting the capacity of leaf node's entries as 2×k-1. We can retrieve and gather all the leaf nodes in the R-tree to create a k-anonymous table. Leaf nodes contain index record entries of the form (I, SI), where SI refers to the sensitive information for a tuple, and $I = (I_1, I_2, ..., I_d)$ is a d-dimensional rectangle which is the bounding box of one *equivalence class*'s spacial QI values. Here d is the number of dimensions and I_i is a bounded interval [a,b] describing the QI value along dimension i. In this way, the whole information of an *equivalence class* can be easily retrieved from the leaf node. Non-leaf nodes contain entries of the form $(I, childPointer)$, where $childPointer$ is the address of a lower node in the R-tree and I tightly covers all rectangles in the lower node's entries. Here I is also regarded as a minimum bounding rectangle(mbr). Fig.6 shows a two-dimensional R-tree where 12 tuples $(a, b, c, ..., l)$ are clustered into 5 R-tree leaf nodes $R_3, R_4..., R_7$ according to their spatial proximity, then recursively grouped into notes R_1, R_2 which become the entries of the root.

5.2 Insertion and Node Splitting

The R-tree construction is based on the insertion algorithm. We create the R-tree by inserting every object into the indexing structure. Given a new object,

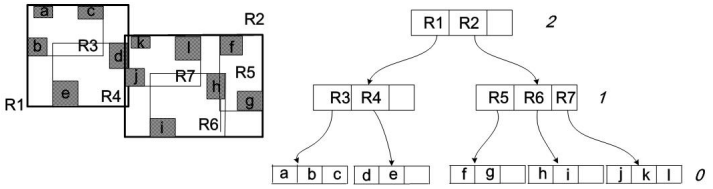


Fig. 6. R-tree structure

in a greedy way, the insertion algorithm chooses the best branch to follow at each level of the tree. Assume that we insert an object a which is contained by I of leaf node R_3 (in Fig.6). At the root level, the algorithm chooses the entry whose rectangle needs the least area enlargement to cover a, so R_1 is selected for its rectangle does not need to be enlarged, while the rectangle of R_2 needs to expand considerably. In the same way, R_3 is selected next. When following the leaf node, the SI of a is added into the index record entries of leaf node R_3 and the I of leaf node R_3 may be enlarged a little to contain a. The rectangles of its parent(like R_1) will be recursively updated.

An *overflow* occurs when the leaf node reached (i.e., data object m is added to the index record of leaf node R_3) is full(i.e., it already contains the maximum number of entries). In this case, the algorithm performs *Node Splitting*. The first step picks two seeds from the entries that would get the largest area enlargement when covered by a single rectangle. Then, of the remaining records, one at a time is chosen to be put in one of the two groups. The one chosen is the one with the greatest difference in area expansion. If one group gets too full (more would force the other group to violate min fill requirement), then the other group gets the rest. Finally, the entries are successfully split into two groups.

6 Performance Analysis

We evaluate the distributed anonymization protocol on Amazon's EC^2 platform. Our system was implement in Java 1.6.0.13 and run on a set of EC^2 computing units. Each computing unit is a small instance of EC^2 with 1.7GHz Xeon processor, 1.7GB memory and 160GB hard disk. The computing units are connected via 250Mbps network links. The number of computing units in our system is 10. We use three different datasets with $Uniform$, $Gaussian$ and $Zipf$ distribution to evaluate our distributed anonymization scheme.

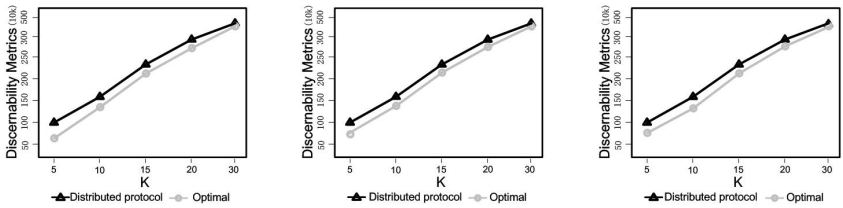
6.1 Performance of Distributed Anonymization Protocol

We first perform experiments comparing the quality using the discernibility metrics, which is based on the size of the equivalence classes G_j in D^*. Intuitively, the *discernibility metric* (DM) assigns each tuple r_i^* in D^* a penalty, which is determined by the size of the equivalence class containing it, that is $DM(D, D^*) = \sum_{i=1}^{m} |G_i|^2$. We present an evaluation of the distributed anonymization protocol

compared to the optimal centralized anonymization approaches in terms of this metric.

Dataset and Setup. For each synthetic dataset, we generate 10K tuples for each node. We report results for the following scenarios: 1) All the 100K tuples is located in one centralized database and R-tree generalization algorithm was used to generalize the database to be K-anonymous. In this scenario, we get the optimal discernibility metric value. 2) Data are distributed among the 10 nodes and we use the distributed anonymization approach presented in Section 4.

The first set of experiments compared these two approaches with varied K. We fixed the total number of tuples at 100K, and the number of attributes at 3. Our distributed anonymization protocol generalizes the 10 distributed databases to K-anonymous in global view but local generalized database with k_i-anonymous, where the k_i is selected independently and randomly from [1, K]. Results for the two generalization approaches are shown in Fig.7.



(a) Uniform (b) Gaussian (c) Zipf

Fig. 7. Discernability Metrics vs. k

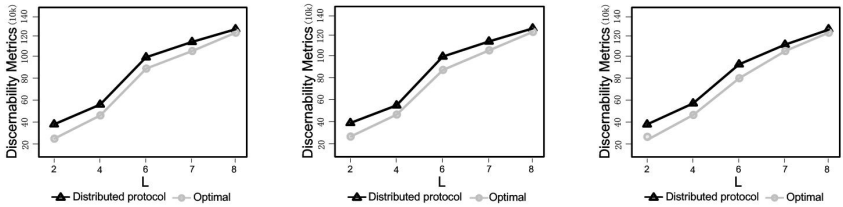
The next set of experiments in Fig. 8 measured the quality for varied diversity l, we extend the privacy principle from k-anonymous to l-diversity. We also fixed the number of attributes at 3 and used three kinds of distribution to perform our algorithm. The third set of experiments compared these two algorithms for varied dimensionality d, with diversity $l = 6$, as shown in Fig. 9.

Result. From all the above three sets of experiments, we observe that our distributed anonymization protocol performs close to the optimal anonymization.

6.2 Effect of R-Tree Generalization Method

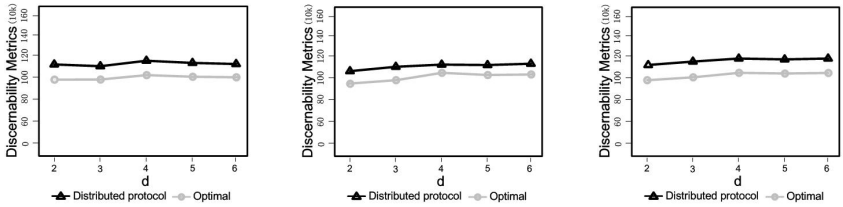
We perform experiments to evaluate generalization quality comparing with the *Mondrian* algorithm. Concretely, each private database uses these two methods to generalize data to be k_i-anonymous, then the union of them will be sent to the master for achieving a virtual K-anonymous database.

We measure the absolute error, i.e., $|actual - estimate|$, where *actual* is the correct range query answer number and *estimate* is the number of candidate set computed from the anonymous table. For each considered setting, we pose 100 queries that span a certain percentage of the entire domain space and report the



(a) Uniform (b) Gaussian (c) Zipf

Fig. 8. Discernability Metrics vs. l

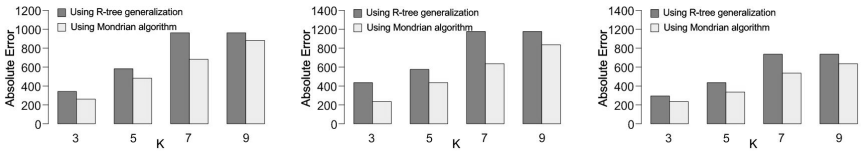


(a) Uniform (b) Gaussian (c) Zipf

Fig. 9. Discernability Metrics vs. d

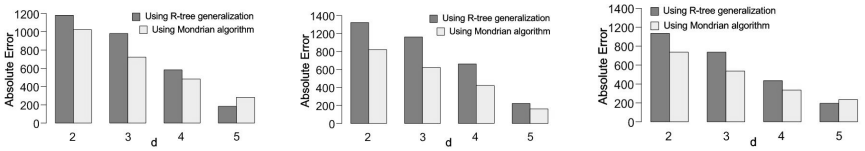
average absolute error of these queries under the same parameter settings. The first set of experiments measured the absolute errors with fixed dimension at 3 and varied K. In the next set of experiments, K is fixed at 7 and the dimension d is varied from 2 to 5.

Result. As shown in Fig.10 and Fig.11, queries on data anonymized by our method are more accurate than those on data anonymized by the Mondrian algorithm. Also as expected, the naive partition algorithm (*Mondrian* algorithm uses KD-tree for top-down data partition) suffers in data utility, this is caused by Mondrian making use of KD-tree, which does not group the closest data objects into the same minimum bounding rectangles.



(a) Uniform (b) Gaussian (c) Zipf

Fig. 10. Absolute Error vs. K



(a) Uniform (b) Gaussian (c) Zipf

Fig. 11. Absolute Error vs. d

7 Conclusion

We have presented a distributed anonymization protocol for privacy-preserving data publishing from multiple data providers in a cloud system. Our method performs a personalized anonymization to satisfy every data provider's requirements and the union forms a global anonymization to be published. We also presented a new anonymization algorithm using R-tree index structure. Our future work will continue along several directions. First, we are interested in developing a protocol toolkit incorporating more privacy principles like *differential privacy*. In particular, dynamic or serial releases of data with updates are extremely relevant in our cloud setting. Second, as compared to existing systems based on cryptographic approaches, we are interested in building efficient indexes based on anonymized cloud data to offer more efficient and reliable data analysis.

Acknowledgements. This work was supported by the ARC discovery grant DP110103142 and the National Natural Science Foundation of China under grant 61100060.

References

1. Vaquero, L.M., Merino, L.R., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. In: ACM SIGCOMM, pp. 50–55 (2009)
2. AMAZON. Nimbus health, `http://aws.amazon.com/solutions/case-studies/nimbus-health/`
3. AMAZON. Nimbus health, `http://aws.amazon.com/solutions/case-studies/sharethis/`
4. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. In: INFOCOM (2010)
5. Jurczyk, P., Xiong, L.: Distributed Anonymization: Achieving Privacy for Both Data Subjects and Data Providers. In: Gudes, E., Vaidya, J. (eds.) Data and Applications Security 2009. LNCS, vol. 5645, pp. 191–207. Springer, Heidelberg (2009)
6. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In: INFOCOM (2010)
7. Xiao, Y., Lin, C., Jiang, Y., Chuang, X., Liu, F.: An Efficient Privacy-Preserving Publish-Subscribe Service Scheme for Cloud Computing. In: GLOBECOM (2010)
8. Cao, N., Yang, Z., Wang, C., Ren, K., Lou, W.: Privacy-Preserving Query over Encrypted Graph-Structured Data in Cloud Computing. In: ICDCS (2011)

9. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds. In: Proc. of ACM Conference on CCS, pp. 199–212 (2009)
10. Kamara, S., Lauter, K.: Cryptographic cloud storage. In: RLCPS (2010)
11. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. (2002)
12. Wong, R., Li, J., Wai-Chee Fu, A., Wang, K.: (a, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In: KDD (2006)
13. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. TKDD (2007)
14. Xiao, X., Tao, Y.: m-Invariance: Towards Privacy Preserving Re-publication of Dynamic Datasets. In: SIGMOD (2007)
15. Bu, Y., Fu, A.W.C., Wong, R.C.W., Chen, L., Li, J.: Privacy Preserving Serial Data Publishing By Role Composition. In: VLDB (2008)
16. Xiao, X., Wang, G., Gehrke, J.: Differential Privacy via Wavelet Transforms. In: ICDE (2010)
17. Baig, M., Li, J., Liu, J., Wang, H.: Cloning for Privacy Protection in Multiple Independent Data Publications. In: ACM CIKM (2011)
18. Emekci, F., Agrawal, D., Abbadi, A.E., Gulbeden, A.: Privacy Preserving Query Processing using Third Parties. In: ICDE (2006)
19. Bayardo, R., Agrawal, R.: Data privacy through optimal k-anonymization. In: ICDE (2005)
20. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: SIGMOD (2005)
21. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems (2002)
22. Xiao, X., Tao, Y.: Personalized Privacy Preservation. In: SIGMOD (2006)
23. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: SIGMOD (1984)
24. Tao, Y., Papadias, D., Sun, J.: The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. In: VLDB (2003)
25. Fung, B., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey of recent developments. ACM Computing Surveys, CSUR (2010)
26. Jiang, W., Clifton, C.: A secure distributed framework for achieving k-anonymity. VLDB Journal (2006)
27. Zhong, S., Yang, Z., Wright, R.N.: Privacy-enhancing k-anonymization of customer data. In: PODS (2005)
28. Lindell, Y., Pinkas, B.: Secure multipart computation for privacy-preserving data mining. Cryptology ePrint Archive Report, `http://eprint.iacr.org`
29. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: IEEE ICDE (2006)
30. Li, N., Li, T., Suresh, V.: t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In: IEEE ICDE (2007)
31. Ganta, S.R., Kasiviswanathan, S.P., Smith, A.: Composition attacks and auxiliary information in data privacy. In: ACM SIGKDD (2008)

Subscription Privacy Protection in Topic-Based Pub/Sub

Weixiong Rao[1,2,⋆], Lei Chen[2], Mingxuan Yuan[2], Sasu Tarkoma[3], and Hong Mei[4]

[1] School of Software Engineering, Tongji University, Shanghai, China
rweixiong@gmail.com
[2] Department of Computer Science and Engineering, The Hong Kong Uni. of Sci. and Tech.
{leichen,csyuan}@cse.ust.hk
[3] Computer Science Department, University of Helsinki
sasu.tarkoma@cs.helsinki.fi
[4] Department of Computer Science and Technology, Peking Univeristy
meih@pku.edu.cn

Abstract. Topic-based publish/subscribe (pub/sub) is a popular paradigm to decouple message producers and consumers with the help of brokers. However, third-party brokers may be hacked, sniffed, subpoenaed, or impersonated. Thus, the brokers cannot be trusted. In particular, the collusion attack between compromised subscribers and untrusted brokers easily exposes the privacy of honest subscribers. Given the untrusted brokers and collusion attacks, traditional security techniques alone cannot protect subscribers' privacy. By adopting the k-anonymity model to the topic-based pub/sub, we propose to use cloaked subscriptions to blur subscribers' real interests. Such cloaked could protect the subscription privacy but meanwhile incur high forwarding cost. Thus, we minimize the forwarding cost meanwhile satisfying the privacy requirement, and formulate an integer programming (IP)-based optimization problem. After relaxing the IP problem to a linear programming (LP) problem, we design a new rounding algorithm that optimally minimizes the expected forwarding cost. The experiments show that our scheme efficiently achieves the trade-off between the forwarding cost and the privacy requirement.

1 Introduction

The topic-based publish/subscribe (pub/sub) [10] is a simple yet very popular paradigm to decouple message producers and consumers. *Subscribers* declare their interests by specifying logic topics in subscription conditions, and *brokers* maintain channels associated with topics specified in subscriptions. Publications are identified by specific topics. On receiving the publication messages from *publishers*, brokers forward publications to interested subscribers in a *one-to-many* manner.

Unfortunately, there are security concerns with respect to (w.r.t) the brokers: they may be hacked, sniffed, subpoenaed, or impersonated. Thus, the brokers cannot be trusted. If some users subscribe to sensitive publications (e.g., corporation or military or political/religious affiliations), the untrusted brokers could expose such subscribers. In particular, by deploying brokers as public third-party servers, many modern applications, such as online games, RSS feeds, and social computing platforms, have adopted

⋆ Part of this work was done when the first author was at the HKUST.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 361–376, 2013.

the pub/sub paradigm which allows end users to subscribe to favorite content. Attacks against public third-party servers could easily leak sensitive privacy information belonging to subscribers. On April 27 2011, Sony admitted that its PSN platform had been hacked, leading to the information leakage of 70 million users [3]. This example confirms that untrusted public servers could lead to users information leakage.

Due to the existence of untrusted broker servers and potentials of information leakage, a number of research proposals [15,21,20] have been proposed to address the security issues in pub/sub. Most of them are related to publication access control, data (i.e., publication) confidentiality, secure publication routing (using cryptographic techniques), and so on. However, none of these works is related to the subscriber privacy. In particular, the *collusion attack* [8] between compromised subscribers and untrusted brokers easily leaks subscribers' privacy. For example, a group of users subscribe to a specific topic pertaining to sensitive publications. Suppose one of such users is compromised and colludes with the untrusted broker. Due to the one-to-many communication pattern, the broker can link the sensitive publications with a set of recipients (including the compromised user and the remaining honest ones). Though we can encrypt the publications (and even the topic name), the compromised user decrypts such encrypted publications. Then, attackers can easily infer that the honest users are also interested in such sensitive publications.

In this paper, to protect subscribers' privacy, we adopt the k-anonymity [22,19] to the topic-based pub/sub, called *k-subscription-anonymity*. With this privacy model, the collusion between an untrusted broker and compromised subscribers exposes honest subscribers with probability at most $1/k$. To implement the privacy model, we propose to use *cloaked subscriptions*, and register the cloaked subscriptions to the channels of topic-based pub/sub. Thus, among the cloaked subscriptions (including both fake and real subscriptions) in the channel of a topic, it is indistinguishable which subscriptions are truly interested in the topic.

A naive approach to create the cloaked subscriptions is to register all subscriptions on every channel. This approach offers the privacy protection but incurs high network traffic. Thus, we propose to minimize the overall forwarding cost, and formulate an integer programming (IP)-based optimization problem. After relaxing it to a linear programming (LP) problem. Differing from the classic rounding approach [16], we design a new rounding algorithm that guarantees to optimally minimize the expected overall forwarding cost. To summarize, we make the following contributions.

- We identify that that the traditional secured pub/sub alone cannot defend against the collusion attack between untrusted brokers and compromised subscribers.
- We introduce an anonymizer engine to separate the roles of brokers, propose the k-subscription-anonymity model, and achieve the trade-off between the privacy protection and efficiency goal.
- Our experiments show that our solution only consumes a slightly higher cost to offer the subscription privacy protection (e.g., for a large anonymity level $k = 40$, the proposed scheme uses only 2.48 folds of cost compared with the original pub/sub).

The rest of this paper is organized as follows. Section 2 gives the preliminaries. Next, Sections 3 introduces the proposed technique and derives the criteria to meet the privacy requirement. After that, Section 4 designs the algorithm to achieve the trade-off

between the privacy requirement and the efficiency goal. Section 5 evaluates the proposed scheme. Section 6 investigates the related work. Finally Section 7 concludes this paper. Due to the space limit, the proofs of the theorems refer to our full report [2].

2 Preliminaries

In this section, we first give an overview of the topic-based pub/sub. After that, we define the k-subscription-anonymity model, and state our problem.

2.1 Topic-Based Publish/Subscribe

The topic-based pub/sub is a popular paradigm to decouple message publishers and subscribers, due to its simple interface, inherent scalability, and widely acceptance by both academic and industry communities. Many applications adopt the topic-based pub/sub to offer asynchronous delivery services [6,7], including RSS feeds, on-line gaming, etc.

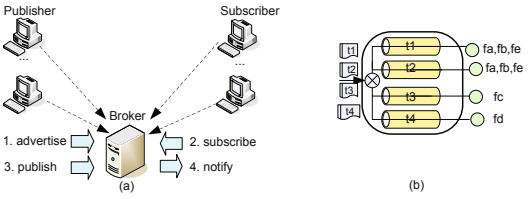


Fig. 1. Topic-based Pub/Sub: (a) architecture; (b) broker internals

Fig. 1(a) consists of publishers, subscribers and a broker server. The broker dispatches messages, which are regulated by advertisements, subscriptions, and publications [5]. Publishers first advertise publications that they intend to publish by means of *advertisements*. The advertisement contains valid topics and statistics associated with to-be-published publications. Subscribers specify the topics of interest by means of *subscriptions*. Frequently, subscribers share *subscription proxy services* (e.g., proxy servers in campus and large companies) to save the network traffic between brokers and subscribers. For every topic t_i in subscriptions, the broker maintains an associated channel, and registers the subscriptions to the channel of t_i. When publications come, the broker checks whether or not the topics of the publications are exactly the same as the ones defined by the filters. If true, the broker consequently forwards publications to the matching subscribers via *notifications*.

Fig. 1(b) illustrates the channels maintained by a broker. Among the five subscriptions, three of them, f_a, f_b and f_e, are interested in two topics t_1 and t_2, and the remaining subscriptions f_c and f_d are interested in t_3 and t_4, respectively. The broker registers each filter to the associated channel. When 4 publications are published, based on the channel of each topic, the publications are forwarded to interested subscribers.

Notations: Given the sets of topics $\mathcal{T}$, publications $\mathcal{M}$ and subscribers $\mathcal{N}$, we assume that each topic $t_i \in \mathcal{T}$ with $1 \leq i \leq T$ is associated with m_i publications and n_i subscribers. Denote the cardinality of $\mathcal{M}$ and $\mathcal{N}$ to be M and N. We have $\sum_{i=1}^{T} m_i = M$ and

$\sum_{i=1}^{T} n_i = N$, respectively. We note that the advertisements given by publishers contain the statistical parameters like m_i, M, etc.

Given the above sets $\mathcal{T}$ and $\mathcal{N}$, we define a binary coefficient $x_{ij} = 1$ if a subscriber $f_j \in \mathcal{N}$ is interested in a topic $t_i \in \mathcal{T}$, and otherwise $x_{ij} = 0$. For a specific topic t_i, multiple subscribers may be interested in t_i and we denote the set of the associated subscriptions to be Θ_i. Next a subscriber f_j could be interested in multiple topics, and we denote the set of the associated subscriptions to be Φ_j. Intuitively, given the T topics and N subscribers, we then treat the subscriptions x_{ij} as the elements of a $T \times N$ matrix. The Θ_i (resp. Φ_j) can be as a row (resp. column) of the matrix in terms of t_i (resp. f_j).

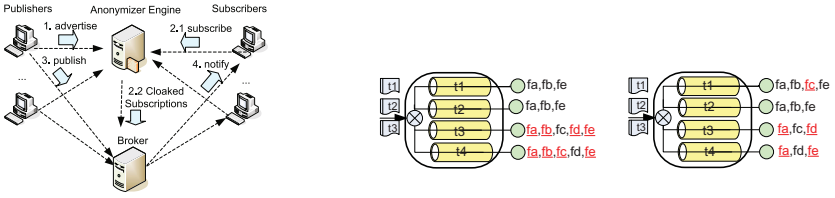
Consider that subscribers shre subscription proxies to save network traffic. We then define an indicator $z_{jj'} = 1$ if two subscribers f_j and $f_{j'}$ $(1 \leq j \neq j' \leq N)$ share the same proxy and otherwise $z_{jj'} = 0$. Given the indicator, we build a matrix Z consisting of $N \times N$ elements $z_{jj'}$. We call the property that subscribers share the same proxy as the *subscriber locality* property. This property shares the network bandwidth between brokers and subscribers and thus saves network traffic cost.

2.2 Privacy Model

Brokers offer the excellent decoupling property for subscribers and publishers. Unfortunately, the decoupling property meanwhile leads to the leakage of subscribers' privacy, because the the broker inherently works by the two roles: (i) registering subscriptions with (encrypted) topics t_i to associated channels, and (ii) forwarding (encrypted) publications with (encrypted) topics t_i to subscribers inside associated channels. Thus, via the topic t_i, the broker can *link* sensitive publications of t_i with a set of recipients. Due to the linkage, the following collusion attack exposes subscribers' privacy.

Collusion Attack: *For a subscription set Θ_i and a honest subscriber $f_j \in \Theta_i$, the broker and up to $(N - k)$ compromised subscribers (except the other $k - 1$ honest subscribers in Θ_i) collude together against f_j.*

Given the collusion attack, for a specific topic t_i, attackers can link sensitive (though encrypted) publication of t_i with a set of recipients (including both compromised and honest subscribers). After the compromised subscribers decrypt the encrypted publications, attackers then infer that honest subscribers are truly interested in such sensitive publications. Therefore, even if the publications are encrypted, attackers correspondingly expose the privacy of the honest subscribers.



(a) Anonymization Engine (b) Cloaked Subscriptions (c) Defending Collusion Attack

Fig. 2. Privacy-aware Pub/Sub

To overcome the above issue, we introduce an *anonymizer engine*, illustrated in Fig. 2(a). After receiving subscription requests with encrypted topics (step 2.1), the anonymizer engine then generates *cloaked subscriptions* and sends them to the broker (step 2.2). Cloaked subscriptions contain both *real subscriptions* (truly interested in the associated topics) and *fake subscriptions* (not interested in the associated topics). Consistent with the notations in Section 2.1, we denote the set of cloaked subscriptions of t_i to be Θ'_i and a cloaked subscription to be x'_{ij}.

After receiving Θ'_i from the anonymizer engine, the broker builds a channel for each topic t_i inside Θ'_i. Next, still following the original forwarding protocol, the broker checks the topics t_i of publications and the associated channels, and forwards the publications to all (cloaked) subscriptions inside the channels. Since the set Θ'_i contains both real and fake subscriptions, subscription proxies may receive useless publications. In view of this, our purpose is to minimize the total number of useless publications. Thus, subscription proxies spend least effort to filter out useless publications and notify subscribers only of truly useful publications.

To enforce subscribers' privacy, we adopt the k-anonymity [22,19] to the topic-based pub/sub, called *k-subscription-anonymity*. Because the subscriptions x'_{ij} in Θ'_i contain both real and fake subscriptions, then given $x'_{ij} = 1$, the claim of $x_{ij} = 1$ holds with probability at most $1/k$. That is, (i) the set Θ'_i contains at least k subscriptions; and (ii) given the set Θ'_i and $x'_{ij} = 1$, attackers cannot identify that $x_{ij} = 1$ must hold.

Until now, the anaonymizer engine and traditional cryptography technique can work together to defend against the collusion attack. In detail,

- First, though the broker forwards sensitive publications to subscribers, some subscriptions are fake and it is indistinguishable which subscribers are really interested in the sensitive publications. Thus, the broker, even with the help of $(N - k)$ compromised subscribers, cannot reveal the honest subscribers' privacy.
- Second, though the anonymizer engine receives real subscriptions, typically the topics are encrypted using the cryptography technique. Thus, a curious anonymizer engine alone cannot reveal the subscribers who are truly interested in such sensitive topics. It is because the topics inside real subscriptions can be encrypted as meaningless ciphertext.

Therefore, neither the anonymizer engine nor the broker alone can separately reveal the subscription privacy, unless the anonymizer engine and the broker (plus compromised subscribers) collude together. Obviously, it disobeys the definition of the collusion attack. We note that the anonymizer engine is typically operated by the trusted authority (the same situation occurs for the widely used certificate authority (CA)). Thus, the strong collusion between the anonymizer engine and the pub/sub services is practically infeasible. Nevertheless, to defend against such strong collusion, we use classical cryptographic techniques such as secure multi-party computation [24,12]. Specifically, a subscriber, together with the remaining $(k - 1)$ honest subscribers, registers its filter to the subscription proxy by the technique of secure multi-party computation. This allows a set of N subscribers to register subscriptions and receive content of interests without revealing subscriptions to each other or to outside observers of their publication traffic. It shields every subscriber even against the collusion attack plus the collusion between the untrusted broker and curious anonymizer engine. Thus, our solution is not to replace

cryptographic techniques. Instead, they work together to offer the complete solution to protect subscribers' privacy.

2.3 Problem Statement and Challenges

Given the privacy models above, we state our problem: the privacy aware topic-based pub/sub scheme should satisfy four following requirements.

- *Function Requirement*: each subscriber, if truly interested in a topic t_i, should receive all publications of t_i without false negatives;
- *Privacy Requirement*: given the collusion attack, a subscriber is exposed to be interested in t_i by probability at most $1/k$;
- *Capacity Constraint*: for each subscription proxy, the number of received publications is no more than the proxy's capacity constraint;
- *Efficiency Objective*: when M publication messages are published, the total number of forwarded publications from the broker to subscription proxies is minimized.

We will formally formulate the above problem by an integer programming form in Section 4. The **challenge** of the above problem is to meet all requirements. For example, if without meeting the efficiency goal and the capacity constraint, the anonymizer engine registers all N subscriptions to each channel. Then, it is the most difficult to expose a subscriber. However, this solution leads to the largest number $M * N$ of messages from the broker to subscription proxies. It incurs the lowest efficiency and easily breaks the capacity constraint.

Differing from the solution above, Section 3 will derive the criteria to protect subscription privacy, and Section 4 will finally solve the proposed problem.

3 Privacy Protection

In this section, we first give an overview of cloaked subscriptions. Next, we derive the criteria to meet the k-subscription-anonymity and to defend against the collusion attack.

3.1 Overview and Analysis Model

Our basic idea is to ensure that the channel of a topic t_i registers both fake subscriptions and real subscriptions (together called cloaked subscriptions). That is, though the subscriber f_j is interested in $t_{i'}$ but not interested in t_i ($\neq t_{i'}$), i.e., $x_{i'j} = 1$ but $x_{ij} = 0$, the processing of the anonymizer engine ensures that f_j is registered to the channels of both t_i and $t_{i'}$. That is, the (cloaked) subscriptions w.r.t f_j include both $x'_{i'j} = 1$ and $x'_{ij} = 1$. Therefore, it is indistinguishable which subscriptions are truly interested in t_i and which are not.

Example 1: To protect the subscriber f_c that is truly interested in t_3 (see the real subscriptions of Fig. 1(b)), we register four other subscribers (f_a, f_b, f_d and f_e), though not truly interested in t_3, to the channel of t_3, shown in Fig. 2(b). Thus, the cloaked subscription set Θ'_3 contains five subscriptions f_a, f_b, f_c, f_d and f_e where f_c is truly interested in t_3. Similar situation occurs in the cloaked subscription set Θ'_4 associated with t_4.

Given the anonymity number $k = 2$, we show that the collusion attack cannot expose f_c of Fig. 2(b). In the cloaked subscription set Θ'_3, even if knowing the interests of any $(N - k) = 3$ subscribers (e.g., f_a, f_b and f_e), attackers cannot distinguish which subscriber (f_c or f_d) is truly interested in t_3. Meanwhile, since f_c is registered to the channels of both t_3 and t_4, attackers cannot identify which topic, either t_3 or t_4, f_c is interested in. Thus, f_c (plus f_a, f_b and f_d) is safe against the collusion attack.

Analysis Model: Before deriving the criteria to meet the privacy protection, we first report the used analysis model, which helps formulate the proposed integer programming problem. Consider the subscriber set $\mathcal{N}$ and the topic set $\mathcal{T}$. Recall that $x_{ij} = 1$ if a subscriber $f_j \in \mathcal{N}$ is truly interested in a topic $t_i \in \mathcal{T}$ and otherwise $x_{ij} = 0$. Based on x_{ij}, we build a matrix X consisting of T rows and N columns. The element of X (i.e., the subscription x_{ij}), is associated with a topic t_i and a subscriber f_i. The matrix X has the following properties:

- The *row* of X w.r.t a topic $t_i \in \mathcal{T}$, i.e., Θ_i, represents the subscriptions that are interested in t_i. Thus, the sum of all elements in the row of t_i, i.e., $\sum_{j=1}^{N} x_{ij}$, is the total number of subscribers inside $\mathcal{N}$ that are truly interested in t_i. For any two topics t_i and $t_{i'}$, then we easily verify that $\sum_{j=1}^{N}(x_{ij} \cdot x_{i'j})$ is the number of subscribers that are interested in both t_i and $t_{i'}$.
- The *column* of X w.r.t a subscriber $f_j \in \mathcal{N}$ (i.e., Φ_j) represents the topics that f_j is truly interested in. Thus, the sum of all elements in the column of f_j, i.e., $\sum_{i=1}^{T} x_{ij}$, is the total number of topics that f_j is interested in. Note that we assume that each subscriber f_j is interested in at least one topic, then $\sum_{i=1}^{T} x_{ij} \geq 1$ holds.

Given the matrix X, the anonymizer engine generates a cloaked matrix X', also consisting of $T \times N$ element. The element $x'_{ij} \in X'$ indicates whether a subscriber f_j is added to the cloaked subscription set Θ'_i. Section 4 will present the details to build X'.

3.2 Privacy Criteria

Before deriving the privacy criteria, we first present a theorem to study the cases that subscribers are exposed (here we assume that f_j is truly interested in t_i).

Theorem 1. *f_j is exposed to be interested in t_i with probability higher than $1/k$, if either of the following cases occurs: (i) Θ'_i contains fewer than k subscriptions; or (ii) f_j appears in fewer than k cloaked subscription sets.*

Now, based on the above theorem, we derive the criteria to meet the privacy requirement. First, in terms of any topic t_i, if the row of t_i in X' contains at least k elements equal to 1, then the probability of identifying that t_i is of interest to f_j with probability at most $1/k$. Thus, we have **Criterion (1)**: if $\exists t_i$ with $x_{ij} = 1$, then $\sum_{j=1}^{N} x'_{ij} \geq k$.

Next, in terms of any topic f_j, the column of f_j in X' contains at least k elements equal to 1. Otherwise, it is easy to infer that f_j must be interested in t_i with probability higher than $1/k$. That leads to **Criterion (2)**: if $\exists f_j$ with $x_{ij} = 1$, then $\sum_{i=1}^{T} x'_{ij} \geq k$.

Criterion (2) is important as follows. Consider that f_j is truly interested in t_i and even a very large number of fake subscriptions are registered to the channel of t_i (due to Criterion (1)). If f_j appears in only one channel (e.g., t_i), f_j is then exposed to be certainly interested in t_i.

Besides the above two criteria that are used to meet the k-subscription-anonymity, we need to consider how the collusion attack can expose subscribers' privacy.

Theorem 2. *Though Criteria (1) and (2) are satisfied, the collusion attack can still expose f_j to be interested in t_i.*

Example 2: We use Fig. 2 (c) as an example (the k-anonymity number is 2) to verify the above theorem. The channel of t_3 contains three subscribers, and Criterion (1) thus holds. Meanwhile, f_c, truly interested in t_3, appears in the channel of t_1 as a fake subscription. Thus, Criteria (2) also holds. Similar situation occurs for the channel of t_4. Now, given the collusion attack, we assume that in channel of t_3, two subscribers f_c and f_d are honest (due to $k = 2$) and all other three subscribers f_a, f_b, and f_e are comprised. Among the three subscribers f_a, f_d and f_e in the channel of t_4, the two subscribers f_a and f_e are compromised (due to the collusion attack), and they are not interested in t_4. Because t_4 is of interest to at least one subscriber, attackers infer that f_d must be interested in t_4. Thus, the privacy of f_d is exposed.

By Example 2, we find Criteria (1-2) cannot defend against the collusion attack. Thus, we derive **Criterion (3)** together with Criteria (1-2) to defend against the attack: if $\exists t_i$ with $\sum_{j=1}^{N} x_{ij} = 1$, then $\sum_{j=1}^{N} x'_{ij} \cdot x'_{i'j} \geq (k+2)$ and $\sum_{j=1}^{N} (x'_{ij} + x'_{i'j} - 2x'_{ij}x'_{i'j}) \leq (k-2)$ hold.

4 Building Cloaked Subscription Matrix X'

Besides the privacy requirement in Section 3, in this section, we build the cloaked matrix X' to satisfy the other requirements of the problem definition in Section 2.3.

4.1 Overview

Among all four requirements of the proposed problem, we consider the problem to build the matrix X' as an integer programming-based optimization problem, where the *objective* is to minimize the forwarding cost, and three criteria in Section 3 are as *constraints*. It is an integer problem because the element x'_{ij} in X' is either 0 or 1.

To minimize the overall forwarding cost, we first leverage the *subscriber locality* property of subscription proxies (see Section 2.1) to reduce the forwarding cost (Section 4.2). After that, we formulate an integer programming based optimization problem, and relax it to a linear reprogramming problem with fractional results (Section 4.3). Finally, instead of the simple rounding algorithm [16], we propose a guaranteed randomized rounding algorithm to satisfy the required privacy criteria and optimally minimize the expected forwarding cost (Section 4.4).

4.2 Optimization Policy

Recall that the subscriber locality property means some subscribers share the same subscription proxies to save the network bandwidth between the broker and subscription proxies. Thus, the overall forwarding cost of the privacy-aware pub/sub is the total number of messages forwarded from the broker to subscription proxies.

For each topic t_i, we denote the number of publications of t_i to be m_i (the number is given in advertisements). Then, due to the effect of subscriber locality, the cost to forward these publications is $m_i \cdot [\sum_{j=1}^{N} x'_{ij} - \sum_{j=1}^{N} \sum_{j'=1}^{j-1} (z_{jj'} \cdot x'_{ij} \cdot x'_{ij'})]$. Here, the first item $\sum_{j=1}^{N} x'_{ij}$ is the total number of publications of t_i from the broker to subscription proxies when the subscriber locality is not adopted. The second item $\sum_{j=1}^{N} \sum_{j'=1}^{j-1} (z_{jj'} \cdot x'_{ij} \cdot x'_{ij'})$ means that if f_j and $f_{j'}$ use the same proxy (i.e., $z_{jj'} = 1$), the forwarding of a publication of t_i to the two subscribers f_j and $f_{j'}$ needs only one message copy from the broker to the proxy which both f_j and $f_{j'}$ share. Given T topics, the overall forwarding cost is $\sum_{i=1}^{T} m_i \cdot [\sum_{j=1}^{N} x'_{ij} - \sum_{j=1}^{N} \sum_{j'=1}^{j-1} (z_{jj'} \cdot x'_{ij} \cdot x'_{ij'})]$.

Background Knowledge Attack: First consider that a large number of subscribers are truly interested in a topic t_i, i.e., the set set Θ_i contains a large number of member subscriptions. Next, if more (cloaked) subscriptions inside the subscription set Θ'_i share the same subscription proxies, the overall forwarding cost becomes smaller. Given our optimization objective to minimize the overall forwarding cost, purposely adding more fake subscriptions, which share the same subscription proxies as those real subscriptions inside Θ_i, to the set Θ'_i then helps reduce the overall forwarding cost.

The above optimization essentially is a greedy policy. It does reduce the forwarding cost, but meanwhile incurs a potential risk of exposing real subscriptions if attackers know the optimization policy and some background knowledge. We call this attack *background knowledge attack*. In terms of the background knowledge, it is well-known that the number of subscribers who are interested in topics typically follows a Zipf distribution [13]. That is, more users are interested in popular topics and few are interested in unpopular topics. If the greedy optimization policy is adopted, the channel of a popular topic will register more fake subscriptions. Next, by counting the number of cloaked subscriptions inside the channels, attackers correspondingly derive the following observations: (i) the channels having the smallest number of subscribers might be associated with unpopular topics; and (ii) most subscribers registered to unpopular channels are real and have more potential to be truly interested in the associated unpopular topics.

Since the privacy criteria in Section 3.2 do not consider the background knowledge attack, we need to set up an upper bound H to limit the number of subscribers registered to each channel. The number H is inside the range between $\sum_{j=1}^{N} x_{ij}$ and N; otherwise, incurring an infeasible solution for our optimization problem. Considering that $\sum_{j=1}^{N} x_{ij}$ subscribers are truly interested in t_i, we can set up H smaller than $k \cdot \sum_{j=1}^{N} x_{ij}$. It makes sense because the probability to identify any of those subscribers truly interested in t_i is at most $1/k$, which is consistent with the definition of k-anonymity. Now, we improve **Criterion (1)**: if $\exists t_i$ with $x_{ij} = 1$, then $k \leq \sum_{j=1}^{N} x'_{ij} \leq H$.

4.3 Problem Formulation

We formulate an optimization problem to build the matrix X'_i (called Cloaked Subscription Matrix problem, in short CSM) as follows.

Given	matrix X, matrix Z, and m_i with $1 \le i \le T$
To build	matrix X' with $T \times N$ elements x'_{ij}
Minimize	$\sum_{i=1}^{T} m_i \cdot [\sum_{j=1}^{N} x'_{ij} - \sum_{j=1}^{N} \sum_{j'=1}^{j-1} (z_{jj'} \cdot x'_{ij} \cdot x'_{ij'})]$
Subject to	(1)$\exists t_i$ with $x_{ij} = 1$, then $k \le \sum_{j=1}^{N} x'_{ij} \le H$;
	(2)$\exists f_j$ with $x_{ij} = 1$, then $k \le \sum_{i=1}^{T} x'_{ij} \le C_j$;
	(3)$\exists t_i$ with $\sum_{j=1}^{N} x_{ij} = 1$, then $\sum_{j=1}^{N} x'_{ij} \cdot x'_{i'j} \ge (k+2)$, and $\sum_{j=1}^{N} (y_{ij} + y_{i'j} - 2x'_{ij}x'_{i'j}) \le (k-2)$;
	(4)$\exists f_j$ with $x_{ij} = 1$, then $x'_{ij} = 1$;

In the above CSM problem, there exist four criteria. Among these criteria, **Criteria (1-3)**, pertaining to the privacy requirement, are given by Section 3.2. The original **Criterion (2)** is improved by setting $\sum_{i=1}^{T} x'_{ij} \le C_j$, where C_j is the capability limitation of the associated proxy. It ensures $\sum_{i=1}^{T} x'_{ij}$, i.e., the number of publications forwarded to the proxy associated with f_j, is no larger than C_j. In addition, the **Criterion (4)** ensures the function requirement (see Section 2.3). That is, if a subscriber f_j is truly interested in t_i (i.e., $x_{ij} = 1$), then $x'_{ij} = 1$ must hold.

We show that CSM is NP-hard. Due to space limit, we ignore the details of the proof (refer to our technical report). Thus, we relax the 0/1 element x'_{ij} into an fractional element $y_{ij} \in [0.0, 1.0]$, and replace x'_{ij} with y_{ij}. Then, the CSM problem is transformed into a linear programming problem (in short LPCSM).

Note that the subitems $\sum_{j=1}^{N} \sum_{j'=1}^{j-1} (z_{jj'} \cdot x'_{ij} \cdot x'_{ij'})$ and $\sum_{j=1}^{N} x'_{ij} \cdot x'_{i'j}$ in the objective and Criterion (3) of CSM are not in strict linear programming (LP) form. Thus, we need to simplify both subitems to a LP form. In detail, (i) in the subitem $\sum_{j=1}^{N} \sum_{j'=1}^{j-1} (z_{jj'} \cdot x'_{ij} \cdot x'_{ij'})$, we replace the inner variables j' by those variables j' satisfying $x_{ij'} = 1$. Since $x_{ij'}$ is given by X, the objective of CSM becomes the LP form. The intuition of this simplification is that for any subscriber $f_{j'}$ truly interested in t_i with $x_{ij'} = 1$, we expect those to-be-registered fake subscriptions share the same proxies as real subscribers $f_{j'}$. It is consistent with the optimization policy in Section 4.2. (ii) to simplify Criterion (3), we note that Criterion (3) is to ensure that the set memberships of Θ'_i and $\Theta'_{i'}$ should be common as much as possible. Thus, we set $x'_{ij} = x'_{i'j}$ for all $1 \le j \le N$ for the simplification of Criterion (3) such that Θ'_i and $\Theta'_{i'}$ have exactly the same memberships. In this way, we relaxe Criterion (3) to a LP form.

4.4 Rounding Algorithm

Until now, the simplified CSM becomes the strict 0/1 LP form, which can be solved by the classical simplex algorithm with polynomial-time. The result of LPCSM can be intuitively viewed as a fractional scheme, where a subscriber f_j can be split into arbitrary parts and registered to a channel by probability y_{ij}.

When the fractional result y_{ij} of LPCSM is ready, the next step is the rounding scheme. The simple closest integer rounding or the classic randomized rounding approach [16]. However, such approaches incur problems. For example, by the approach [16] (we call it simple rounding scheme, in short SRS), in the fractional results related to f_j, there are two elements with $y_{ij} = 0.5$ and $y_{i'j} = 0.5$. It means that f_j is added to two sets Θ'_i and $\Theta'_{i'}$ respectively with the equal probability 0.5. In a trial of adding f_j to Θ'_i and $\Theta'_{i'}$, SRS might not add f_j to any set of Θ'_i and $\Theta'_{i'}$, and break the criteria of

LPCSM. Even with more trials, SRS might add f_j to one set, for example, Θ'_i. However, given $m_i \geq m_{i'}$, adding f_j to Θ'_i, instead of $\Theta'_{i'}$, incurs a larger forwarding cost.

Algorithm 1. Random_Alg (matrix Y with $0 \leq y_{ij} \leq 1$)

```
1: initiate matrix X' with x'_ij = 0;
2: L_u ← Σ_{i=1}^{N} C_j, L_l ← k · N, L ← 0;
3: while (L_l ≤ L ≤ L_u) is NOT satisfied do
4:     {/* There still exist subscribers dissatisfying Criteria (2) */}
5:     uniformly set y with a random value between 0.0 and 1.0; uniformly set t with a random num. between 1 and T;
6:     for j = 1 to N do
7:         ℓ_u ← C_j, ℓ_l ← k, ℓ ← Σ_{i=1}^{T} x'_ij ;
8:         if (ℓ_l ≤ ℓ ≤ ℓ_u) is NOT satisfied and y ≤ y_tj then
9:             {/* for each f_j dissatisfying Criteria (2), add f_j to the channel of t with probability y_tj */}
10:            x'_ij ← 1; y_tj ← y_tj − 1.0; L ← L + 1.0.
11:        end if
12:    end for
13: end while
```

To avoid the issues above, we develop a new rounding algorithm (Alg. 1). This algorithm can optimally minimize the *expected* forwarding cost and strictly satisfy all required criteria (except that Criterion (1) is expectedly satisfied). Its intuition is to ensure that the event of adding f_j to Θ'_i with probability y_{ij} occurs by multiple trials, until Θ'_i strictly registers at least k and at most C_j cloaked filters, (i.e., Criteria (2) is met), and on the overall, at least $k \cdot N$ (and at most $\sum_{i=1}^{T} C_j$) cloaked filters are created.

Finally, we analyze the objective. Given a subscriber $f_j \in \mathcal{N}$ uses a proxy P, we are interested in the probability that f_j and other subscribers using the same proxy P are registered to the same channel. We assume that P contains N_P subscribers (including f_j) with $1 \leq j' \leq N_P$. Then, we have the following theorem.

Theorem 3. *A subscriber f_j and other subscribers using the same proxy P are registered to the same channel with probability at least* $1 - \sum_{i=1}^{T} \sum_{j'=1, j' \neq j}^{N_P} min(y_{ij}, y_{ij'})$.

By the item $1 - \sum_{i=1}^{T} \sum_{j'=1, j' \neq j}^{N_C} min(y_{ij}, y_{ij'})$, Theorem 3 guarantees that Alg. 1 maximizes the *expected occurrence* of registering f_j and $f_{j'}$ into the same channel. It immediately means that the *expected* forwarding cost of LPCSM is minimized and equal to the optimal cost of CSM.

5 Evaluations

To generate the experimental data set (including the number of real subscriptions per topic and per subscription proxy, and the number of publications per topic), we follow the previous work [6,7] to use the Zipf distribution. Each subscriber specifies at least one subscription. After that, we use the popular LPSOLVE [1] to solve LPCSM. For each running instance of LPCSM, we translate it to an input file of LPSOLVE. Based on the fractional results, we then follow Alg. 1 to have integer results. By the integer results, we then register (cloaked subscriptions) to channels and calculate the forwarding cost. Table 1 gives main parameters. Note that, for the parameter H w.r.t each topic t_i, if the default value $k \cdot \sum_{j=1}^{N} x_{ij}$ is a small value, it could incur an infeasible solution of LPCSM. Thus, we relax H to set $H = N$. Similar situation occurs for C_j: if a small value of C_j leads to an infeasible solution, we set $C_j = M$.

Table 1. Parameters Used in Experiments

Parameter	Allowable Range	Default Value
T: # of topics	$40 - 1,000$	100
N: # of subscribers	$100 - 10,000$	1,000
M# of publications		1,000
S# of subscriptions		$N * 10$
H: max. subs per topic		$k \cdot \sum_{i=1}^{N} x_{ij}$
C_j: capacity of f_j		$k \cdot C'_j$
P: # of proxies	$10 - 10,000$	100
α: Zipf parameter	$0.01 - 1.0$	0.5
k: anonymity number	$2 - 80$	40

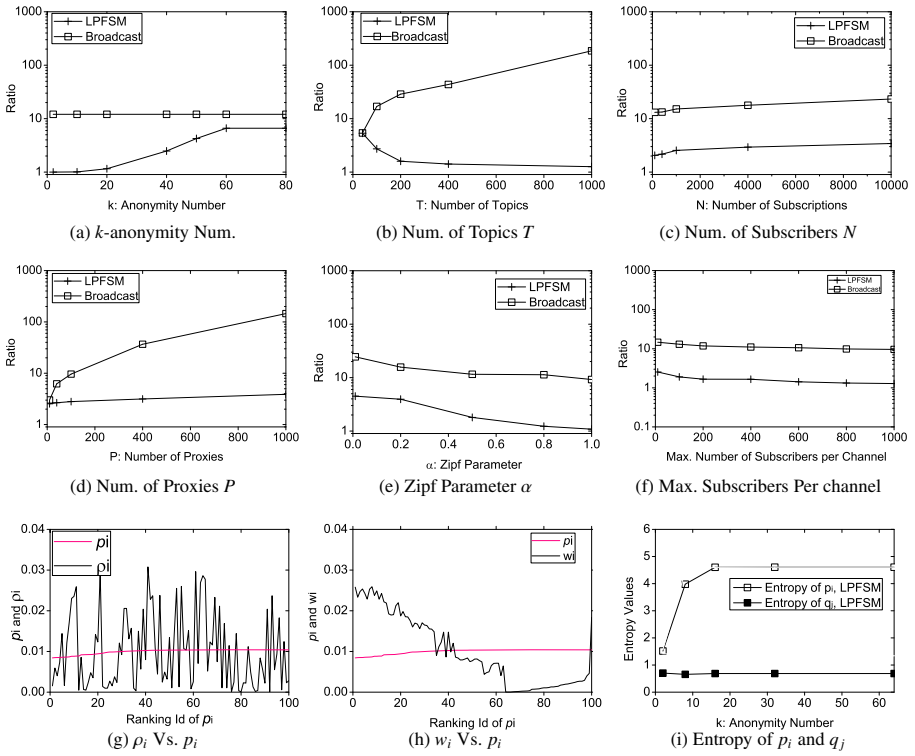
5.1 Efficiency Study

We first study the efficiency of the privacy aware pub/sub. We measure the efficiency by the ratio between the number of publications used by the privacy aware pub/sub and the number of publications of the original pub/sub. Such a metric is called *cost ratio*. In addition, we compare the LPCSM solution with the approach that broadcasts each publication to all proxies (in short *broadcast* solution).

First, Fig. 3(a) studies the effect of the anonymity number k. When the anonymity number k is larger, the cost ratios of both approaches become larger. This is because Criteria (1) and (2) directly require more fake subscriptions. In this figure, when $k = 40$, the cost ratio is only 2.48. It means that offering the high anonymity level $k = 40$ does not incur significantly high cost. For $k > 60$, the cost ratio of LPCSM keeps stable (= 6.62). That is, the proposed optimization policy in Section 4.2 ensures that fake subscriptions and real subscriptions share the same proxies, and thus even given a large k, LPCSM at most forwards the publications to the channels on which all real subscriptions are registered. It thus reaches such an upper bound. Instead, the broadcast approach has the cost ratio of 12.06, independent upon the anonymity number k. Obviously the broadcast approach incurs larger forwarding cost than LPCSM.

Second, in Fig. 3(b), we vary the number of topics T. In this figure, when $T = 40$ is equal to the default anonymity number k, the cost ratio of LPCSM is exactly equal to that of the broadcast approach. That is, given $T = k = 40$, LPCSM adds each subscription to all channels and each subscription proxy receives all publications, which just is the broadcast approach. After $T > 40$, more topics in LPCSM lead to lower cost ratio. When more channels are allowable to register fake subscriptions, LPCSM has more chance to optimize the forwarding cost, and achieves a smaller cost ratio. Meanwhile, due to the fixed number of subscriptions, more topics (i.e., channels) mean a diverse distribution of subscriptions over channels and a smaller average of subscriptions per channel. It thus help have more chance to select the best channels to register fake subscriptions, leading to a smaller cost ratio. Instead, for the broadcast approach, when T is larger, it always registers each subscriber to all channels, and thus the cost ratio of the broadcast approach becomes larger.

Fig. 3(c) shows the effect of subscribers. A larger number N of subscribers incurs higher cost ratios for both approaches. When N is larger, each channel has to register more subscriptions. Thus, more cost is paid to forward publications to associated proxies. Note that, since the number of subscription proxies (and topics) is fixed, the increased forwarding cost is relatively slight as shown in this figure.



(a) k-anonymity Num. (b) Num. of Topics T (c) Num. of Subscribers N

(d) Num. of Proxies P (e) Zipf Parameter α (f) Max. Subscribers Per channel

(g) ρ_i Vs. p_i (h) w_i Vs. p_i (i) Entropy of p_i and q_j

Fig. 3. Efficiency Study and Attack Resilience

Fig. 3(d) studies the effect of proxies. When the number P of proxies is larger, the cost ratio of the broadcast approach grows very fast because each publication is forwarded to all proxies. Instead, for LPCSM, the optimization policy ensures that fake and real subscriptions share the same proxies. Thus, even if P in increased, LPCSM ensures fake subscriptions share the bandwidth between the broker and proxies as much as possible. Therefore, a larger number of proxies will not significantly increase the cost ratio as the broadcast approach.

Fig. 3(e) studies the effect of the Zipf parameter α. When α is larger, the distribution of publications (and subscribers) across topics is skewer, i.e., most topics are unpopular and only several topics are very popular. Thus, LPCSM can optimize the assignment of fake subscriptions and register them to the channels of unpopular topics, and achieve a lower cost ratio. For the broadcast approach, since the number of channels associated with popular topics is very small, the overall number of subscribers registered to these popular channels is correspondingly small (also indicating a small number of subscription proxies). Thus, a very small number of proxies receive popular publications but most proxies receive unpopular publications, and the overall forwarding cost of the broadcast approach is smaller.

Finally Fig. 3(f) shows the effect of the maximal number H of real subscribers per channel. When such number is larger, then given a fixed number of subscribers, some channels register more subscribers than others. This indicates an uneven distribution of

subscribers across all channels. Thus, when H becomes larger, the cost ratio becomes smaller. Note that, the decreasing trend of this figure is relatively smooth than Fig. 3(e), because a larger H only affects the distribution of subscribers across channels but does not change the total number of publications.

5.2 Attack Resilience

Next, we proceed to evaluating the resilience of the proposed solution against the background knowledge attack. Following [23], we measure the strength of the subscription privacy protection by the following metrics.

- Given a topic t_i, we compute its popularity in Θ'_i and Θ_i by p_i=$\sum_{j=1}^{N} x'_{ij} / \sum_{i=1}^{T}\sum_{j=1}^{N} x'_{ij}$ (where the denominator is the total number of all cloaked subscriptions, and the numerator is the number of clacked subscriptions w.r.t t_i) and $\rho_i = \sum_{j=1}^{N} x_{ij} / \sum_{i=1}^{T}\sum_{j=1}^{N} x_{ij}$. We are interested in (i) the correlation between w_i and p_i and (ii) the correlation between ρ_i and p_i. We also calculate the entropy of p_i. A large entropy means an even distribution, which helps guard against the background knowledge attack.
- If a subscription f_j is registered to T'_j channels (denoted as $\mathcal{T}'_j$), we define the following formula: $q_j = \frac{\sum_{i=1}^{T} x_{ij}}{\sum_{i=1}^{T} x'_{ij}} \cdot (1 - \frac{\text{\#of subscriptions commonly registered to } \mathcal{T}_j}{\text{\#of all subscriptions registered to } \mathcal{T}_j})$. In this formula, the former subitem indicates the rate of the channels that f_j is truly interested in against all channels to which f_j is registered; its smaller value indicates better privacy protection (due to Criterion (2)). The latter subitem is related to the rate of subscriptions commonly registered to the channel set $\mathcal{T}_j$ against all subscriptions registered to $\mathcal{T}_j$; its smaller value also indicates better privacy protection (due to Criterion (3)). Besides, we calculate the entropy value of normalize q_j.

Fig. 3(g) shows the relation between ρ_i and p_i, where the x-axis shows the sorted ranking Id and the y-axis shows the corresponding ρ_i and p_i. In this figure, the originally popular topics in X (i.e., a larger ρ_i) might be unpopular in X' (i.e., a smaller p_i) and vice versa. Thus, given a skew distribution of topic popularities, it is indistinguishable which topics are of interest to subscribers, and the background knowledge attack cannot easily expose subscribers' privacy.

Fig. 3(h) shows the relationship between w_i and p_i. The x-axis shows the ranking Id of p_i in ascending order, and the associated w_i and p_i are respectively given in the y-axis. This figure clearly indicates low correlation between w_i and p_i. It prevents the background knowledge attack from exposing real subscriptions. The reason is that our optimization policy in Section 4.2 does not greedily register fake subscriptions to the channels with the lowest w_i. Instead, it registers the faked subscriptions at the same proxies as real subscriptions.

Fig. 3(i) plots the entropy values of p_i and q_j for LPCSM. First, when k becomes larger, the entropy value of p_i increases, indicating that the distribution of p_i becomes uniform. It helps defend against the background knowledge attack in Section 4.2. Second, the entropy value of q_j keeps unchanged because Criteria (2) and (3) are independent upon the value of k. Finally, for $k > 16$, the entropy value of p_i for LPCSM becomes stable, consistent with Fig. 3(a). Due to the space limitation, we do not plot the figures of those entropy values for other parameters like T, N, α and C, but with similar curves as Fig. 3(i).

6 Related Works

The k-anonymity privacy model [22,19] prevents attackers from identifying an individual with probability $1/k$. The main techniques of the works include generalization and suppression. In addition, ℓ-diversity [14] guards against attackers with homogeneity attack and background knowledge attack. Such works significantly differ from our work. First, they are completely different areas: [22,19,14] focus on the generalization and suppression of the micro data and the protection of the individual's privacy (e.g., healthy records), and our solution is designed for middleware systems. Second, [22,19,14] focus on the protection of the published data. Instead, our solution protects the subscription privacy, and does not generalize publications; otherwise, subscribers cannot receive correct and precise content. Finally, different from [22,19,14], our solution and the traditional cryptographic techniques works together to protect subscription privacy.

Many location services adopt location k-anonymity [11], location ℓ-diversity [4], and road segment s-diversity [23]. For example, the location k-anonymity mainly utilizes a cloaked region to represent the client location and this region needs to contain at least $(k-1)$ other client locations.The main difference between our work and the location privacy is the attack model. We consider the collusion of the broker server and all other $(N-k)$ compromised subscribers. The location privacy does not consider such collusion attack, and only focuses on the attack that attackers know privacy protection algorithms and some background knowledge, which are also considered in this paper.

Secured pub/sub systems [15,21,20] utilize cryptographic techniques to protect the data confidentiality, secure routing, publishers' privacy, but they did no consider the privacy issue of subscribers.

Finally, our recent work [18] focuses on privacy protection for content-based pub/sub, leading to the correspondingly different solutions. Another recent work [25] instead proposed privacy protection utilities used to to publish a privacy preserving graph.

7 Conclusion

Untrusted brokers in pub/sub lead to the leakage of subscribers' privacy. To address this problem, we propose a k-subscription-anonymity model and use cloaked subscriptions to protect subscribers' privacy. To trade-off the efficiency goal and privacy requirement, we consider an integer programming-based optimization problem, and relax it to a linear programming problem. We propose a guaranteed rounding algorithm to optimally minimize the expected forwarding cost. The experimental results indicate that the solution requires a slightly higher cost to offer the privacy protection. As the future work, we are interested in adapting other stronger privacy protection model (e.g., differential privacy [9]) in the pub/sub. In addition, we are planning to plug-in the privacy-aware solution into more semantic filtering pub/sub [17].

Acknowledgment. This work is supported in part by Hong Kong RGC grants HKUST612/09, NSFC/RGC Joint Research Project 60931160444 and National Grand Fundamental Research 973 Program of China under Grant 2012CB316200 We also would like to thank the anonymous DASFAA 2013 reviewers for their valuable comments that helped improve this paper.

References

1. http://sourceforge.net/projects/lpsolve
2. https://sites.google.com/site/rweixiong/topic.pdf
3. http://tracehotnews.com/sony-admitted-psns-70-million-users-information-leakage
4. Bamba, B., Liu, L., Pesti, P., Wang, T.: Supporting anonymous location queries in mobile environments with privacygrid. In: WWW (2008)
5. Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and evaluation of a wide-area event notification service. ACM Trans. Comput. Syst. 19(3), 332–383 (2001)
6. Chockler, G., Melamed, R., Tock, Y., Vitenberg, R.: Constructing scalable overlays for pub-sub with many topics. In: PODC (2007)
7. Chockler, G., Melamed, R., Tock, Y., Vitenberg, R.: Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In: DEBS (2007)
8. Curtmola, E., Deutsch, A., Ramakrishnan, K.K., Srivastava, D.: Load-balanced query dissemination in privacy-aware online communities. In: SIGMOD (2010)
9. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part II. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
10. Eugster, P.T., Felber, P., Guerraoui, R., Kermarrec, A.-M.: The many faces of publish/subscribe. ACM Comput. Surv. 35(2), 114–131 (2003)
11. Ghinita, G., Kalnis, P., Skiadopoulos, S.: Prive: anonymous location-based queries in distributed mobile systems. In: WWW (2007)
12. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
13. Liu, H., Ramasubramanian, V., Sirer, E.G.: Client behavior and feed characteristics of rss, a publish-subscribe system for web micronews. In: IMC, pp. 29–34 (2005)
14. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. In: ICDE (2006)
15. Opyrchal, L., Prakash, A., Agrawal, A.: Supporting privacy policies in a publish-subscribe substrate for pervasive environments. JNW (2007)
16. Raghavan, P., Thompson, C.D.: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. Combinatorica 7(4), 365–374 (1987)
17. Rao, W., Chen, L., Fu, A.: Stairs: Towards efficient full-text filtering and dissemination in dht environments. The VLDB Journal 20, 793–817 (2011)
18. Rao, W., Chen, L., Tarkoma, S.: Towards efficient filter privacy-aware content-based pub/sub systems. Accepted by IEEE Trans. Knowl. Data Eng. (2012)
19. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. 13(6) (2001)
20. Shang, N., Nabeel, M., Paci, F., Bertino, E.: A privacy-preserving approach to policy-based content dissemination. In: ICDE (2010)
21. Shikfa, A., Önen, M., Molva, R.: Privacy-preserving content-based publish/subscribe networks. In: Gritzalis, D., Lopez, J. (eds.) SEC 2009. IFIP AICT, vol. 297, pp. 270–282. Springer, Heidelberg (2009)
22. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5), 557–570 (2002)
23. Wang, T., Liu, L.: Privacy-aware mobile services over road networks. In: PVLDB, vol. 2(1), pp. 1042–1053 (2009)
24. Yao, A.C.-C.: Protocols for secure computations (extended abstract). In: FOCS, pp. 160–164 (1982)
25. Yuan, M., Chen, L., Rao, W., Mei, H.: A general framework for publishing privacy protected and utility preserved graph. In: ICDM, pp. 1182–1187 (2012)

Feel Free to Check-in: Privacy Alert against Hidden Location Inference Attacks in GeoSNs

Zheng Huo[1,⋆], Xiaofeng Meng[1], and Rui Zhang[2]

[1] School of Information, Renmin University of China, Beijing, China
{huozheng,xfmeng}@ruc.edu.cn
[2] Department of Computing and Information Systems, University of Melbourne, Australia
rui@csse.unimelb.edu.au

Abstract. Check-in services, one of the most popular services in Geo-Social Networks (GeoSNs) may cause users' personal location privacy leakage. Although users may avoid checking in places which they regard as sensitive, adversaries can still infer where a user has been through linkage of multiple background information. In this paper, we propose a new location privacy attack in GeoSNs, called *hidden location inference attack*, in which adversaries infer users' location based on users' check-in history as well as check-in history of her friends and similar users. Then we develop three inference models (*baseline inference model*, *CF-based inference model* and *HMM-based inference model*) to capture the hidden location privacy leakage probability. Moreover, we design a privacy alert framework to warn users the most probable leaked locations. At last, we conduct a comprehensive performance evaluation using two real-world datasets collected from Gowalla and Brightkite. Experiment results show the accuracy of our proposed inference models and the effectiveness of the privacy alert framework.

Keywords: Privacy-preserving, location privacy, location-based social network, inference attack.

1 Introduction

Geo-Social Network (GeoSN) is a kind of social network services where one or more individuals of similar interests or commonalities, connect with each other based on their geographical locations, such as, Foursquare, Gowalla and Brightkite. One of the most popular services in GeoSNs is **check-in service**, from which a user can report publicly which POI (Point of Interest) she has visited. Moreover, users can share their experiences at these places, such as giving commentary on the service or taste of food in a restaurant. The popularity of check-in service is not only because of its fun and online connectivity to friends, but also because of the material benefits. For example, free drinks or coupons are given to users who frequently check in a coffee shop. Therefore,

⋆ This research was partially supported by the grants from the Natural Science Foundation of China (No. 61070055, 91024032, 91124001); the National 863 High-tech Program (No. 2012AA010701, 2013AA013204); the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University (No. 11XNL010).

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 377–391, 2013.

users usually have a strong incentive to check in as more POIs as they can if there are no privacy concerns.

Generally speaking, users engaged in GeoSNs are in two categories: *real name users* and *pseudonym users*. Real name users use their true names as identifiers, people might easily associate their usernames in GeoSNs with individuals in reality. Pseudonym users prefer to use pseudonyms as their identifiers, which can avoid being directly associated with themselves in reality. For both kinds of users, publication of locations from check-in services raise severe privacy threats. Location is considered as private when it is itself sensitive or it may lead to disclosure of sensitive information. For example, by knowing a person is at a church during a religious festival, adversaries may infer the user's religious belief with a high probability. Even if the user is using her real name in GeoSNs, she may not be willing to expose her religious belief. On the other hand, location itself may act as a quasi-identifier. When location information is linked with other external knowledge, it may compromise user's anonymity and hence allow adversaries to associate the user's identity to sensitive information. People may argue that, users should be aware of location privacy leakage and avoid checking in sensitive places, or places that may expose their identities. This is undesirable for the following two reasons: (i). Users are eager to check in when they go to somewhere because of mental joy and material benefits. If too many POIs are considered as sensitive and could not be checked in, it may cause a negative impact on user experience; (ii). Although users may avoid checking in sensitive places (e.g., churches, hospitals or bars), they may not be aware of their *hidden location privacy leakage*, which is caused by linkage of users' check-in time, historical check-in behavior, as well as check-in behavior of other users, etc. That is to say, if a user does not check in a POI which she regards as sensitive, adversaries can still infer the probability of the user's visit to the POI. Leakage of hidden location privacy raises even more serious privacy threats to GeoSN users, since most of these POIs are where users intend to hide. A detailed example of hidden location privacy leakage is shown in Figure 1.

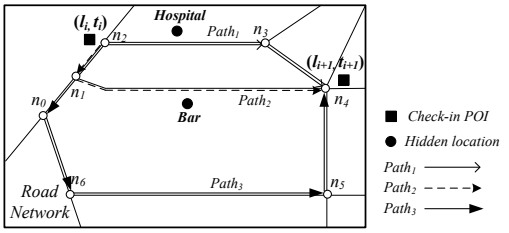


Fig. 1. An example of hidden location privacy leakage

As shown in Figure 1, a GeoSN user u_k checks in POI l_i at time t_i. When she reaches l_{i+1} at time t_{i+1}, she sends a check-in request at l_{i+1}. There might be privacy threat in this situation. Since the road network is publicly accessible, adversaries know that u_k might have taken one of the paths among $Path_1(n_2n_3n_4)$, $Path_2(n_1n_4)$ and $Path_3(n_1n_0n_6n_5n_4)$ to get to l_{i+1}. Since $Path_3$ is a long way, u_k might not prefer to travel on it. Suppose there is an AIDS hospital on $Path_1$ and a bar on $Path_2$. Although u_k doesn't check in the bar or the hospital for fear of privacy leakage, adversaries may

infer how likely she has visited one of them through linkage of multiple background information, summarized as follows.

- Geographical information, e.g., road networks, travel distances and check-in time, etc. If the time spent between l_i and l_{i+1} is less than or equal to the shortest time that one can reach l_{i+1} from l_i, then u_k definitely does not visit any other POIs during her movement.
- Historical information, e.g., u'_ks historical check-ins and her friends' historical check-ins, etc. If adversaries know the majority of users may check in the bar on $Path_2$ when they move from l_i to l_{i+1}, adversaries may infer that u_k probably visited the bar during her movement from l_i to l_{i+1}.
- Social information, e.g., social relationships, user closeness and similarity, etc. Suppose u_k checks in l_i and l_{i+1} at t_i and t_{i+1} respectively, a friend of u_k, say u_j checks in l_i around t_i and l_{i+1} shortly before t_{i+1}. Apparently, u_k was hanging out with u_j. If u_j checks in the bar on $Path_2$, adversaries may infer u_k also visited the bar although she didn't check in.

In this paper, we study the problem of **hidden location inference attack** in GeoSNs. The key challenge of our proposal is how to accurately and efficiently evaluate the hidden location leakage probability (HLPL-probability) and rank hidden locations based on the probability, as well as designing a privacy alert framework for GeoSN users. Contributions of this paper are summarized as follows.

- We propose a new privacy attack model, called *hidden location inference attack* in GeoSNs, from which users' most probable visited locations can be inferred through linkage of multiple background information.
- We propose three inference models to derive the HLPL-probability, they are baseline inference model, CF-based inference model and HMM-based inference model. The basic idea is that a user's visit behavior may be inferred by historical check-ins, as well as check-ins of her close friends and similar users. Then we employ Bayes' law, collaborative filtering and hidden Markov model respectively to derive the HLPL-probability.
- We design a novel privacy alert framework for GeoSN users to detect hidden location privacy leakage. We then implement our proposed inference models in this framework under road network constraints.
- Finally, we experimentally evaluate the inference models on two real-world datasets. Results show accuracy of the inference models and the effectiveness of the privacy alert framework.

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 formally defines concepts that we study in this paper. In section 4, we illustrate our proposed inference models. Section 5 proposes the privacy alert framework. Experimental results are presented in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

We conduct a brief review of existing studies on GeoSN data analysis, privacy-preserving and location prediction in GeoSNs.

Studies on GeoSN data analysis mainly focus on analyzing GeoSN users' behavioral patterns, social relationships, how social relationships and user mobilities affect each other, etc. Noulas et al. analyze user check-in dynamics, demonstrating how it reveals meaningful spatio-temporal patterns in Foursquare [6]. Studies in [8] mainly focus on how users' social relationships are affected by their distances and how users' social ties stretch across space. The above two papers study either spatial property or social property in GeoSNs separately. Cho et al. study both GeoSN user mobilities and social relationships jointly in [2]. It is claimed that more than 50% of GeoSN users' movements are affected by their friends. This fully support our research in this paper. Meanwhile, new privacy threats arise in the context of GeoSNs. In [3], authors notice that contents published by GeoSN users may lead to users' location privacy leakage or absence privacy leakage. The former is caused by uncontrolled exposure of users' geographical locations, while the latter concerns on the absence of a user from a geographic position. Authors propose a spatial generalization and publication delay method to solve these problems. Predicting users' social ties and locations is a new research topic in GeoSNs. Backstrom et al. predict home addresses of Facebook users based on addresses of users' friends [1]. It is claimed that this method dominates an IP-based one. More recently, [9] proposes to predict a trip's destination against data sparsity problems, then they develop a method which can select a minimum number of locations that a user has to hide in order to avoid privacy leak. The most related work to ours is proposed by Sadilek et al. in [7], which proposes to predict both social ties and locations of a user. In location prediction, a dynamic Bayesian network model is utilized to predict un-observed locations discretely.

Our proposal is different from [7] in the following two aspects: (i). We infer users' HLPL-probability between users' two observed check-ins, and rank hidden locations based on the probability. In our proposal, hidden locations are real-world semantic places where users might visit. However, situations in [7] are quite different, the inferred locations might be any geographic location. (ii). We propose a privacy alert framework under road network constraints, the most probable leaked hidden locations and corresponding HLPL-probability are pushed to users, which is obviously different from [7].

3 Preliminaries

A check-in activity consists of three factors: user, POI and check-in time. A GeoSN user may check in several POIs in her everyday life trajectories. Checks-in activities ordered by time constitute of a check-in sequence of the user.

Definition 1. ***(Check-in Sequence)*** *A user's check-in sequence* $\mathbb{S}$ *is a set of POIs ordered by check-in time* $\mathbb{S}=\{u_k, (l_1, t_1), \ldots, (l_i, t_i), \ldots, (l_n, t_n)\}$*, where* u_k *is the identifier of the user,* l_i *and* t_i *represent where* u_k *checks in and when she checks in respectively.*

In real-world, a user may repeatedly check in several POIs where she frequently visits. Therefore, a POI may appear more than once in one's check-in sequence, with different timestamps. A similar definition of check-in sequence is *visit sequence*, which consists of a set of doublets of POIs where the user has visited and corresponding timestamps. Note that, *visit sequence* is a superset of check-in sequence, since people may visit a number of POIs where they do not check in.

Definition 2. ***(Hidden Location)*** *Given POIs l_i and l_{i+1} which are checked in by user u_k at time t_i and t_{i+1} respectively. A hidden location l_m is a POI that might be visited by u_k when she moves from l_i to l_{i+1}, but not checked in by u_k at time t_m $(t_i < t_m < t_{i+1})$.*

Hidden location inference attack is a kind of location privacy attacks, from which adversaries can infer users' most probable visited hidden locations. One of the most serious consequences of hidden location privacy leakage is that users cannot control the privacy leakage, sometimes they do not even aware of their hidden location privacy leakage. Therefore, a privacy alert mechanism is required to warn users when their hidden location privacy is threatened.

4 Inference Models

The GeoSN service providers collect user generated data, including check-in sequences, social relationships and user profile information, etc. We make a widely accepted assumption that GeoSN service providers are un-trusted, which means service providers may analyze user data itself or may share it with the third party. Either of them may cause users' hidden location privacy leakage. Take user u_k as an example. Suppose u_k has checked in POI l_i and l_{i+1} at t_i and t_{i+1} respectively, l_i and l_{i+1} are called *observed locations*. The time interval between two check-ins can be represented as $\Delta t = t_{i+1} - t_i$. Given a hidden location l_m between two observed locations, we propose three inference models to infer the HLPL-probability, explained in the following subsections.

4.1 Baseline Inference Model

Users' check-in behaviors follow certain patterns or change periodically between two POIs, this can be obtained by aggregating users' history check-in behaviors [6]. Thus, adversaries can use majority users' behavioral patterns to "guess" how likely one would visit a POI. Given a hidden location l_m and the time interval between two check-ins Δt, the HLPL-probability can be denoted as a posterior probability $P(V_k^{i,m,i+1}|\Delta t)$, where $V_k^{i,m,i+1}$ denotes u_k's *sub-visit sequence* that contains l_i, l_m and l_{i+1} sequentially. Since u_k has already checked in l_i and l_{i+1}, the key point of $P(V_k^{i,m,i+1}|\Delta t)$ is how likely u_k would have visited l_m during her movement from l_i to l_{i+1}. According to Bayes' Law, $P(V_k^{i,m,i+1}|\Delta t)$ can be calculated as follows.

$$P(V_k^{i,m,i+1}|\Delta t) = \frac{P(\Delta t|V_k^{i,m,i+1})P(V_k^{i,m,i+1})}{P(\Delta t)} \tag{1}$$

Given a definite Δt for u_k, $P(\Delta t)$ is a constant, equation (1) can be simplified to $P(V_k^{i,m,i+1}|\Delta t) \approx P(V_k^{i,m,i+1}) \times P(\Delta t|V_k^{i,m,i+1})$. The baseline inference model aggregates users' historical check-in behaviors to draw an inference. Thus, $P(V_k^{i,m,i+1})$ can be calculated by equation (2).

$$P(V_k^{i,m,i+1}) = \frac{\sum_s C_s^{i,m,i+1}}{\sum_s C_s^{i,i+1}} \tag{2}$$

Equation (2) measures the fraction of sub-check-in sequences which check in l_m from all sub-check-in sequences that move from l_i to l_{i+1}. $C_s^{i,i+1}$ represents a sub-check-in sequence s that consists of l_i and l_{i+1} sequentially. $C_s^{i,m,i+1}$=1 if and only if one checks in l_i, l_m and l_{i+1} sequentially in s, otherwise, $C_s^{i,m,i+1}$=0.

In equation (1), Δt= t_{i+1} - t_i which represents the time interval between u_k's two observed check-ins also has spatial reach-ability meaning. If u_k cannot reach l_{i+1} from l_i within Δt, u_k definitely does not have time to visit any other locations, $P(V_k^{i,m,i+1}|\Delta t)$ must be zero. Note that, the time interval of two check-ins may not be exactly the same. Thus, we make use of an upper bound of Δt. Formally, we have $P(\Delta t|V_k^{i,m,i+1}) \leq P(\Delta t_s \leq \Delta t|V_k^{i,m,i+1})$, which can be calculated by equation (3).

$$P(\Delta t|V_k^{i,m,i+1}) \leq P(\Delta t_s \leq \Delta t|V_k^{i,m,i+1}) = \frac{\sum_s C_s^{i,m,i+1} \times P(\Delta t_s \leq \Delta t)}{\sum_s C_s^{i,m,i+1}} \tag{3}$$

where Δt_s represents the time interval between checking in l_i and l_{i+1} of sub-check-in sequence s. We then multiply $P(V_k^{i,m,i+1})$ and $P(\Delta t|V_k^{i,m,i+1})$ to get u_k's HLPL-probability of hidden location l_m.

• ***Weighted by Friend Closeness***

In practice, friends tend to have similar behaviors because they are friends and might share lots of common interests or always hang out together, thus leading to similar visit behaviors. Besides, friends in GeoSNs tend to have friendships in reality [4]. Previous studies show that friendships in GeoSNs have more influences over one's behaviors [8]. Thus, we refine the baseline inference model by introducing a parameter called *friend closeness*. Given two users u_j and u_k, u_j is one of u_k's friends, we adopt a formula in [10] to compute closeness of u_k and u_j, also shown as follows.

$$\omega_c(u_k, u_j) = \alpha \frac{|F_k \cap F_j|}{|F_k \cup F_j|} + (1-\alpha)\frac{|L_k \cap L_j|}{|L_k \cup L_j|} \tag{4}$$

where α is a turning parameter ranging within [0,1]. F_k and F_j represent friend sets of u_k and u_j respectively, L_k and L_j represent POI sets where u_k and u_j have checked in respectively. Friends with more common friends means they have close social ties; friends who show more similar check-in behaviors should have much similar tastes. We have an observation that, friends with close social ties and similar tastes may have higher probability to hang out together. Therefore, we give it a higher weight in the inference equation, thus, $P(V_k^{i,m,i+1})$ can be calculated by equation (5).

$$P(V_k^{i,m,i+1}) = \frac{\sum_s (1+\omega_c(u_k,u_j))C_s^{i,m,i+1}}{\sum_s (1+\omega_c(u_k,u_j))C_s^{i,i+1}} \tag{5}$$

where, $C_s^{i,m,i+1}$ and $C_s^{i,i+1}$ have no difference from equation (2) and $P(\Delta t|V_k^{i,m,i+1})$ is calculated by equation (3) as previously. By inclusion of friend closeness, the HLPL-probability is more affected by one's close friends than other normal users. It should be noted that, the baseline inference model can deal with one hidden location each time, if there are multiple hidden locations among two observed check-ins, it should be executed several times.

4.2 CF-Based Inference Model

Collaborative Filtering (CF) is a method of making predictions about interests of a user by collecting preferences or taste information of many other users, a user's *rating on an object* can be inferred through ratings of her similar users. In GeoSNs, similar users who might share a lot of common interests tend to have similar visit behaviors. Accordingly, adversaries can infer the HLPL-probability using CF. In order to calculate user similarity, we introduce a concept called *visit probability sequence* to evaluate users' visit probabilities to a set of POIs.

Definition 3. ***(Visit Probability Sequence)*** *Given a sub-check-in sequence* $s=\{l_1, l_2, \ldots, l_n\}$. $u'_k s$ *visit probability sequence of* s *is a set of probabilities* $PV_{uk}=\{PV^1_{uk}, PV^2_{uk}, \ldots, PV^n_{uk}\}$, *where* $PV^i_{u_k} \in [0,1]$ *denotes* $u'_k s$ *visit probability to POI* l_i.

User similarity between u_k and u_j can be calculated by the cosine similarity of their visit probability sequences, as shown in equation (6).

$$sim(u_k, u_j) = \frac{\sum_i PV^i_{u_k} PV^i_{u_j}}{\sqrt{\sum_i {PV^i_{u_k}}^2}\sqrt{\sum_i {PV^i_{u_j}}^2}} \tag{6}$$

In equation (6), u_k and u_j do not need to have friendships in GeoSNs. $PV^i_{u_k} \in [0,1]$ and $PV^i_{u_j} \in [0,1]$ denote u_k and u'_js visit probability to POI l_i respectively. A special case of visit probability sequence is the check-in sequence, in which $PV^i_{uk}=1$ if u_k checks in l_i and $PV^i_{uk}=0$ if u_k does not. For each user u_k, we utilize two matrices to calculate u'_ks visit probability to hidden locations, as shown in Figure 2. Matrix S is a user-user matrix, each value s_{kj} in S represents the similarity between u_k and u_j, calculated by equation (6). Matrix U is a location-user matrix, each value u_{kn} represents u'_ks visit probability to l_n. Users in matrix U are the top-n most similar users of u_k and locations in matrix U are u'_ks hidden locations between two observed check-ins. Matrix U is initialized by u'_ks check-in sequence. We then calculate user similarity through their check-in sequences, and put user similarities into matrix S for initialization. In matrix U, u'_ks visit probabilities to hidden locations are the missing values. Equations (7) and (8) illustrate how to infer the missing values using classical CF method.

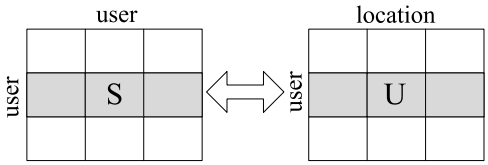


Fig. 2. Matrices in CF-based inference model

$$r_{u_k,l_n} = k \times \sum_{u_j \in S_k} sim(u_k, u_j) \times r_{u_j,l_n} \tag{7}$$

$$k = \frac{1}{\sum_{u_j \in S_k} |sim(u_k, u_j)|} \tag{8}$$

where r_{u_k,l_n} represents u_k's visit probability to hidden location l_n, the higher the value is, more probable that u_k has visited l_n. S_k denotes u_k's similar user set, r_{u_j,l_n} is u_j's visit probability to location l_n, where, u_j is one of u_k's similar users. In equation (7), the similarity between u_k and u_j, $sim(u_k,u_j)$ is used as a weight, more similar u_k and u_j are, more weight r_{uj,l_n} will carry in predicting r_{uk,l_n}. When we say u_j is a similar user of u_k, we mean $sim(u_k,u_j) > 0$. After initialization of matrices S and U, we begin to calculate u_k's ratings on hidden locations in matrix U. We get u_k's ratings on hidden locations in matrix U through repeatedly calculating the following steps, until the matrices converge.

- Calculate users' visit probabilities to hidden locations through equation (7) and (8), then update matrix U.
- Calculate user similarities using equation (6), then update matrix S.

We then compute the posterior probability to derive the HLPL-probability according to equation (1). Given Δt, the HLPL-probability $P(V_k^{i,m,i+1}|\Delta t)$ can be calculated by equation (9).

$$P(V_k^{i,m,i+1}|\Delta t) = r_{u_k,l_m} \times \frac{\sum_{u_j \in S_k} C_{u_j}^{i,m,i+1} P(\Delta t_{u_j} \leq \Delta t)}{\sum_{u_j \in S_k} C_{u_j}^{i,i+1}} \tag{9}$$

In equation (9), S_k denotes u_k' similar user set, u_j is one of u_k's similar users. Δt_{uj} is the time interval between u_j checks in l_i and l_{i+1}. Once CF-based inference model is executed, the missing values in matrix U is completed, thus, u_k's HLPL-probability for each hidden location is derived.

4.3 HMM-Based Inference Model

Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with hidden states. If one can make predictions for the future of the process based solely on its present state, it is a Markov process. In the previous section, we hold the opinion that users' visit probability to a POI can be inferred though historical information, but this does not mean users' next location can be predicted based on all historical check-in sequences. Actually, given the moving speed and road network conditions, u_k's current location can be inferred solely based on the previous state (where u_k checks in, as well as where u_k's friends and similar users check in, etc.). Therefore, we utilize Hidden Markov Model (HMM) to represent users' transitions among hidden locations.

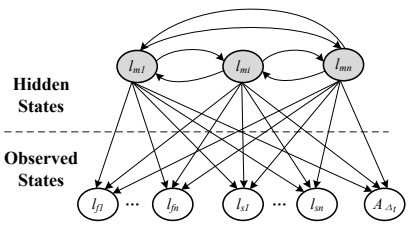


Fig. 3. HMM-based Inference Model

Given two POIs l_i and l_{i+1} which u_k has checked in, we build a HMM-based inference model for each user, as shown in Figure 3. Nodes in white are observed nodes, while nodes in gray are hidden nodes which correspond to hidden locations that u_k might visit during two observed check-ins. Here we just list three hidden nodes as an example. The observed nodes l_{f1} to l_{fn} are where u'_k s top-n closest friends check in within time Δt (time interval between two observed check-ins). The observed nodes l_{s1} to l_{sn} represent where u'_ks top-n similar users check in within time Δt, but without users who appear in the top-n closest friends. Node $A_{\Delta t}$ is the attribute of Δt, e.g., weekends or weekdays, morning, noon or evening of a day. All the observed nodes and hidden nodes are discrete.

We train the HMM model using the real-world datasets through Expectation - Maximization (EM) algorithm. In the expectation step, we complete the dataset using the current parameters $\theta^{(t)}$ (always start with random values) to calculate the expected values of the log likelihood function, denoted as equation (10).

$$Q(\theta|\theta^{(t)}) = E_{H|O,\theta^{(t)}}[logP(O,H|\theta)] \tag{10}$$

where O is a set of observed nodes, while H is a set of hidden nodes. In the Maximization step, we use the *completed* dataset to find a new maximum likelihood estimate for a vector of unknown parameter θ, the target function is denoted as equation (11).

$$\theta^{(t+1)} = \arg_\theta maxlog(Q(\theta|\theta^{(t)})) \tag{11}$$

where $\theta^{(t+1)}$ is the final parameters of the HMM model. The algorithm repeats the two steps until it converges. After we get a set of parameters of the HMM model, transition probabilities among hidden locations are derived, which happens to be the HLPL-probability in our settings. Then we infer the hidden location sequence that is most likely to have generated the observed locations. We use Viterbi algorithm to efficiently infer the hidden location sequence. The target function can be denoted as equation (12).

$$H^* = \arg maxlog(P(H|O)) \tag{12}$$

Equation (12) represents the conditional probability of the hidden node H, given observations O. We apply dynamic programming in this problem, thus achieving a time complexity of $O(|O| \times |H|^2)$, where $|O|$ is the number of observations and $|H|$ is the number of hidden nodes.

5 Hidden Location Privacy Alert Framework

5.1 System Model

Our hidden location privacy inference framework is based on the *client-inference server-GeoSN server* model as depicted in Figure 4.

There are two main components in the inference server, namely, *hidden location finder* and *probability estimator*. Hidden location finder finds out all the hidden locations between l_i and l_{i+1} for each user, and calculates the shortest road network distance between l_i and l_{i+1}. Probability estimator is in charge of estimating users' HLPL-probabilities using the inference models we previously defined and rank hidden locations based on the probability. Each client u_k, sends her check-in request $C_k(l_{i+1}, t_{i+1})$ at location l_{i+1} along

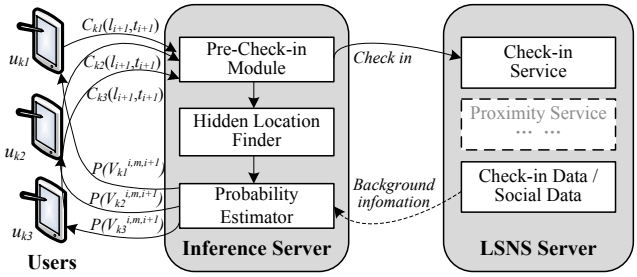


Fig. 4. System Architecture

with check-in time t_{i+1} to the *pre-check-in module*. The received requests are sent to the *hidden location finder*, which derives u_k's latest check-in location l_i and check-in time t_i. Hidden locations between two check-ins are found out by *hidden location finder*, and passed to *probability estimator*. The background information for inference is acquired from *GeoSN server*. At last, the most probable leaked hidden locations with corresponding HLPL-probability are pushed to users, as a privacy alert. Users could decide whether to check in or not according to their privacy preferences.

5.2 Algorithms

The inference models and the privacy alert framework are implemented in the road network space rather than free space. In our settings, a road network is represented as an undirected graph $G(V,E)$; each road segment is represented by edges in G, intersections of the routes are represented by vertices in V. GeoSN users' moving speed is bounded by the average maximum moving speed of road segments.

Given a user u_k, we prune two situations that u_k may not visit any hidden location: (i). If $\Delta t \leq \frac{Dist(l_i,l_{i+1})}{v_{max}}$, u_k must definitely not visit any hidden location, since the time is not enough for u_k to visit any location when she moves from l_i to l_{i+1}. Where $Dist(l_i,l_{i+1})$ is the shortest road network distance between l_i and l_{i+1}, v_{max} is the average maximum moving speed of routes $l_i \rightarrow l_{i+1}$. (ii). If $\frac{Dist(l_i,l_{i+1})}{v_{max}} < \Delta t < Min(\Delta t_j)$, u_k may probably not visit hidden locations, since the time spent between l_i and l_{i+1} is less than any sub-check-in sequence that begins with l_i and ends with l_{i+1}. Where $Min(\Delta t_j)$ is the minimum time cost between two observed check-ins for any user. In the last case, if $\Delta t \geq Min(\Delta t_j)$, we need to find out all the hidden locations between l_i and l_{i+1}, which is accomplished by algorithm **FindHiddenLocation**, then we utilize three inference models to calculate the HLPL-probability (due to space limitation, the details will not be explained again in this subsection).

Finding out all the un-checked-in POIs between l_i and l_{i+1} as hidden locations is unrealistic and useless, since POIs located on a far or unpopular path may rarely be visited by GeoSN users. Therefore, we only take two kinds of POIs into consideration: (i). POIs on the shortest path between l_i and l_{i+1}. (ii). POIs on the popular paths between l_i and l_{i+1}. The details are shown in Algorithm 1.

In Algorithm 1, we utilize the A^* algorithm to find the shortest path between l_i and l_{i+1}, since the shortest path is always a choice to users when they go to somewhere (line

Algorithm 1. FindHiddenLocation

Input : POIs l_i and l_{i+1}; road network $G(V,E)$; a set of history check-in sequences $\overline{\mathbb{S}}$;
Output: A set of hidden locations L_m; length of the shortest path $Dist(l_i, l_{i+1})$;

1 $Path_{sh} \leftarrow A^*(G, l_i, l_{i+1})$;
2 $L_m \leftarrow$ POIs on $Path_{sh}$;
3 $Dist(l_i,l_{i+1}) \leftarrow$ distance between l_i and l_{i+1} on $Path_{sh}$;
4 Tag each check-in sequence in $\overline{\mathbb{S}}$ as *unscanned*;
5 **while** *exists a check-in sequence in* $\overline{\mathbb{S}}$ *unscanned* **do**
6 Scan the next check-in sequence $\mathbb{S}$;
7 Tag $\mathbb{S}$ as scanned;
8 **if** $count(l_i \xrightarrow{\Delta t_1<\delta t} l_m \xrightarrow{\Delta t_2<\delta t} l_{i+1}) > \delta_p$ **then**
9 $L_m \leftarrow L_m \bigcup l_m$
10 **Return** L_m and $Dist(l_i, l_{i+1})$;

1-3). Without loss of generality, we scan each check-in sequence in $\overline{\mathbb{S}}$ and identify POIs that frequently appear on a popular path between l_i and l_{i+1}, then put them into L_m. If the support of a check-in pattern $l_i \xrightarrow{\Delta t_1<\delta t} l_m \xrightarrow{\Delta t_2<\delta t} l_{i+1}$ is larger than the support threshold δ_p, l_m can be regarded as a hidden location (line 4-9). We have a key observation that the temporally consecutive check-ins of u_k could signal correlations between two POIs, but as the time interval increases, that two check-ins are not strictly consecutive. In order to solve this problem, we make use of a time threshold δ_t, which is used to estimate whether a check-in pattern is valid or not. Δt_1 and Δt_2 represent the corresponding check-in time intervals. At last, hidden location set L_m and the shortest path distance between l_i and l_{i+1} are returned.

6 Experiments

The design of the experiments aims to achieve the following goals: (i) Analyze properties of the experiment datasets; (ii) Learn the performance of each inference model; (iii) Learn the effectiveness of the privacy alert framework.

6.1 Experimental Setup

We run our experiments on two real-world GeoSN datasets, made available by Cho et al. [2]. The datasets collect users' social relationships and check-in behaviors from Feb. 2009 to Oct. 2010 in Gowalla and Apr. 2008 to Oct. 2010 in Brightkite. We also obtain the road network data of California, which contains 21,693 edges and 104,407 POIs. We pre-process both datasets, check-ins in California and related users are left. As a result, the Gowalla dataset contains 15,116 users and 675,809 check-ins, while the Brightkite dataset contains 9,435 users and 541,169 check-ins. Detailed information about both datasets are shown in Table 1.

Besides, we randomly select 22,495 pairs of consecutive check-ins, distributions of check-in time interval and distance between two consecutive check-ins are shown

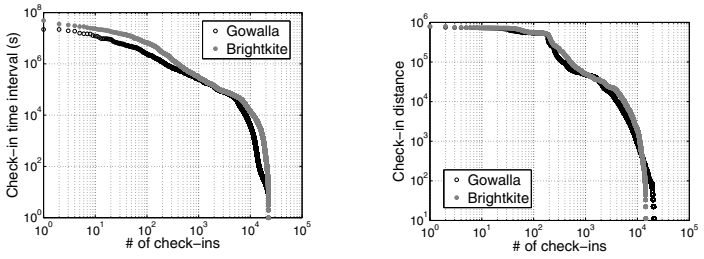
Table 1. Detailed information about Gowalla and Brightkite

Total check-ins		**Total users**		**Area size** (km^2)		**Avg. check-in interval**(h)	
Gowalla	Brightkite	Gowalla	Brightkite	Gowalla	Brightkite	Gowalla	Brightkite
675,809	541,169	15,116	9,435	443,556	443,556	40.31	56.81
Density($user/km^2$)		**Check-in / user**		**Check-in / POI**		**Avg. check-in distance**(km)	
Gowalla	Brightkite	Gowalla	Brightkite	Gowalla	Brightkite	Gowalla	Brightkite
0.03	0.02	44.71	57.36	6.29	6.10	19.02	15.39

in Figure 5. It can be seen that, about 60% consecutive check-ins occur within a time interval of $10^4 s$ in Gowalla and 33% in Brightkite, and about 55% consecutive check-ins occur within a distance of $1km$ for both datasets.

6.2 Performance Metrics

Since real hidden locations are invisible in check-in datasets, we make an assumption of our dataset that a user visits a POI if and only if she checks in the POI. Then two hidden location datasets are generated in our experiments. Given a user u_k, *hidden location set I* is generated by marking off a part of POIs that u_k has checked in; *hidden location set II* is generated by adding POIs which are geographically located between l_i and l_{i+1} that u_k does not check in. We then evaluate four performance metrics: (i) The ratio of recovered hidden locations to the number of hidden locations in hidden location set I, represented as *true positive rate*; (ii) Average HLPL-probability of hidden location set I, represented as $AVG@I$; (iii) The ratio of recovered hidden locations to the number of hidden location set II, represented as *false positive rate*; (iv) Average HLPL-probability of hidden location set II, represented as $AVG@II$. The *true negative rate* and *false negative rate* can be derived from the above metrics.



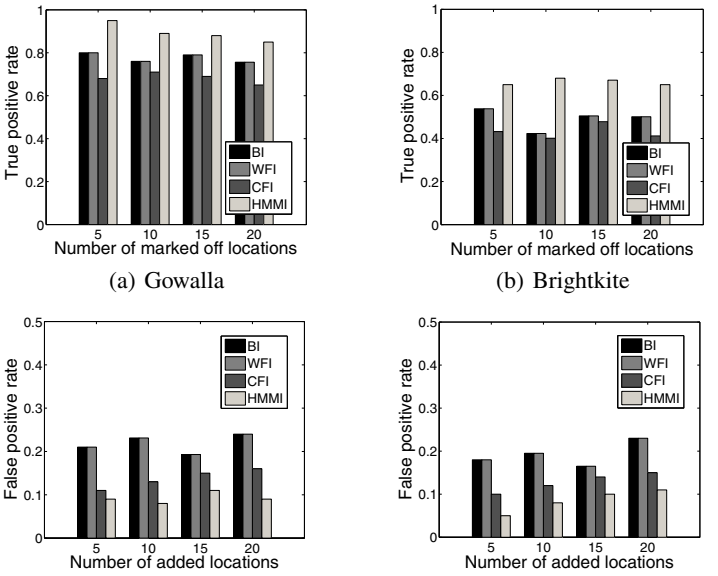
(a) Check-in interval (b) Check-in distance

Fig. 5. Datasets attribute

6.3 Study on Accuracy

We evaluate the performance of following inference models: baseline inference model, denoted by BI; baseline inference model weighted by friend closeness, denoted by WFI; CF-based inference, denoted by CFI; HMM-based inference, denoted by $HMMI$. We measure $AVG@I$, $AVG@II$, *true positive rate* and *false positive rate* on both datasets. For each inference model, we randomly select 3,000 users and choose l_i and l_{i+1} in each

user's check-in sequence. The selection of l_i and l_{i+1} should satisfy a constraint that time interval Δt is bounded by 5 times of average check-in time interval (*since more than 96.6% consecutive check-ins have a time interval within 5 times of average check-in time interval for Gowalla and 97.1% for Brightkite*). For each user, we randomly mark off 5, 10, 15 and 20 visited POIs between l_i and l_{i+1} as *hidden location set I*, and add 5, 10, 15 and 20 un-checked-in POIs that geographically locate between l_i and l_{i+1} as *hidden location set II*. In the HMM-based inference model, we use Matlab toolbox for HMM [5] to train the HMM model. We utilize historical check-ins of one's top-5 close friends and top-5 similar users as training set.



(a) Gowalla (b) Brightkite

(c) Gowalla (d) Brightkite

Fig. 6. Performance estimation on true / false positive rate

The experimental results in Figure 6 and Figure 7 indicate that, performance on all the metrics are not seriously affected by the increasing number of the marked off / added locations, showing that our inference models work with large number of hidden locations. In Figure 6, the higher the *true positive rate* is, the better the inference model performs, and the *false positive rate* represents just the opposite. It can be seen that the HMM-based inference model always exhibits the best performance in terms of *true positive rate* and *false positive rate*, under all values of marked off / added locations, showing the strength of combining locations of one's friends and similar users to infer one's hidden location. Besides, the learning technique which captures the transition probability between one's visited locations also makes the HMM-based inference model outstanding. *BI* and WFI have the same *true positive rate* and *false positive rate*, since WFI is a special case of *BI*, it can recover hidden locations which *BI* can also do. It should be noted that, although the *BI* and WFI performs not bad on true positive rate, but the performance differences among users are quite huge, about 16.1% users have

zero true positives on Gowalla, and about 18% on Brightkite. This is because some users' check-in sequences are un-popular, making it hard to infer wether the user has visited the hidden locations through historical check-in sequences. The HMM-based inference model is much better, only less than 5% users have zero true positives on both datasets. Another drawback of *BI* and WFI inference models is that, they have high false positive rate, especially in areas with dense check-in activities. It should be noted that, CF-based inference model have both low *true positive rate* and low *false positive rate*, which is also caused by the data sparsity problem. Since there are few similar users that visit a hidden location in a given time span, thus resulting in the poor performance on true positive rate.

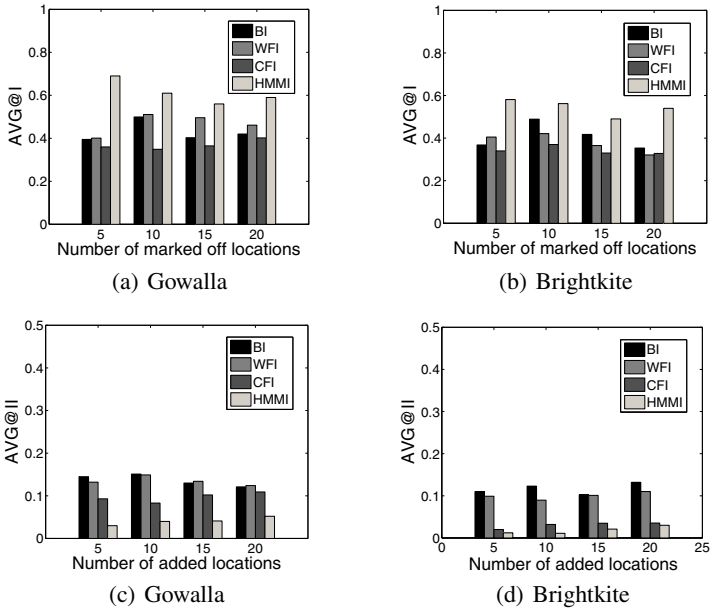


Fig. 7. Performance estimation on average HLPL-probability

In Figure 7, the higher the $AVG@I$ value is, the better the inference model performs, while $AVG@II$ value just represent the opposite. It can be seen that, the $AVG@I$ value dominate the $AVG@II$ value on all numbers of marked off / added locations, showing that our inference models can recover the hidden locations and derive the HLPL-probabilities effectively. WFI and *BI* have the same true positive rate and false positive rate, the difference is that, WFI performs slightly better on both metrics than that of *BI*, since both datasets are sparse (as shown in Table 1), the co-appearance of two users does not always happen, making the WFI model performs not as well as expected. The HMM-based inference model performs the best on $AVG@I$ and $AVG@II$ among all inference models, the $AVG@I$ value of $HMMI$ reaches almost 60% and the $AVG@II$ value is even below 5% on all values of marked off / added locations. We do not test the precision of hidden location visit sequences generated by $HMMI$, since it is not our concern in this paper.

6.4 Study on Efficiency

We study the efficiency of the privacy alert framework, showing that the average response time from sending a check-in request to the inference server to get a privacy alert is in minutes level. Although the response time does not satisfy the real-time applications well, the framework is still available, since the response time is much shorter than users' average stay time at a place. How to improve the efficiency of the privacy alert framework will be considered in the future work. One possible solution is to separate the whole procedure into the *training phase* and the *inference phase* [9]. The training phase can be performed offline, which may sharply reduce the response time.

7 Conclusions and Future Work

Check-in service in GeoSNs raises serious privacy concerns. In this paper, we propose a new privacy attack called hidden location inference attack. To accurately derive the HLPL-probability, we propose three inference models. At last, we design a privacy alert framework to warn users the most probable leaked hidden locations/visit sequence. We evaluate the inference models on two real-world datasets, experiment results indicate that the HMM-based inference model has the best performance among all.

Our future work are in two-fold. First, we will investigate how to improve the performance of the inference models against data sparsity problem. Second, we plan to study how to protect users' hidden location privacy against the attack models. One of the possible solutions could be adding fake check-in POIs when a check-in activity happens. While the challenging problems are how to balance the privacy and utility of users' check-in, as well as considering users' privacy preferences, etc.

References

1. Backstrom, L., Sun, E., Marlow, C.: Find me if you can: improving geographical prediction with social and spatial proximity. In: WWW 2010, pp. 61–70 (2010)
2. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: KDD 2011, pp. 1082–1090 (2011)
3. Freni, D., Vicente, C.R., Mascetti, S., Bettini, C., Jensen, C.S.: Preserving location and absence privacy in geo-social networks. In: CIKM 2010, pp. 309–318 (2010)
4. Gruzd, A., Wellman, B., Takhteyev, Y.: Imagining twitter as an imagined community (2011)
5. MIT. Hidden markov model (hmm) toolbox for matlab, `http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html`
6. Noulas, A., Scellato, S., Mascolo, C., Pontil, M.: An empirical study of geographic user activity patterns in foursquare. In: ICWSM 2011 (2011)
7. Sadilek, A., Kautz, H.A., Bigham, J.P.: Finding your friends and following them to where you are. In: WSDM 2012, pp. 723–732 (2012)
8. Scellato, S., Noulas, A., Lambiotte, R., Mascolo, C.: Socio-spatial properties of online location-based social networks. In: ICWSM 2011 (2011)
9. Xue, A.Y., Zhang, R., Zheng, Y., Xie, X., Huang, J., Xu, Z.: Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In: ICDE 2013 (2013)
10. Ye, M., Yin, P., Lee, W.-C., Lee, D.L.: Exploiting geographical influence for collaborative point-of-interest recommendation. In: SIGIR 2011, pp. 325–334 (2011)

Differentially Private Set-Valued Data Release against Incremental Updates

Xiaojian Zhang[1,2,⋆], Xiaofeng Meng[1], and Rui Chen[3]

[1] School of Information, Renmin University of China, Beijing, China
[2] School of Computer and Information Engineering, Henan University of Economics and Law
{xjzhang82,xfmeng}@ruc.edu.cn
[3] Department of Computer Science, University of British Columbia, Vancouver, Canada
rui_chen@cs.ubc.ca

Abstract. Publication of the private set-valued data will provide enormous opportunities for counting queries and various data mining tasks. Compared to those previous methods based on partition-based privacy models (e.g., *k*-anonymity), differential privacy provides strong privacy guarantees against adversaries with arbitrary background knowledge. However, the existing solutions based on differential privacy for data publication are currently limited to static datasets, and do not adequately address today's demand for up-to-date information. In this paper, we address the problem of differentially private set-valued data release on an incremental scenario in which the data need to be transformed are not static. Motivated by this, we propose an efficient algorithm, called *IncTDPart*, to incrementally generate a series of differentially private releases. The proposed algorithm is based on top-down partitioning model with the help of item-free taxonomy tree and update-bounded mechanism. Extensive experiments on real datasets confirm that our approach maintains high utility and scalability for counting query.

Keywords: Differential privacy, set-valued data, incremental updates.

1 Introduction

Set-valued data, in which a set of items are associated with an individual, is common in database ranging from web query logs, to credit card transactions, and to shopping transaction databases of customers' behavior. Publishing and sharing set-valued data is important, since they enable researchers to analyze and explore interesting patterns and knowledge. For example, revealing strong correlations and trends from collected patient health records can be a valuable knowledge base for society; for a retail company, analyzing common customer behavior from online shopping data can provide useful information for advertising. However, such data usually contains specific sensitive information (e.g., pregnancy test, health care), and directly releasing raw data could violate individual privacy and may result in unveiling the identity of the individual associated with a particular transaction. To prevent such information leakage, set-valued

⋆ This research was partially supported by the grants from the Natural Science Foundation of China (No. 61070055, 91024032, 91124001); the National 863 High-tech Program (No. 2012AA010701, 2013AA013204); the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University(No. 11XNL010).

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 392–406, 2013.

data must be sanitized before release. Previous works have been made in addressing the problem of set-valued data publication. Terrovitis et al. [1] propose a local and global recoding method k^m-anonymity, and He and Naughton [2] enhanced [1] by a top-down, partition-based approach to handling the same question. However, recent efforts have shown that the above partition-based privacy models are vulnerable to many types of privacy attacks, such as composition attack [3], and foreground knowledge attack [4]. Recently, differential privacy [5] has emerged as one of the most promising models for releasing different types of private data, because it can provide strong privacy guarantees against adversaries with arbitrary background knowledge (this claim may not be valid in some cases where there exist correlations among records [6], but in this paper we assume that records are independent of each other). The main idea of differential privacy is to inject noise into a dataset so that an adversary cannot decide whether a particular record is included in the dataset or not. The noise level is controlled by privacy budget ε. There have been a few set-valued data publishing algorithms proposed in the recent work, such as [7, 8], that efficiently publish such data under differential privacy. These approaches, however, only deal with static set-valued data releases. That is, all these approaches assume that they work in a one-time fashion: sanitize the entire database and obtain the statistic information. This assumption often heavily limits the applicability of these differentially private methods, as in many dynamic applications set-valued datasets are updated incrementally.

Example 1. Consider a supermarket's store T, shown in Table 1, which is required to share the transactional items purchased by its various customers with market researchers. In order to protect customers' privacy, the supermarket sanitize all the items prior to releasing. At first glance, the task seems reasonable straightforward, as existing techniques in [7, 8] can efficiently anonymize the items. The challenge is, however, that T is growing daily due to the appending of newly purchased items for existing customers and/or insertion of new shopping items for new customers, shown in Table 2, and it is critical for the market researchers to receive up-to-date items in timely manner.

Table 1. The original database T

TID	Items Purchased
T_1	$\{apple, banana, cherry, melon\}$
T_2	$\{apple, cherry\}$
T_3	$\{cherry, melon\}$
T_4	$\{apple, melon\}$

Table 2. The Incremental Update ΔT_1

TID	Items Purchased
T_5	$\{banana, cherry, melon\}$
T_6	$\{melon, cherry\}$
T_7	$\{apple, melon\}$

Due to the inherent dynamics and high-dimensionality of set-valued data in the context of incremental updates, it is challenging to apply differential privacy to incrementally publishing set-valued data. In the real world transactional data usually arrives in batches to update original database incrementally. Thus, a differentially private mechanism must periodically update the published statistics as new data items are inserted or removed. We assume the coming update is large enough that it can support differential privacy.

In order to employ the existing methods to publish the updates, two straightforward solutions may be developed.

Solution 1: Sanitizing Incremental Updates. Support a series of updates arriving. This solution is to sanitize and publish each update independently so that it satisfies differential privacy. For example, we employ DiffPart method [7] to separately release Table 1 and Table 2. Then researchers can merge multiple those released datasets together for comprehensive analysis. Although straightforward, the major drawback of this solution is when researchers merge multiple noisy statistics, the error is accumulated in the merged results. So the results become noisier over time, which may lead to lower data quality.

Solution 2: Publishing Sanitization of Current Versions. This solution is to sanitize and publish the entire dataset whenever the dataset is augmented with new updates (e.g., merging the update ΔT_1 to T). In this way, researchers are always provided with up-to-date statistics information. Although this can be easily accomplished by using the existing methods, there are two significant drawbacks. First, K such updates will raise the privacy budget to $K\varepsilon$. This means that the more updates we have, the higher the amount of noise we need to add to each release. Another, if the number of updates is infinite, then ε-differential privacy will be eroded.

In the above solutions, the error increases with the number of updates, either because the noise accumulates (as in solution 1), or because the amount of noise relies on the number of updates (as in solution 2). This means that we cannot have an infinite number of updates. In other words, streaming approaches proposed in [10, 9] cannot be applied in our scenario because data stream has its own characteristics such as dynamics, continuity, and infinity. To tackle the above challenges, we investigate the problem of how to publish set-valued data against incremental updates while maintaining ε-differential privacy. First, we design a *update-bounded mechanism* to limit the number of updates which can help to reduce the error caused by the above two solutions. Second, based on update-bounded mechanism, we propose an efficient algorithm, called *IncTDPart* (Incrementally Top-Down Partitioning manner) to release set-valued data with the help of item-free taxonomy tree. In our algorithm, we incrementally maintain a tree structure, called *TBP-Tree* (Taxonomy-Based Partitioning Tree), in which leaf nodes store *p-sum* value (i.e., accumulated noisy counts) that can be used to construct a release after each update. Third, extensive experiments on several real datasets demonstrate that the proposed methods generate high utility for incrementally counting queries and scales to large datasets.

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 briefly overviews ε-differential privacy and problem statement. Section 4 proposes a differentially private algorithm *IncTDPart* to support set-valued data releases against incremental updates. The experimental evaluation of our methods is presents in Section 5, and Section 6 concludes our work.

2 Related Work

The notion of differential privacy was presented by Dwork et al in [5]. The same authors also propose the addition of Laplace noise to guarantee differential privacy [11]. Work

on differential privacy has initially focused on answering statistical queries (e.g., count queries, range queries). However, a few recent works started addressing non-interactive data release that achieves differential privacy such as histograms publication [13, 12], and search logs releases [14]. McSherry et al [15]. present differentially private recommendation algorithms in Netflix prize competition. More recently, several works studied differentially private mechanism for releasing set-valued data [8, 7]. Chen et al. [7] presented the publishing of set-valued data while satisfying differential privacy, and in [8] they studied differentially private transit data publication. They present algorithms in [8, 7], which partition the set-valued data in top-down fashion guided by taxonomy tree, and release the noisy counts of the set-valued data at leaf nodes. Their methods generate synthetic set-valued data which can support counting queries. However, the existing publication approaches are currently limited to static datasets, do not adequately address the incremental updates. Among the existing approaches, the ones most related to ours are by Chan et al. [10] and Dwork, et al. [9], which continuously release statistics in the context of data streams. Due to the characteristics of data stream itself, their methods are not favorable to releasing set-valued datasets against updates.

In addition, there is a series of works [16, 18, 17] which are based on partition-based privacy models (e.g., k-anonymity) for incrementally releasing relational databases. The work [16] is among the first to identify possible attacks in the dynamic scenario. This work analyzes various inference channels that may exist in multiple anonymized datasets and discusses how to avoid such inferences. Xiao et al. [17] propose the novel m-invariance framework. This simple yet elegant method is the first work that successfully anonymizes a fully dynamic dataset. To enhance the methods in [17], He et al. [18] propose a graph-based anonymization algorithm to cope with equivalence attack. However, recent works have shown that these methods based on partition-based models are much weaker privacy notion than differential privacy.

3 Preliminaries

Let $I=\{I_1, I_2, ..., I_{|I|}\}$ be the universe of items, where $|I|$ is the size of the universe. The $T=\{t_1, t_2, ..., t_{|T|}\}$ denotes the initial set-valued table as it is created before updating, where each record $t_i \in T$ is a non-empty subset of I. Let ΔT_1, ΔT_2,... present the incremental updates (insertions only in this paper) to the table T. We assume that in the series of tables T, ΔT_1, ΔT_2,..., the item domain is fixed, that is, the universe of items is not changed. Suppose the item universe $I=\{I_1, I_2, I_3, I_4\}$, Table 3 presents an example of initial set-valued database T. Table 4 and Table 5 present the incremental updates ΔT_1, ΔT_2 to Table 3.

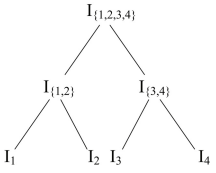


Fig. 1. IFT-Tree

Table 3. initial T

TID	Items
t_1	$\{I_1, I_2\}$
t_2	$\{I_2\}$
t_3	$\{I_1\}$

Table 4. update ΔT_1

TID	Items
t_4	$\{I_1, I_2, I_3, I_4\}$
t_5	$\{I_2, I_4\}$
t_6	$\{I_2\}$

Table 5. update ΔT_2

TID	Items
t_7	$\{I_1, I_2, I_3, I_4\}$
t_8	$\{I_2, I_3, I_4\}$
t_9	$\{I_1, I_2\}$

3.1 Item-Free Taxonomy Tree

In this paper, we assume that the series of tables T, ΔT_1, ΔT_2,..., are associated with a single taxonomy tree. In classical generalization mechanism, taxonomy tree relies on the original semantic of generalization to map several differential items to a single value in the destination nodes. In our method, we release only original items and counts, regardless of their semantics. Therefore, the taxonomy tree could be item free.

Definition 1. *(**Item-Free Taxonomy Tree**). An Item-Free Taxonomy Tree (IFT-Tree) is a taxonomy tree, whose internal nodes consist of their leaves, not necessarily to take into account the semantic generalization of the leaves.*

For example, Fig. 1 presents an item-free taxonomy tree for Table 3, Table 4, and Table 5, and one of its internal nodes $I_{\{3,4\}}$=$\{I_3, I_4\}$. An item can be mapped to an internal node if it is in the node's set. In this example, items $\{I_1, I_2\}$ can be generalized to $I_{\{1,2\}}$, items $\{I_3, I_4\}$ can be generalized to $I_{\{3,4\}}$, and the two sets $I_{\{1,2\}}$, $I_{\{3,4\}}$ can be further generalized to $I_{\{1,2,3,4\}}$.

3.2 Differential Privacy

Differential privacy, in general, guarantees that changing or removing any record from a database has negligible impact on the output of any analysis based on the databases. Therefore, an adversary will learn nothing about an individual, regardless of whether her record is present or absent in the database. Formally, in the context of incremental updates, differential privacy [5] is defined below.

Definition 2. *(**ε-Differential Privacy**). A randomized mechanism Ag for supporting incremental updates satisfies ε- differential privacy, iff for any output O of Ag and for any two neighbor databases T_1 and T_2, we have*

$$Pr[Ag(T_1) = O] \leq exp(\varepsilon) \cdot Pr[Ag(T_2) = O] \tag{1}$$

where ε is the privacy budget, and the probability is taken over the randomness of Ag. The neighbor database is obtained by removing on arbitrary record from either the original set-valued data, or any of the updates.

A principal technique for achieving differential privacy is *Laplace mechanism* [11]. A fundamental concept of this technique is the *global sensitivity* of a function f that maps underling databases to vectors of reals.

Definition 3. *(**Global Sensitivity**). For any function $f : T \rightarrow \mathbb{R}^d$, the sensitivity of f is defined as follow.*

$$\Delta f = \max_{T_1, T_2} \|f(T_1) - f(T_2)\|_1 \tag{2}$$

for all T_1, T_2 differing in at most one record.

The global sensitivity is also called L_1-sensitivity due to the L_1-norm used in its definition, which takes the maximum over all pairs of neighboring databases.

Laplace Mechanism. Dwork et al. [11] propose the Laplace mechanism which takes a database T, a function f, and the privacy budget ε. It first computes the true output $f(T)$, and then adds properly calibrated Laplace noise to the output. The noise is sampled from a Laplace distribution with the probability density function $Pr(x|b)=\frac{1}{2b}e^{-|x|/b}$, where b is dominated by both Δf and the allocated privacy budget ε.

Theorem 1. *For any function $f: T\rightarrow \mathbb{R}^d$, the private mechanism Ag*

$$Ag(T) = f(T) + \langle Y_1(\Delta f/\varepsilon), Y_2(\Delta f/\varepsilon), ..., Y_d(\Delta f/\varepsilon)\rangle \tag{3}$$

gives ε-differential privacy, where $Y_i(\Delta f/\varepsilon)(1 \leq i \leq d)$ are i.i.d Laplace variables with scale parameter $\Delta f/\varepsilon$.

Differential privacy has two important properties that are extensively used when differential privacy is employed to support combined computations. These two properties are known as *sequential* and *parallel* compositions [18].

3.3 Utility Metrics

In the incremental update case, sanitized set-valued data is mainly used to answer count queries that are crucial to counting queries task. We employ *relative error* [19] to measure the utility of the sanitized data. Given a series of databases $T, \Delta T_1, \Delta T_2, ...$, and let T_i be $T \cup_{j=1}^{i} \Delta T_i$ after appending the i-th incremental update ($i \geq 1$).

Definition 4. ***(Incremental Count Query)**. Given an initial dataset T, after the i-th incremental update ($i \geq 1$), for a given set of items I' drawn from the universe I, an incremental count query Q over T_i is defined to be $Q(T_i)=|\{t \in T_i : I' \in t\}|$.*

relative error(*RE*): This measures the error to the actual answer on the actual database T_i, which is formalized as follows.

$$RE = \frac{|Q(\widetilde{T_i}) - Q(T_i)|}{max\{Q(T_i), b\}} \tag{4}$$

where $Q(\widetilde{T_i})$ denotes the answer on the sanitized database $\widetilde{T_i}$, $Q(T_i)$ denotes the true answer on the actual database T_i, and b denotes a sanity bound used to mitigate the influences of queries with extremely small selectivities.

Problem Statement. Given a private parameter ε, a transactional table T and a series of updates $\Delta T_1, \Delta T_2, ...$ to T, our objective is to generate a series of publications which satisfy differential privacy against incremental updates.

4 Update-Bounded Sanitization Algorithm

In our setting, due to the set-valued dataset incremental update, a private mechanism must update the published statistics as new updates arrive. Thus, traditional differentially private mechanisms either fail to apply directly to our setting, or result in an

unsatisfactory loss in terms of utility or privacy if applied naively (e.g., Solution 1 and Solution 2). To tackle the drawbacks caused by Solution 1 and Solution 2, we propose a novel constraint method, called *Update-Bounded Mechanism*, to limit the number of incremental updates.

Definition 5. *(**Update-Bounded Mechanism**). Given $U \in \mathbb{N}$, an incremental release mechanism Ag is U-bounded if it only accepts updates at most U. In other words, Ag needs to require a priori knowledge of an upper bound on the number of updates.*

Based on update-based mechanism, we present the *IncTDPart* algorithm that recursively partitions the series of set-valued datasets with the help of an IFT-Tree against incremental updates.

We first provide an overview of our *IncTDPart* algorithm in Algorithm 1. The algorithm first builds the IFT-tree H by iteratively grouping f nodes from one level to an upper level until a root is reached. If the size of the item universe is not divided by f, the remainder can be as a group. Given T_i=$\{T, \Delta T_1, \Delta T_2, ...\}$, a privacy budget ε, an upper bound of updates U and an IFT-Tree H, it returns a series of sanitized databases satisfying differential privacy. Based on the given IFT-Tree H, we employ DiffPart method to sanitize the initial dataset T, and release $\widetilde{T}$. *IncBuildTBP-Tree* incrementally maintains a noisy taxonomy-based partitioning tree by a top-down manner, and releases the series of sanitized datasets $\widetilde{T}_i$.

Algorithm 1. IncTDPart

Input: $T, \Delta T_1, \Delta T_2, ..., \Delta T_U$: The initial dataset T, and a series of updates; U: The upper bound on updates; ε: The privacy budget; f: The fan-out of IFT-Tree

Output: The sanitation $\widetilde{T}$, $\widetilde{T}_i$,...

1: SemiEmSet $\leftarrow \emptyset$; NoEmSet $\leftarrow \emptyset$;
2: Construct an IFT-Tree H with fan-out f;
3: $\varepsilon' \leftarrow \frac{\varepsilon}{U+1}$;
Sanitizing the initial dataset T
4: $TBP\text{-}Tree_{(0)} \leftarrow DiffPart(T, H, \varepsilon')$;
5: SemiEmSet$\leftarrow$semi-non empty nodes of $TBP\text{-}Tree_{(0)}$;
6: NoEmSet$\leftarrow$non empty nodes of $TBP\text{-}Tree_{(0)}$;
7: Release $\widetilde{T}$;
Sanitizing the incremental updates
7: **for** each $\Delta T_i (1 \leq i \leq U)$ **do**
8: Root of $TBP\text{-}Tree_{(i-1)} \leftarrow$ all records in ΔT_i;
9: $TBP\text{-}Tree_{(i)} \leftarrow$ **IncBuildTBP-Tree**(ΔT_i, ε', H);
10: Update SemiEmSet, NoEmSet;
11: Release leaf nodes' information of $TBP\text{-}Tree_{(i)}$;
12: Return $\widetilde{T}$, $\widetilde{T}_i$,...;

4.1 The Initial Dataset Sanitization

In principle, we can use any set-valued data sanitization algorithm to sanitize the initial dataset, as long as the sanitized results are differentially private. For example, given the initial dataset T, ε', and H, we use DiffPart method to release $\widetilde{T}$. The method recursively distributes the records in T into disjoint sub-datasets with more specific representations

in a top-down manner. In this method, $\frac{\varepsilon'}{2}$ budget is used to guide the partitioning process of sub-datasets, and the rest $\frac{\varepsilon'}{2}$ plus the budget left from the partitioning process to construct the release in the leaf nodes. In the top-down partitioning manner, all the records in T can be generalized a common generalization, called *hierarchy cut* which consists of a set of nodes in H. A record can be generalized to a hierarchy cut if every item in the record can be generalized to a node in the cut and every node in the cut generalizes some items in the record. For example, the record t_1=$\{I_1, I_2\}$ in Table 1 can be generalized to the cuts $\{I_{\{1,2\}}\}$ and $\{I_{\{1,2,3,4\}}\}$, but not $\{I_{\{1,2\}}, I_{\{3,4\}}\}$, while t_4=$\{I_1, I_3\}$ in Table 2 can be generalized to the cut $\{I_{\{1,2\}}, I_{\{3,4\}}\}$. Fig.2 demonstrates partitioning the three records T=$\{t_1, t_2, t_3\}$ into three leaf nodes of the TBP-tree such that each node contains the noisy count (e.g., 1 is the noisy count of the leftmost leaf node). Every node in the TBP-Tree consists of three fields: *hierarchy cut*, *records*, and *nc*, where *hierarchy cut* denotes the generalization of children of the node, *records* registers which records contain the hierarchy cut or its subsets, and *nc* refers to the accumulated noisy counts from the initial dataset T to the current update ΔT_i.

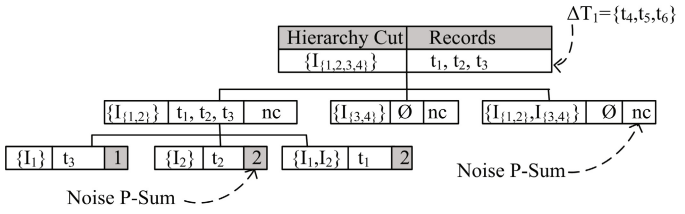


Fig. 2. The TBP-Tree on the initial dataset T

To conveniently partition the nodes in TBP-Tree, we propose an *noisy p-sum* mechanism to record the accumulated noisy counts. That is, each node v (or a *hierarchy cut*) in the TBP-Tree is associated with a *p-sum*(v). After a new batch of incremental updates, the *IncTDPart* method will release noisy versions of these *p-sums* in leaf nodes in TBP-Tree.

Definition 6. ***(noisy p-sum)****. The noisy p-sum of a hierarchy cut in each node of TBP-Tree is the number of records in consecutive updates which contain the hierarchy cut or its subsets. Let* $1 \leq m \leq n$*. We use the notation* $\sum[m,n] = \sum_{j=m}^{n} nc_j(\{I_i\})$ *to denote the noisy p-sum involving the hierarchy cut* $\{I_i\}$ *m trough n, where m and n denote the update timestamps in the sequence of updates.*

For example, in Fig.2, the *p-sum* of the hierarchy cut $\{I_2\}$ is $\sum[0,0]$=$nc_0(\{I_i\})$=2. According to the definition, the $\sum[m,n]$ of some hierarchy cut can be computed by $\sum[1,n]$-$\sum[1,m]$. This is quite consentient for researchers to obtain the noisy counts at every update step or any range of updates.

4.2 TBP-Tree Incremental Construction

Our strategy for *IncBuildTBP-Tree* is to recursively group records in ΔT_i into disjoint subsets based on H, either allocate records into relative nodes of the previous $TBP\text{-}Tree_{(i-1)}$, or create some new nodes that do not exist in the previous tree. Procedure 1

presents the details of *IncBuildTBP-Tree*. To incrementally build *TBP-Trees* based on the coming updates, we employ a uniform privacy budget allocation scheme, that is, divide the total privacy budget ε into equal portions $\varepsilon'=\frac{\varepsilon}{U+1}$, each is used for incrementally maintaining a *TBP-Tree*. For constructing the nodes of the *TBP-Tree* in each update, we use the same budget allocation scheme as DiffPart method that reserves $\frac{\varepsilon'}{2}$ to generate the noisy sizes of leaf nodes, and the rest $\frac{\varepsilon'}{2}$ to guide the top-down partitioning process.

Procedure 1. IncBuildTBP-Tree

Input: ΔT_i: The i-th update; ε': The allocated privacy budget for i-th update; H: The IFT-Tree; $TBP\text{-}Tree_{(i-1)}$: The (i-1)-th TBP-Tree after the (i-1)-th update;
Output: The i-th taxonomy-based partition tree $TBP\text{-}Tree_{(i)}$

1: Vector $V_1 \leftarrow$all sub-partitions of $T, \Delta T_1, \Delta T_2, ..., \Delta T_{i-1}$;
2: Partition $p \leftarrow$all records in ΔT_i;
3: Add p to the root of $TBP\text{-}Tree_{(i-1)}$;
4: $p.cut \leftarrow$ the root of H;
5: $p.\widetilde{\varepsilon'}=\frac{\varepsilon'}{2}$; $p.\alpha=\frac{p.\widetilde{\varepsilon'}}{|InternalNodes(p.cut)|}$;
6: Select a node v from $p.cut$ to partition;
7: Generate all non-empty sub-partition to P;
8: Allocate record in ΔT_i to P;
9: **for** each sub-partition $p_i \in P$ **do**
10: **if** $p_i \in V_1$ **then**
11: **if** $p_i \in$ NoEmSet **and** p_i is not a leaf node **then**
12: **FollowPreviousTraces**($p_i, H, p.\alpha$);
13: **if** $p_i \in$ SemiEmSet **and** $p_i.(\sum_{j=0}^{i-1} nc_j + nc_i) \geq \theta_1$ **then**
14: $p_i.\widetilde{\varepsilon'}=p.\widetilde{\varepsilon'}$-$p.\alpha$;
15: $p_i.\alpha=\frac{p_i.\widetilde{\varepsilon'}}{|InternalNodes(p_i.cut)|}$;
16: Add p_i to V_1;
17: **else**
18: **if** $p_i.nc_i \geq \theta_1$ **then**
19: Repeat Lines 14-16;
20: **for** $j = 1$, $j \leq 2^l$-$|P|$ **do** // l is the number of v's children
21: **if** $p_j.nc \geq \theta_1$ **then**
22: Randomly generate an empty sub-partition p'_j;
23: Repeat Lines 14-16;
24: **if** $p_i.nc_i \geq \theta_2$ **and** p_i is a leaf node **then**
25: Add nc_i copies of $p_i.cut$ to $\widetilde{T_i}$;
26: **else**
27: Add p_i to V_1;
28: Return $TBP\text{-}Tree_{(i)}$;

To make the partition process easier in the updates, we define *Non-Empty node* and *Semi-Non Empty node* in each *TBP-Tree*.

Definition 7. ***(Non-Empty node and Semi-Non Empty node)***. *Let θ_1 be a pre-defined threshold. Give any non-leaf node v of the $TBP\text{-}Tree_{(i)}$, and let $p\text{-}sum(v)=\sum_{j=0}^{i-1} nc_j + nc_i$ be its current accumulated noisy counts. The two concepts are defined as follow.*

$$v = \begin{cases} \textit{Non-Empty node}, \textit{ if } \textit{p-sum}(v) \geq \theta_1 \\ \\ \textit{Semi-Non Empty node}, \textit{ if } 0 < \textit{p-sum}(v) < \theta_1 \end{cases} \tag{5}$$

NoEmSet and SemEmSet are two node sets that consist of Non-Empty nodes, and Semi-Non Empty nodes, respectively.

As the description of Procedure 1, on the i-th update ΔT_i arriving, we first insert the records in ΔT_i into the root of $TBP\text{-}Tree_{(i-1)}$, and recursively partitions them into disjoint subsets. If some records is added to those non-leaf nodes which are non-empty, we call *FollowPreviousTraces* to track the children traces of the non-leaf nodes in $TBP\text{-}Tree_{(i-1)}$, and allocate relative records (Lines 11-12). To some records are allocated to semi-non empty nodes which are in internal levels, we check whether the *p-sums* of the semi-non empty nodes exceed the threshold θ_1. If so, then we either further to partition the nodes or start to construct new releases in leaf level (Lines 14-16). If some records are added to new nodes that do not exist in $TBP\text{-}Tree_{(i-1)}$, Lines 18-19 are called to check whether to further partition or release these new nodes.

During partitioning ΔT_i, many empty nodes will be generated, which are associated with zero number of records. It is critical to prune out the empty nodes in that may lead to poor utility of the release. [20] has indicated that the number of empty nodes k follows the *binomial distribution* $B(m, p_{\theta_1})$, where m is the total number of empty nodes we have to check and $p_{\theta_1}=\frac{exp(-\alpha\theta_1)}{2}$. We can select k uniformly random empty nodes without replacement with noisy counts sampled from the cumulative distribution function $P(x)$=1-$exp(\alpha\theta_1 - \alpha x)$ $(\forall x \geq \theta_1)$. Lines 20-23 show the details of how to generate the empty nodes. Each non-leaf partition $p_i.cut$ in Procedure 1 tracks its unused privacy budget $\widetilde{\varepsilon'}$ and computes the portion of privacy budget α for the next partition operation.

To calculate the budget α, we have to obtain the maximum number of partition operations $InternalNodes(p_i.cut)$ from $p_i.cut$ to leaf nodes. Chen et al. [7] points out that $|InternalNodes(p_i.cut)|=\sum_{u_i \in cut} |InternalNodes(u_i, H)|$, where $|InternalNodes(u_i, H)|$ denotes the number of internal node of the subtree of H rooted at u_i. After the i-th update, in the leaf node level of the $TBP\text{-}Tree_{(i)}$, if a leaf node p_i satisfying $p_i.nc_i \geq \theta_2$ (Lines 24-25), we add nc_i copies of p_i to the $\widetilde{T_i}$. Fig.3 shows the sanitized release after the ΔT_1, ΔT_2 appended the initial dataset T.

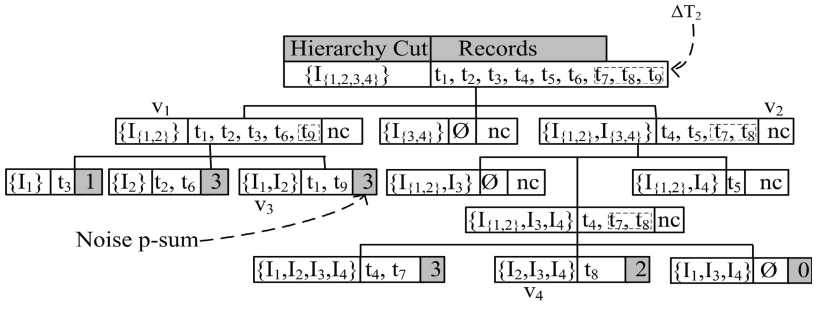


Fig. 3. The TBP-Tree after ΔT_1, ΔT_2 updates

Example 2. Given the update ΔT_2 in Table 5, the privacy budget ε', Procedure 1 works as follows (see Fig.3 for an illustration). It first distributes t_9 into the v_1 node, and t_7, t_8 into the v_2 node. Due to the same partitioning traces of the v_1 node in TBP-Tree$_{(2)}$ as TBP-Tree$_{(1)}$, we directly add t_9 to the v_3 leaf node. The p-sum(v_3) is 3 that equals $nc_0(v_3)$+$nc_2(v_3)$. The v_2 node in TBP-Tree$_{(1)}$ is semi-non empty node in terms of p-sum$(v_2) \leq \theta_1$. The inserted t_7, t_8 records change the node into a non empty node. So, we further split the v_2 node, and obtain three leaf nodes (e.g., the v_4 node).

4.3 Thresholds θ_1, θ_2 Computation

A node in the *TBP-Tree*$_{(i)}$ is further expanded if its *p-sum* value is not less than the threshold θ_1. In the static scenario of *TBP-Tree*, the threshold of non-empty and empty nodes θ_1=$\frac{C1\sqrt{2}}{\alpha}$ (constant times of the standard deviation of noise), and the threshold of leaf nodes publication θ_2=$\frac{C2\sqrt{2}}{\alpha}$, where α is the privacy budget assigned to the nodes. However, for the incremental updates, θ_1 and θ_2 may be changed at each time of update. A straightforward scheme would be using a fixed multiple value of the standard deviation of noise, that is, in different nodes of each *TBP-Tree* , one can use multiple value of standard deviation of noise to guide the partitions in terms of the budget α. Differing from the simple method, we propose an more significant scheme using the *mean* of multiple value of the standard deviation of noise. Let $\alpha_i, ..., \alpha_j$ be the allocated privacy budgets in the series of datasets $T, \Delta T_1, ..., \Delta T_U$ for a non-empty or empty node. The threshold θ_1 (i.e., *mean*) can be defined as follow. According to the same idea as θ_1, we can obtain θ_2. Here, $C1, C2$ are two constants defined by data publishers.

$$\theta_1 = \frac{\sum_{m=i}^{j} \frac{\sqrt{2}C1}{\alpha_m}}{j-i+1} \tag{6}$$

$$\theta_2 = \frac{\sum_{n=i}^{j} \frac{\sqrt{2}C2}{\alpha_n}}{j-i+1} \tag{7}$$

Example 3. In Fig.3, there are four records t_4, t_5, t_7, t_8 in node v_2, two of which came from ΔT_1, the other two from ΔT_2. In TBP-Tree$_{(1)}$, the privacy budget assigned to v_2 is α_1=$\frac{\varepsilon'}{6}$, and α_2=$\frac{\varepsilon'}{6}$ in TBP-Tree$_{(2)}$. According to the above equations, we get θ_1=$\frac{3\sqrt{2}C1}{\varepsilon'}$. Due to p-sum$(v_2) \geq \theta_1$, we further partition the node v_2.

4.4 Analysis

Privacy Analysis. We give the differential privacy guarantee of our method below.

Theorem 2. *Each of the updates of IncTDPart algorithm is $\frac{\varepsilon}{U+1}$-differentially private, where U is the upper bound on the number of the updates.*

Proof. We prove the theorem by the definition of ε-differential privacy. Consider two neighboring datasets T_1 and T_2. We first consider Lines 7-11 of Algorithm 1, that is, the incremental construction of TBP-Tree. Let this part be denoted by Ag. Given any update

timestamp $i(1 \leq i \leq U)$, and ε' ($\varepsilon'=\frac{\varepsilon}{U+1}$), and let the i-th TBP-Tree be denoted by $Tree_i$. In essence, $Tree_i$ is constructed on the noisy answers to a set of incremental counting queries. Let each root-to-leaf path be indexed by k. We denote a node in level l and path k by v_{kl}, its privacy budget by ε'_{kl}, and its incremental count in T_1 and T_2 by $Q(T_1)_{kl}$ and $Q(T_2)_{kl}$, respectively. We first claim that a single record can only affect at most 2^c ($c \leq f$) root-to-leaf paths, where 2^c is the children number of the node v_{kl}. Then we have

$$\frac{Pr(Ag(T_1) = Tree_i)}{Pr(Ag(T_2) = Tree_i)} = \prod_{l=1}^{h}\prod_{k=1}^{2^c} \frac{exp(-\varepsilon'_{kl}\frac{|TC(v_{kl})-Q(T_1)_{kl}|}{2^c})}{exp(-\varepsilon'_{kl}\frac{|TC(v_{kl})-Q(T_2)_{kl}|}{2^c})}$$
$$\leq exp\left(\frac{\sum_{l=1}^{h}\sum_{k=1}^{2^c}\varepsilon'_{kl}|Q(T_1)_{kl}-Q(T_2)_{kl}|}{2^c}\right)$$
$$\leq exp\left(\frac{1}{2^c}\sum_{l=1}^{h}\sum_{k=1}^{2^c}\varepsilon'_{kl}\right)$$

where $TC(v_{kl})$ is the true count of the node v_{kl}, h is the height of the $Tree_i$. Since $\sum_l \varepsilon'_{kl}=\varepsilon'$, we have $\frac{Pr(Ag(T_1)=Tree_i)}{Pr(Ag(T_2)=Tree_i)} \leq exp(\varepsilon')$.

The use of privacy budget on different updates follows *sequential composition* [18].

Theorem 3. ***(Sequential Composition)****. If a randomized algorithm Ag runs a sequence of $Ag_1(T)$, $Ag_2(T)$,..., $Ag_U(T)$ over the dataset T, where each Ag_i provides ε_i differential privacy, then Ag(T) is $\sum_{i=1}^{U}\varepsilon_i$-differentially private.*

We now show that Algorithm 1 satisfies ε-differential privacy.

Theorem 4. *IncTDPart algorithm is ε-differentially private.*

Proof. Given the upper bound U on updates. According to the Theorem 2, we obtain that each of the updates of IncTDPart algorithm satisfies $\frac{\varepsilon}{U+1}$-differential privacy. Besides the initial dataset, we run IncTDPart algorithm $U+1$ times. According to Theorem 3, we get $\sum_{i=1}^{U+1}\frac{\varepsilon}{U+1}=\varepsilon$. Therefore, IncTDPart algorithm is ε-differentially private.

5 Experiments

In this section, we evaluate the utility of our proposed algorithm in terms of utility for incremental counting queries, and examine the scalability of our method for processing large-scale datasets. Our implementation was done in C++, and all experiments were performed on an Intel Core 2 Duo 2.94GHz Pc with 4GB RAM. Extensive experiments were performed on two real datasets, *MSNBC* [21], and *kosarak* [22], which record the URL categories visited by users in time order, and clickstream data, respectively.

Table 6. Shows more details of the two datasets

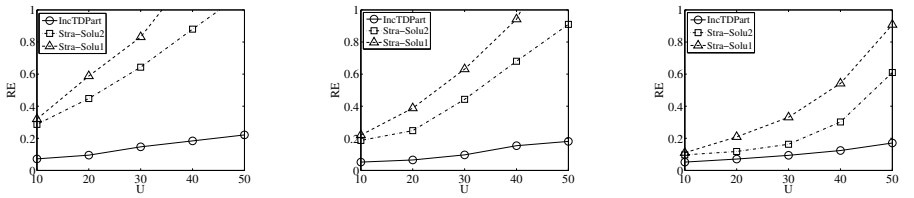
Dataset	***N***	$\|I\|$	$Avg\|t\|$
MSNBC	989,818	17	1.72
Kosarak	990,002	41,270	8.1

The characteristics of the datasets are summarized in Table 6, where N is the number of records in each datasets, $|I|$ the number of distinct items and $Avg|t|$ the average record length. We compare the utility and scalability of our method IncTDPart described in Algorithm 1 to the two straightforward solutions discussed in Section 1. We use Stra-Solu1 and Stra-Solu2 to denote the two basic methods, respectively.

5.1 Utility

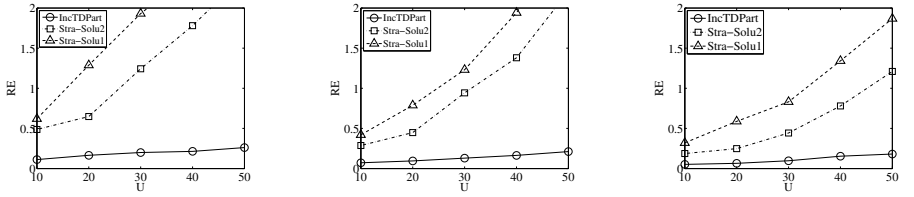
Based on the above two datasets, we perform our experiment to demonstrate the utility of our method under different number of updates. In the experiment, we randomly chose 400,000 records as the initial dataset T from *MSNBC* and *Kosarak*, respectively, and chose another 10,000 records without replacement as the incremental update ΔT_i.

Effect of U on Utility. In the first set of experiments, we examine the relative error (RE) of incrementally counting queries on the sanitized datasets under varying the upper bound U from 10 to 50, and fixing f=10. For each dataset, we randomly generate a counting query whose items is drawn from I. Fig. 4 and Fig.5 show the relative error of IncTDPart, Stra-Solu1, and Stra-Solu2 with respect to different privacy budget ε. The relative error decreases when ε varies from 0.5 to 1.0 because less Laplace noise is injected. From the two figures, we can see that the error of the two straightforward solutions are larger than that of IncTDPart algorithm in all cases. When the upper bound U increases, the performances of Stra-Solu1 and Stra-Solu2, especially Stra-Solu1, deteriorate sharply because they do not employ the update-bounded mechanism to incrementally release the sanitized datasets.



(a) ε=0.5 (b) ε=0.75 (c) ε=1.0

Fig. 4. Relative error under MSNBC dataset



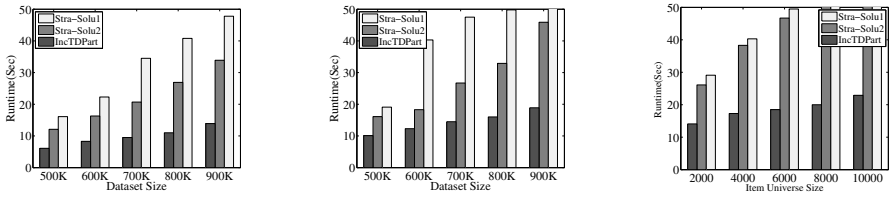
(a) ε=0.5 (b) ε=0.75 (c) ε=1.0

Fig. 5. Relative error under Kosarak dataset

5.2 Scalability

In the last set of experiments, we examine the scalability of our method against Stra-Solu1 and Stra-Solu2. The runtime is used as our performance metric, which is dominated by the datasets size and item universe size. We first evaluate the efficiency of the three algorithms by varying the dataset size from 500K to 900K and setting ε=1.0, U=50, and f=10. Fig.6(a) and Fig.6(b) show the runtime of the three methods under the

two datasets *MSNBC* and *Kosarak*. As dataset size grows, in particular, the size exceeds 600K, the runtime of Stra-Solu1 and Stra-Solu2 increase more dramatically. This is because when increasing dataset size, the two methods have to take more time to partition numerous candidate nodes under lacking the help of taxonomy-based partitioning tree. As expected, the runtime of our method is linear of the dataset size. Fig.6(c) presents how the runtime varies under different item universe size on the dataset *Kosarak*, where ε=1.0, U=50, and f=10. From the Fig.6(c), it can be seen that the runtime of our algorithm scales linearly with the increase of universe size, however, the runtime of Stra-Solu1 and Stra-Solu2 grows quickly.



(a) Runtime on MSNBC (b) Runtime on Kosarak (c) Runtime on Kosarak

Fig. 6. Scalability under MSNBC and Kosarak datasets

6 Conclusions

In this paper, we have studied the problem of releasing set-valued data against incremental updates in the framework of differential privacy. Based on the update-bounded mechanism, we first proposed an efficient algorithm IncTDPart for answering the incrementally counting queries with the help of taxonomy-based partitioning tree. We dynamically maintained the tree to partition the incremental records. Then we proved our algorithm that satisfied ε-differential privacy. Experiments on real datasets show that our algorithm outperforms the two straightforward solutions. As the future work, we will investigate how to preserve ε-differential privacy against incremental updates for other application scenarios (e.g., releasing sequential data, and temporal data).

Acknowledgement. We sincerely thank Yin Yang (ADSC) for his comments.

References

1. Terrovitis, M., Mamoulis, N., Kalnis, P.: Privacy-preserving anonymization of set-valued data. In: Proceedings of the VLDB Endowment (PVLDB 2008), vol. 1(1), pp. 115–125 (2008)
2. He, Y., Naughton, J.F.: Anonymization of Set-Valued Data via Top-Down, Local Generalization. In: Proceedings of the VLDB Endowment (PVLDB 2009), vol. 2(1), pp. 934–945 (2009)
3. Ganta, S.R., Kasiviswanathan, S.P., Smith, A.: Composition attacks and auxiliary information in data privacy. In: Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008), pp. 265–273 (2008)

4. Wong, R.C.W., Fu, A., Wang, K., Yu, P.S., Pei, J.: Can the utility of anonymized data be used for privacy breaches. ACM Transaction on Knowledge Discovery from Data (TKDD 2011) 5(3), 16 (2011)
5. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part II. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
6. Kifer, D., Machanavajjhala, A.: No free lunch in data privacy. In: Proc. of the 31th International Conference on Management of Data (SIGMOD 2011), pp. 193–204 (2011)
7. Chen, R., Mohammed, N., Fung, B.C.M., Desai, B.C., Xiong, L.: Publishing Set-Valued Data via Differential Privacy. In: PVLDB 2011, vol. 4(11), pp. 1087–1098 (2011)
8. Chen, R., Fung, B.C.M., Desai, B.C., Sossou, N.M.: Differentially private transit data publication: a case study on the montreal transportation system. In: Proc. of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2012), pp. 213–221 (2012)
9. Dwork, C., Naor, M., Pitassi, T., Rothblum, G.N.: Differential pirvacy under continual observation. In: Proc. of the 42nd ACM Symposium on Theory of Computing (STOC 2010), pp. 715–724 (2010)
10. Chan, T.-H.H., Shi, E., Song, D.: Private and continual release of statistics. ACM Transactions on Information and System Security (TISS 2011) 14(3), 26 (2011)
11. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
12. Xu, J., Zhang, Z., Xiao, X., Yang, Y., Yu, G.: Differentially private histogram publication. In: Proc. of the 28th International Conference on Data Engineering (ICDE 2012), pp. 32–43 (2012)
13. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentially private histogram through consistency. In: Proceedings of the VLDB Endowment (PVLDB 2010), vol. 3(1), pp. 1021–1032 (2010)
14. Götz, M., Machanavajjhala, A., Wang, G., Xiao, X., Gehrke, J.: Publishing search logs - a comparative study of privacy guarantees. IEEE Transaction Knowledge and Data Engineering (TKDE 2012) 24(3), 520–532 (2012)
15. McSherry, F., Mironov, I.: Differentially private recommender systems: building privacy into the netflix prize contenders. In: Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009), pp. 627–636 (2009)
16. Byun, J.-W., Sohn, Y., Bertino, E., Li, N.: Secure anonymization for incremental datasets. In: Jonker, W., Petković, M. (eds.) SDM 2006. LNCS, vol. 4165, pp. 48–63. Springer, Heidelberg (2006)
17. Xiao, X., Tao, Y.: m-invariance: Towards privacy preserving republication of dynamic datasets. In: Proc. of the 27th International Conference on Management of Data (SIGMOD 2007), pp. 689–700 (2007)
18. McSherry, F.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: Proc. of the 29th International Conference on Management of Data (SIGMOD 2009), pp. 19–30 (2009)
19. Xiao, X., Bender, G., Hay, M., Gehrke, J.: iReduct: differential privacy with reduced relative errors. In: Proc. of the 31th International Conference on Management of Data (SIGMOD 2011), pp. 229–240 (2011)
20. Gormode, G., Procopiuc, M., Srivastava, D.: Differentially private summaries for sparse data. In: Proc. of the 15th International Conference on Database Theory (ICDT 2012), pp. 299–311 (2012)
21. UCI machine learning repository, `http://archive.ics.uci.edu/ml`
22. Frequent itemset mining dataset repository, `http://fimi.ua.ac.be/data/`

Consistent Query Answering Based on Repairing Inconsistent Attributes with Nulls

Jie Liu, Dan Ye, Jun Wei, Fei Huang, and Hua Zhong

Institute of Software, Chinese Academy of Sciences
Beijing, China 100190
{ljie,yedan,wj,huangfei06,zhongh}@otcaix.iscas.ac.cn

Abstract. Although integrity constraints can successfully capture data semantics, the actual data in the database often violates such constraints. A Consistent Query Answer (CQA) in a possibly inconsistent database is an answer which is true in every minimal repair of the database. It has been proved that for most constraints and queries CQA is a NP problem based on repairing by tuple deletions or tuple insertions. Furthermore, repairing by deleting tuples will also cause information losing. In this paper we present a new repair semantics named repairing with nulls, which replaces the inconsistent attribute values with nulls. To capture all the inconsistent attribute values, we study the transitivity of nulls and provide an algorithm to extend the original constraints. Based on repairing with nulls, there will be only one repair and CQA can be computed in PTIME by SQL query rewriting. Finally, we study the performance of our new approach for CQA by detailed experiments.

Keywords: Consistent Query Answering, integrity constraint, data semantics, query rewritting.

1 Introduction

SQL queries are the basis of data analysis. When there are inconsistencies in the database, the result will also be untrusted. There are two main approaches to deal with the inconsistencies so far: Consistent Query Answering (CQA) [1,10,8] which computes consistent answers without changing the database and data cleaning [15] which repairs the original databases by some strategy. Data cleaning techniques are used to identify the inconsistency and rectify errors to restore the database to be consistent. Data cleaning has some limitations. For example, maybe we have no writing privilege to the date source, maybe we have not enough knowledge to resolve conflicts and maybe we would not like to change the data source anyway.

CQA still have little real applications so far [6], because existing repair semantic have no clear application semantics and most algorithms are of high computing complexity. In this paper we will focus on applying CQA in statistical analysis applications. When data are collected for analysis, each individual is corresponding to one real object but may be with incorrect attributes.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 407–423, 2013.

Some attributes are inconsistent when we define some constraints. We present a new repair semantic, that is to repair inconsistent attributes with NULL. Then the dataset will restore to be consistent.

Example 1. *Consider a database instance* I_1 *in Figure 1 with two relations* order (oid, orderpri, shippri) *and* lineitem (lid, oid, quan,disc,iname) *which are integrated from autonomy databases of several sale departments and contain the information about the custom orders. In* order, oid *is the key attribute,* orderpri *is the priority of the order, and* shippri *whcih records the priority of the shipment of each order entry. The priority is a positive value between 1 to 5 inclusive and the lower value means the higher priority. In* lineitem, lid *is the key attribute,* quan *is the quantity (quan) of the item in the order, (*disc*) is discount and (*iname*) is name of the item. The value of* disc *ranges from 0 to 0.10 inclusive. User specifies an integrity constraint: iff the quantity of the items in* lineitem *less than 20 should not have order priority higher than 3 level, that is* ic_1*:* $\forall xyzwuv$ $\neg(order(x, y) \wedge lineitem(u, y, v) \wedge y < 3 \wedge v < 20)$*. The entry in lineitem whose quantity is less than 15 should not has discount more than 0.06, that is* ic_2*:* $\neg$ $(lineitem(u, y, v, w) \wedge v < 15 \wedge disc > 0.06)$*. The order with* shipprio *less than 3 should not be greater than 3, that is* ic_3*:* $\neg$ $(order(x, y, z) \wedge z < 3 \wedge y > 3)$*.*

	oid	orderpri	shippri
o_1	o105	(2)	[2]
o_2	o210	5	5
o_3	o110	(2)	3
o_4	o101	3	4
o_5	o211	4	4

	lid	oid	quan	disc	iname
l_1	L10	o105	(18)	[0.07]	book1
l_2	L11	o210	12	0.03	cd1
l_3	L12	o110	(16)	0.04	book1
l_4	L13	o101	22	0.05	book1
l_5	L14	o211	(14)	(0.08)	cd2

(t) the value t violating ics directly
[t] the value t impacted by transitivity of nulls

Fig. 1. The order and lineitem relations

By checking each integrity constraint independently, we find that tuple o_1 and l_1 violate ic_1, tuple o_3 and l_3 violates ic_1 and tuple l_5 violates ic_2. According to semantics of *repairing with nulls*, the *orderpri*'s value of o_1 and o_3, the *quan*'s of l_1, l_3, l_5 and the *disc*'s value of l_5 are repaired as *null*. All of these values are surrounded by parentheses. Take o_1 into account, the value of *shippri* less than 3, $shippri(o_1) < 3 \models true$ which means that its *orderpri*'s value should not be greater than 3 in order to satisfy ic_3. Since the *orderprio*'s value of o_1 is repaired with *null* according to ic_1, the *shippri*'s value should be repaired with *null* to satisfy ic_3. With the same reason, due to $disc(l_1) > 0.06 \models true$, the *disc*'s value of l_1 should be repaired as *null* according to ic_2. These values are surrounded by square brackets. Consider o_3, the *shipprio*'s value is 3, which make ic_3 satisfied. Hence we consider the *shipprio* of o_3 is consistent with the value unchanged.

If two integrity constraints (*ic*s) have common attribute variables, the inconsistent attributes w.r.t one *ic* will influence the attributes in the other constraint. We define this influence as *transitivity of nulls*. Hence, the integrity constraints

should be extended to filter out all inconsistent attribute values. In section 3, we will provide a algorithm to compute the extended *ic*s.

The first step in repairing with nulls is to locate all the inconsistent attribute values. We can rewrite the constraints to SQL clause to fetch the inconsistent attribute values. However, if several constraints have common attributes, they will influence each other. We define this issue as the transitivity of nulls and present an algorithm to extend the original constraints. Then we can execute the constraints in any order to get the same answers.

The target of this research is to compute consistent query answer from an inconsistent database instance with the database unchanged. We provide the SQL rewriting techniques to return CQA for the original queries rather than repair the database really. Under this repair semantics we find there is only one repair, hence CQA can be computed in PTIME.

The plan of the paper is as follows. In Section 2, we introduce basic concepts and provide a precise definition of the semantics of repairing with nulls. In Section 3, we illustrate the transitivity of nulls in details and provide an algorithm to extend the constraints to deal with the transitivity. In Section 4 and Section 5, we give the SQL rewriting algorithms for non-aggregation queries and aggregation queries respectively. In Section 6, we show our experiments design and analyze the overhead of the rewriting. In Section 7, we briefly discuss related work. Conclusion and a discussion of possible future research directions will be provided in Section 8.

2 Preliminaries

In this paper, we only consider the denial constraints for the reason that most common integrity constraints are of this form.

Definition 1. (Denial constraints [11]) Denial constraints are in the form of:

$$\forall \bar{x}_1, \ldots, \bar{x}_k, \neg(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m) \wedge \phi(\bar{x}_1, \ldots, \bar{x}_k))$$

Where P_i is a relation predicate, $\bar{x}_i$ is a sequence of attribute variables or constants and $\phi(\bar{x}_1, \ldots, \bar{x}_k)$ is a sequence of atomic formulas referring to built-in predicates, such as $x > 1$, $x = y$ etc. $\mathcal{A}_b$ denotes the set of attribute variables involved in ϕ, and $\mathcal{A}_r$ denotes the other attribute variables in the constraints. Note that, the functional dependencies and exclusion constraints are of the above form.

Definition 2. (Inconsistent Attribute) A database instance I is *consistent*, if I satisfies a set of integrity constraints IC, that is $I \models IC$; *inconsistent* otherwise. A tuple $\bar{t}$ is a *consistent* if $I(\bar{t}) \models IC$, inconsistent otherwise. If $\bar{t}$ is inconsistent, $\mathcal{S}_{ic}(\bar{t})$ is a set of integrity constraints which the inconsistent tuple $\bar{t}$ violated, $\mathcal{A}_v(\bar{t})$ is the set containing attribute variables in $\mathcal{A}_b$ of all *ic*s in $\mathcal{S}_{ic}(\bar{t})$. Tuple $\bar{t}$'s attribute v is *inconsistent attribute* if $v \in \mathcal{A}_v(\bar{t})$; *consistent* otherwise.

Example 2. *Consider example 1, I_1 is inconsistent, o_1 and l_1 are inconsistent tuples. $\mathcal{S}_{ic}(o_1)$ and $\mathcal{S}_{ic}(l_1)$ is $\{ic_1\}$, $\mathcal{A}_v(o_1)$ is* {orderpri} *and $\mathcal{A}_v(l_1)$ is* {quan}.

Definition 3. (Repairing with Nulls) Consider a database instance I with a set of integrity constraints IC. Given a tuple $\bar{t}$, $I(\bar{t}) \not\models IC$, $\forall\ v_i \in \mathcal{A}_v(\bar{t})$, replace the v_i's value of $\bar{t}$ with *null*.

If integrity constraints in IC are unrelated, that is they do not have common attribute variables, we can filter out the inconsistent attribute values easily by adding the constraints into the original quires.

Definition 4. (Transitivity of nulls) Consider two integrity constraints ic_i and ic_j which have common attribute variables $\bar{v}$. $\bar{v}_i$ and $\bar{v}_j$ denote attribute variables only in each constraint respectively. There is a database instance I. $\bar{t}$ is an inconsistent tuple violating ic_1, where $\bar{v}_i$, $\bar{v} \in \mathcal{A}_v(\bar{t})$. $\phi_j(\bar{v}_j)$ is a conjunctive sequence of atomic formulas referring to built-in predicates in ic_j. If $\phi_j(\bar{v}_j) \models$ *true*, the $\bar{v}_j$'s values of $\bar{t}$ are inconsistent and should be repair with *nulls*.

Definition 5. (Database repair) Given an integrity constraints set IC and an inconsistent database instance I. I_{null} is a repair of I w.r.t IC by *nulls*, if $I_{null} \models IC$.

Definition 6. (Consistent Query Answering) Given a database instance I with a integrity constraints set IC. A ground tuple $\bar{t}$ is a consistent answer to a query q over the database I, if $I_{null} \models q(\bar{t})$. The definition is shown as follows:

$$CQA^q_{IC}(I) = QA^q(I')$$

3 Locating the Inconsistent Attributes

To filter out all the inconsistent attributes, the original *ic*s should be extended. If the attributes of tuples satisfy the extended *ic*s, they are consistent. In this section we provide an algorithm to compute the extension of *ic*s. We illustrate it with following examples.

Example 3. (ICs extension) *Consider a database instance I with three integrity constraints:*

ic_1: $\neg(P_1(\bar{A}_1) \wedge \phi_x(x) \wedge \phi_v(v))$,
ic_2: $\neg(P_2(\bar{A}_2) \wedge \varphi_x(x) \wedge \varphi_y(y))$,
ic_3: $\neg(P_3(\bar{A}_3) \wedge \psi_y(y))$

q is a query involving an attribute variable v. $\Psi(v)$ is the extended ic about v for query q, whose calculation process is shown as follows:

$$\begin{aligned}
\Psi(v) &= \neg\{(P_1(\bar{A}_1) \wedge [(\phi_x(x) \wedge \phi_v(v)) \\
&\qquad \vee (P_1(\bar{A}_1) \wedge \phi_v(v) \wedge x = null)]\} && (3.1)\\
&= \neg\{P_1(\bar{A}_1) \wedge [(\phi_x(x) \wedge \phi_v(v)) \\
&\qquad \vee (\phi_v(v) \wedge (P_2(\bar{A}_2) \wedge ((\varphi_x(x) \wedge \varphi_y(y)) \\
&\qquad\qquad \vee (\varphi_x(x) \wedge y = null))]\} && (3.2)\\
&= \neg[(P_1(\bar{A}_1) \wedge \phi_x(x) \wedge \phi_v(v)) \\
&\qquad \vee (P_2(\bar{A}_2) \wedge \phi_v(v) \wedge \varphi_x(x) \wedge \varphi_y(y)) \\
&\qquad\qquad \vee (P_3(\bar{A}_3) \wedge \phi_v(v) \wedge \varphi_x(x) \wedge \psi_y(y))] && (3.3)
\end{aligned}$$

ic_1 is the directly relevant constraints involving v. ic_1 and ic_2 has common attribute variable x, the inconsistent values of x w.r.t ic_2 will impact the variable v in ic_1 which is shown is expression (3.1). Similarly, the inconsistent values of y will impact the variable x in ic_2 and then impact v in ic_1, which is shown in expression (3.2).

To generalize the previous example to an algorithm for all situations. Let us consider how to handle the situation that the relationship of the integrity constraints contains a cycle. We illustrate this in the next example.

Example 4. (CYCLIC ICS EXTENSION) *We modify the Ex 4 with another three ics:*

ic_1: $\neg(P_1(\bar{A}_1) \wedge \phi_x(x) \wedge \phi_z(z) \wedge \phi_u(u))$,
ic_2: $\neg(P_2(\bar{A}_2) \wedge \varphi_x(x) \wedge \varphi_y(y) \wedge \varphi_v(v))$,
ic_3: $\neg(P_3(\bar{A}_3) \wedge \psi_z(z) \wedge \psi_y(y) \wedge \psi_w(w))$.

Each *ic* has a common attribute variable with other two *ic*s. Consider a query q referring variable u. The extended *ic* $\Psi(u)$ is computed as follows:

$$\begin{aligned}
\Psi(u) = \neg[&(P(\bar{A}_1) \wedge \phi_x(x) \wedge \phi_y(y) \wedge \phi_u(u)) && (3.4)\\
&\vee(P_1(\bar{A}_1) \wedge \phi_z(z) \wedge \phi_u(u) \wedge x = null && (3.5)\\
&\vee(P_1(\bar{A}_1) \wedge \phi_x(x) \wedge \phi_u(u) \wedge z = null)]\} && (3.6)\\
(3.2) = P_1(\bar{A}_1) &\wedge \phi_z(z) \wedge \phi_u(u)\\
&\wedge[(\varphi_x(x) \wedge \varphi_y(y) \wedge \varphi_z(z))\\
&\quad \vee(\varphi_x(x) \wedge \varphi_v(v) \wedge y = null)]\\
= (P_1(\bar{A}_1) &\wedge P_2(\bar{A}_2) \wedge \phi_z(z) \wedge \phi_u(u)\\
&\wedge \phi_2(x) \wedge \varphi_y(y) \wedge \varphi_z(z))\\
&\vee(P_1(\bar{A}_1) \wedge P_2(\bar{A}_2) \wedge P_3(\bar{A}_3)\\
&\quad \wedge \phi_z(z) \wedge \phi_u(u) \wedge \varphi_x(x) \wedge \varphi_v(v)\\
&\quad\quad \wedge \psi_z(z) \wedge \psi_y(y) \wedge \psi_w(w) && (3.7)\\
(3.3) = P_1(\bar{A}_1) &\wedge \phi_x(x) \wedge \phi_u(u)\\
&\wedge[(\psi_z(z) \wedge \psi_y(y) \wedge \psi_w(w))\\
&\quad \vee(\psi_z(z) \wedge \psi_w(w) \wedge\ y = null)]\\
= (P_1(\bar{A}_1) &\wedge P_3(\bar{A}_3) \wedge \phi_x(x) \wedge \phi_u(u)\\
&\wedge \psi_z(z) \wedge \psi_y(y) \wedge \psi_w(w))\\
&\vee(P_1(\bar{A}_1) \wedge P_2(\bar{A}_2) \wedge P_3(\bar{A}_3)\\
&\quad \wedge \phi_x(x) \wedge \phi_u(u) \wedge \psi_z(z) \wedge \psi_w(w)\\
&\quad\quad \wedge \varphi_x(x) \wedge \varphi_y(y) \wedge \varphi_v(v) && (3.8)
\end{aligned}$$

After several extensions of atomic formula with nulls*, $\Psi(u)$ reaches a fixed point, $\Psi(u) = \neg((3.4) \wedge (3.7) \wedge (3.8))$.*

Proposition 1. Given a database instance I with a set of integrity constraints IC. q is a query referring a sequence of attribute variables $\bar{v}$ over the database I. $\Psi(\bar{v})$ is the extended constraint over $\bar{v}$ according to transitivity of *nulls*. A ground tuple $\bar{t}$ is consistent to q, if and only if $I(\bar{t}) \models \Psi(\bar{v})$.

In the following section, we provide a algorithm base the *IC Relational Graph* to compute the extended integrity constraints. Let us introduce the definition of IC Relational Graph.

Definition 7. (IC RELATIONAL GRAPH) Consider a set of integrity constraints IC, $G(V_{ic}, E)$ is an Integrity Constraints Relational Graph (ICRG). g_i denotes node of ic_i and e_{ij} means that there are some common attributes involved both in ic_i and ic_j, where $g_i \in V_{ic}$ and $e_{ij} \in E$.

Algorithm 1. *CalCons: Calculate the consistent constraint for the original query.*
$recvQueue_i$: the queue of g_i storing the constraint expression from neighbor nodes;
$curExp_i$: current extended constraint expression for g_i;
$\mathcal{A}_q$: the set of attributes involved in the query q;

foreach $ic_i \in IC$
  $recvQueue_i := \emptyset$
  $expChanged_i := false$
foreach $ic_i \in IC$
  foreach $p \in \mathcal{A}_q$
    if $p \in A_i$
      $currExp_i$:= *calConjuctiveExp(*ic_i*,null,null)*
      foreach g_j *adjacent to* g_i
        ENQUEUE($recvQueue_j$, $currExp_i$*)*
while(there is a $recvQueue_i <> \emptyset$*)*
  $tmpExp$:= *DEQUEUE(*$recvQueue_i$*)*
  foreach conjunctive clause c in tmpExp
    if c is not implied by $currExp_i$
      $currExp_i := currExp_i \vee c$
      $expChanged_i := true$
    if c and ic_i *has common attribute variables*
      $newClause$:= *calConjunctiveExp(*ic_i, ic_j, c*)*
      if newClause is not implied by $currExp_i$
        $currExp_i := currExp_i \vee newClause$
        $expChanged_i := true$
  if $expChanged_i = true$
    foreach g_j *adjacent to* g_i
      ENQUEUE($recvQueue_j$, $currExp_i$*)*
return $currExp$

Algorithm 2. *CalConjuctiveExp(g_i, g_j, c): Calculate the extended conjunctive expression.*
g_i, g_j: the nodes of integrity constraint relational graph;
c: a conjunctive expression from g_i to g_j;

  $newClause := null$
foreach attribute $p \in c$ && $p \notin ic_i$
  $newClause := newClause \wedge \phi_i(p)$
  $expChanged_i := true$
foreach built-in predicate $\phi(q) \in ic_j$
  $newCluase := newClause \wedge \phi_j(q)$
return $newClause$

Given a set of integrity constraints, we can form a IC Relational Graph (ICRG), then compute the extended constraints for attribute variables. We present the algorithm *CalCon* to compute extended constraints. First, the set of original *ic*s are represented as a ICRG. Each node g is associated with a *recvQueue* which stores the constraints from the neighbor nodes and the *currExp* maintains the constraint ever calculated. Given a query q, $\mathcal{A}_q$ is the set of attribute variables involved in q. If the ic_i of node g_i directly referring the attributes in q, g_i is marked as a start of the calculation process. The $currExp_i$ is initialized with ic_i, and is pushed to the $recvQueue_j$ of the g_i's neighbor nodes. When a node g_j receives a constraint expression *recvExp* from its neighborhood,

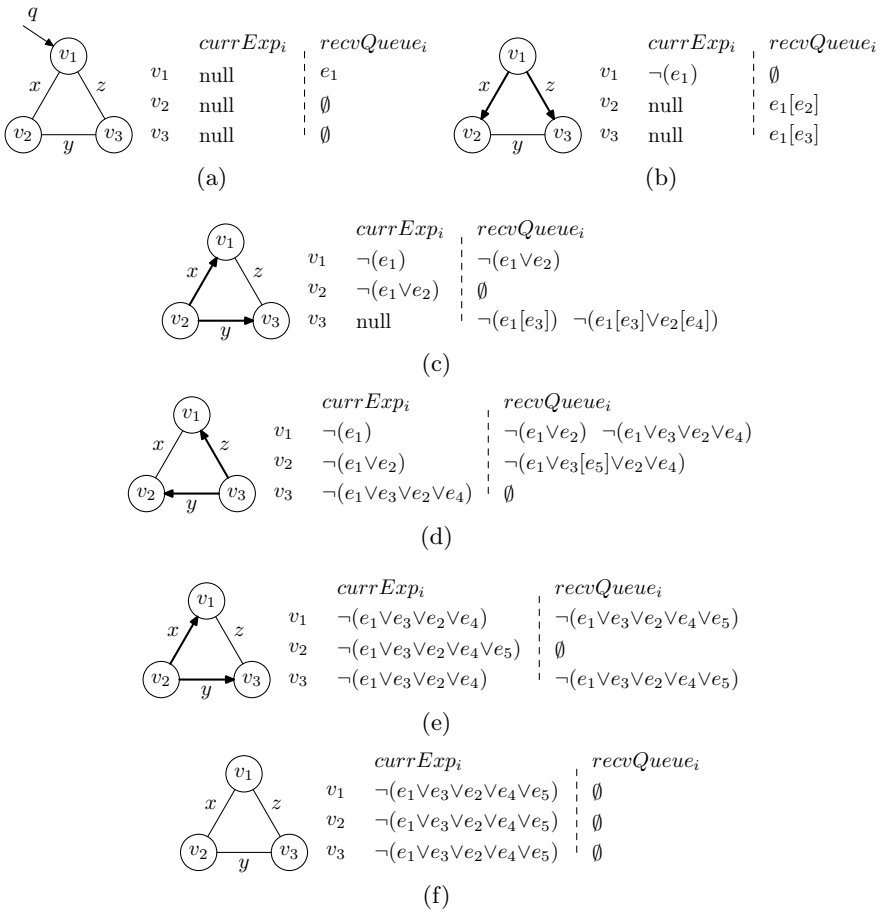


Fig. 2. The execution of CalCons for Example 5

first g_j merges each conjunctive clause of $recvExp_j$ into $currExp_j$ if the clause is not implied by $currExp_j$. Then node g_j extends the clauses which involve the common variables of ancestor node with taking its constraint ic_j into consideration together. If the new conjunctive clause is not implied in the $currExp_j$, it is merged into $currExp_j$ and node g_j refreshes neighbor's *recvQue*s. The algorithm *CalConjunctiveExp* is used to generate the new conjunctive clause in this process. Finally, $\Psi(\bar{v})$ reaches a fixed point with each *currExp* has the same value. In this process, the attribute variables can be pre-sorted, and conjunctive clauses can be indexed to speed up the verification of implication of two constraint expressions. Let us illustrate this algorithm by following example.

Example 5. *The calculation process of $\Psi(u)$ for example 4 is provided in Figure 5. As is shown, each node represents a denial integrity constraint, each edge is marked with the common attribute variables of each two constraints. Node g_1 is associated with query directly, and its $recvQueue_i$ is initialized with its own constraint e_1. Then, the node g_1 calculates the current constraint $\neg(e_1)$ and sends the result to its neighborhoods g_2 and g_3. This is shown in Figure 5(b). Each expression e_i is a conjunctive clause. The expression in the recvQueue is followed by the calculation result surrounded by square brackets, if the original expression is changed in the calculation process. When g_2 and g_3 receive the expressions from g_1, they compute the extended constraint expression and deliver it to their neighborhoods if the expression changed. Finally, as Figure 5(f) shows, the currExp of each node has the same value and recvQueue is empty.*

4 Non-aggregation Query Rewriting

In this section, we illustrate the rewriting strategy for queries which not involve any aggregation expressions. In section 5, we will consider the queries that include aggregation. We illustrate the rewriting approach with the following example.

Example 6. (Ex1 continued) *Consider a query Q_3 over the database instance of example 1. We want to retrieve the order id whose priority is greater than 2.*

Q_3 : *select oid from order o*
where g.orderpri > 2

Q_3 *involves attribute* orderpri. *According to algorithm* CalCons, $\Psi($orderpri$)$ *is:* $\neg$*[order (oid, orderpri, shippri)* $\wedge$ *lineitem (lid, oid, quan, disc)* $\wedge$ *((orderpri* $<$ *3* $\wedge$ *quan* $<$ *20)* $\vee$ *(orderpri* $>$ *3* $\wedge$ *shipprio* $<$ *3)* $\vee$ *(orderpri* $<$ *3* $\wedge$ *quan* $<$ *15* $\wedge$ *disc* $>$ *0.06))]. We merge the extended constraint $\Psi($*orderpri*$)$ into the original query. The consistent query rewriting of Q_3 is described as follows:*

```
with Candidates as(
    select * from order o
        where o.orderpri > 2),
Candidates_1 as (
    select * from Candidates cand join lineitem l on cand.oid = l.oid
    where (cand.orderpri >= 3 or l.quan >= 20) and ( cand.orderpri <= 3 or cand.shippri
>= 3)),
```

Candidates_2 as (
*select * from Candidates_1 cand_1*
where cand_1.orderpri >= 3 or cand_1.quan >= 15 or cand_1.disc <= 0.06))
select oid from Candidates_2

The subquery Candidates *extracts the tuples satisfying the original query conditions. The tuples that violate ic_1 or ic_3 directly are filtered out after the execution of subquery* Candidates_1. *The subquery* Candidates_2 *filter out the tuples whose values of* orderpri *are impacted by the inconsistent values of* quan *according to ic_2. Finally, the last* select *clause return the result set on the projected attribute oid.*

Algorithm 3. *ConsistentQuery*(Q, IC)*: Get the consistent query answers.*
Consider a query Q of following form:
select $\mathcal{S}$ *from* $\mathcal{R}$
where $\mathcal{W}$ *order by* $\mathcal{O}$
$\mathcal{S}$*: the list of attributes selected;*
$\mathcal{W}$*: the* where *clause consisting of several predicates;*
$\mathcal{O}$*: the list of attributes ordered by*
$AttrA := \emptyset$

foreach attribute $p \in \mathcal{S} \mid\mid p \in \mathcal{W} \mid\mid p \in \mathcal{O}$
$AttrA := AttrA \cup p$
$conExp := CalCons(Attr)$
$i := 0$
foreach conjunctive expression c in conExp
$Condidates_{i+1} := CalCandidates(i, c)$
$i := i + 1$
$conAnsers :=$ *select* $\mathcal{S}$ *from* $Candidates_i$ *order by* $\mathcal{O}$

ConsistentQuery is the rewriting algorithm for queries without aggregation expressions. Without loss of generality, we consider the queries without aggregation in following form:

select $\mathcal{S}$ *from* $\mathcal{R}$
where $\mathcal{W}$ *order by* $\mathcal{O}$

Notice that, most of queries are of the above form. Although there are also some queries with nested subqueries, most of them can be unnested.

First the procedure *CalCons* produces the extended integrity constraints $\Psi(\bar{v})$ for the attributes $\bar{v}$ involved int the query. The $\Psi(\bar{v})$ is a disjunctive expression consisting of several conjunctive formulas. Second, we modify the original query q to q' with the *order* clause and projection removed, and then compute the result set of q' as initial set of candidates. The purpose of removing projection is to preserve the attributes which might be used in the following procedure of filtering the inconsistent tuples. By only focusing on the candidates, we can reduce the quantity of tuples for the following filtering sharply. For each conjunctive expression c in $\Psi(\bar{v})$, we filter out the tuples violating c. The tuples left are considered as candidates for the next iteration until all the conjunctive expressions

are processed. Finally, we calculate the consistent query answers according to the last set of candidates by adding projection and the order clause back.

The algorithm *CalCandidates* is to filter out the inconsistent tuples for a conjunctive constraint expression. It takes two parameters as input: a set of candidates and a conjunctive expression which should be satisfied. For each relation predicate in conjunctive c, if it is not in the set of relations of the input set of candidates, we join it with the candidates on certain attributes; otherwise if it is already joined in the candidate but with other attributes, we add a new equivalent expression on the join attributes to the where clause. Last, the formulas referring to built-in predicates in c are merged into the *whereClause* to filter the inconsistent tuples.

Algorithm 4. *CalCandidates(i, c): For a given conjunctive clause c, calculate the subquery* $Candidates_i$
R_p*: the set of relations which are included in* $Candidates_i$*;*
A_p*: the set of attributes which appear in relation predicates of c;*
selectClause := null

WhereClause := null
joinClause := null
foreach relation predicate p in c
  if p is not in R_p
    joinClause := joinClause + "join p"
    foreach attribute $v \in p$
      if $v \in c$
        joinClause := joinClause + "on p.a = $CandidateSet_i.a$*"*
  else if p is joined with other attributes
    foreach join attribute v_j
      whereClause := whereClause + "and"+ $cand_i.v = p.v_j$
foreach built-in predicate p in c
  whereClause := whereClause + "and"+ tranPredictR(p)
$CandidateSet_{i+1}$ *as (select * from* $CandidateSet_i$ *+ joinClause+ whereClause)*
return $CandidateSet_{i+1}$

5 Aggregation Query Rewriting

In this section, we present rewriting strategy for the queries that may have grouping and aggregation.

Example 7. (Ex1 continued) *Consider a query* Q_4 *that computes the average percent of discount of all items in relation* lineitem.

Q_4*: select avg(disc) as aveDisc*
from lineitem

Take tuple l_1 *and* l_5 *into consideration. As illustrated in example 1, the* disc*'s value of* l_2 *is inconsistent w.r.t* ic_1 *and the value of* l_1 *is inconsistent due to the transitivity of* null *value of* ic_1*. As we know, the value of* disc *is allowed with a value between 0 and 0.1 inclusive. Both of the value of* l_1 *and* l_5 *are in the*

range. Hence we assume l'_1 *as* l_1*'s repair:* l'_1*(L10, o105, null, [0, 0.10], book1) and* l'_5 *as* l_5*'s repair:* l'_5*(L14, o211, null, [0, 0.10], cd2). The result for query* Q_4 *is a range [0.026, 0.064].*

The bound strategy for the aggregate expressions is used in [1,10], which returns a range rather than a exact value. In this work, we also adopt this method for queries with aggregation. Like [10], we consider SQL queries of following form:

select $\mathcal{A}$*,* $Agg(e_1)$ *as* $E_1, \cdots,$ $Agg(e_k)$ *as* E_k
from $\mathcal{R}$
where $\mathcal{W}$ *and group by* $\mathcal{A}$

Definition 8. (Consistent answer for aggregate expression) Consider an inconsistent database instance I and a set of integrity constraints IC. q is a query with aggregation expressions. $Agg(E)$ is an aggregate expression. I_{con} is a set of consistent tuples after applying consistent query on attributes which are involved in aggregate expressions and I_{incon} is a set of inconsistent tuples. $Agg(E)(I)$ is computed as follows:

- Consider the situation that the domain of each attribute v_i involved in aggregation expression is finite. I'_{con} is a ranged set of I_{con}, where each tuple $\bar{t}$ in I'_{con} comes from I_{con} and its value of v_i is a range whose bottom bound and upper bound are both the original value of $\bar{t}$. I'_{incon} is a ranged set of I_{incon}, where each tuple $\bar{t}$ in I'_{incon} comes from I_{incon} and its value of v_i is a range whose bottom bound is the bottom bound of DOM(v_i) and upper bound is the upper bound of DOM(v_i). The consistent answers for $Agg(E)(I)$ is:

$$Agg(E)(I) = Agg(E)(I'_{con} \bigcup I'_{incon})$$

- If domain of some attributes is infinite or undefined, the consistent answers for $Agg(E)$ is:

$$Agg(E)(I) = Agg(E)(I'_{con})$$

The algorithm *ConsistentQueryAgg* is for rewriting queries with aggregation and grouping. Unlike algorithm *ConsistentQuery*, we calculate the extended constraints for the set of attributes in aggregation expressions and for the set of attributes in non-aggregation expressions separately. If the attributes v_i involved in aggregation have a finite domain, the values of v_i of inconsistent tuples should be repaired with a range, whose bounds are the bounds of DOM(v_i). This task is finished by algorithm *CalFilteredSet*. The aggregation attributes of final consistent tuples $Candidates_n$ should be also modified with a range for the union operations with the ranged $FilteredSet$s. The bottom and upper bound values of $Candidates_n + 1$ are both the original value. If the domain of the attributes involved in aggregation is undefined, those attributes should be excluded for the query.

Algorithm 5. *ConsistentQueryAgg(i, c):Calculate the consistent answers for aggregation queries*
Consider a query Q of following form:
select $\mathcal{S}$, $agg_1(e_1)$ *as* E_1, ... , $agg_n(e_n)$ *as* E_n
from $\mathcal{F}$ *where* $\mathcal{W}$
group by $\mathcal{G}$ *order by* $\mathcal{O}$
$AttrWGO := \emptyset$

foreach attribute $v \in \mathcal{W} \mid\mid v \in \mathcal{G} \mid\mid v \in \mathcal{O}$
$AttrWGO := AttrWGO \cup v$
$conWGO := CalCon(AttrWGO)$
$Candidates_0 := ConsistentQuery(q', IC)$
foreach $e_i \in Q$
$AttrE_i := \emptyset$
foreach attribute $v \in e_i$
$AttrE_i := AttrE_i \cup v$
$conE_i := CalCons(AttrE_i)$
foreach $i = 1$ *to* $|conE|$
$FilteredSet_i := CalFilteredSet(i, conE_i)$
$Candidates_i := CalCandidates(i, conE_i)$
$n := i$
$Candidates_{n+1}$ *as (*
select $\mathcal{S}$, v_1 *as bottom*(v_1), v_1 *as upper*(v_1)
,..., v_n *as bottom*(v_n), v_n *as upper*(v_n)
from $Candidates_n$*)*
$conAnswers :=$ *select* $\mathcal{S}$,
$agg_1(e_1)$ *as* E_1, ... , $agg_n(e_n)$ *as* E_n
from ($FilteredSet_1$ *union all ...*
union all $FilteredSet_n$
union all $Candidates_{n+1}$*)*
from $\mathcal{F}$ *where* $\mathcal{W}$
group by $\mathcal{G}$ *order by* $\mathcal{O}$

Algorithm 6. *CalFilterSet(i, c): Calculate the filtered set for given candidates.*
R_p*:the set of relations which are in the candidates.*
$joinClause := \emptyset$

foreach relation predicate $p \in c$
if $p \notin R_p$
$joinClause := joinClause +$ *"join p"*
foreach attribute $v \in p$
if $v \in c$
$joinClause := joinClause +$
"on p.a = $CandidateSet_i$*.a"*
$whereClause := \emptyset$

foreach built-in predicate $p_b \in c$
  $whereClause := whereClause$ + "or"
    $tranPredict(p_b)$
$FilterSet_{i+1}$ *as* (
  select $\mathcal{S}$, $bottom(v_1)$, $upper(v_1)$
    ,..., $bottom(v_n)$, $upper(v_n)$
  from $CandidateSet_i$
  $joinClause$
  $whereClause$)
return $FilterSet_{i+1}$

6 Experiment

In this section, we report the results for the experiment in order to quantify the overhead of the execution of the *SQL* rewriting generated by our algorithm.

6.1 The Setting for the Experiments

The experiment were performed on a Dell Optiplex 170L PC with a 2.8 GHz Intel Pentinum 4 CPU and 1GB of RAM using oracle 10g as the relational database under Windows XP Professional.

We present experimental results for queries 3, 6, 10, 12 14 which are taken from the TPC-H specification. The queries in the TPC-H specification are parameterized. In the experiments, we used the values which the standard suggests in all the queries. For each query, the number of relations in the *from* clause, the number of attributes in the *select* clause and the number of attributes in aggregate expressions are provided as follows.

We experimented with a number of inconsistent database, considering the following parameters:

- The size s of the database. We considered database of 0.5GB, 1GB, 2GB and 3GB. A database of 1GB has 8,661,245 tuples.
- The percentage p of the database that is inconsistent, For example on a 1GB database instance where p is 30%, there are total about 2.5 million tuples violate integrity constraints (ic). Without loss of generality, we assume each constraint with same quantity of inconsistent tuples.
- The max number n_r of the database relations involved in all connected subgraphs of ICRG. A connected subgraph is a component where each node can reaches all of other nodes. More relations in a subgraph means that the integrity constraints are more complicated.

DBGEN is a recommended data generator of TPC-H specification. It assigns random values in the domain to some attribute of tuples and certain special values to the others[1]. Hence we develop a program to create inconsistent database instances based on DBGEN. The program works as follows. Support that we

[1] The attribute *o_shippriority* of all tuples in relation *orders* is initialized with 0.

want to generate an instance of 0.5GB where p=30% on with three integrity constraints. We use DBGEN to create a random database instance first. Then we extract the inconsistent tuples. For each *ic*, if there are not enough inconsistent tuples (0.05GB), we modify the consistent tuple into inconsistent, otherwise change the extra inconsistent tuples into consistent.

	Relation	ProjAttrs	AggrAttrs
Q3	3	4	1
Q6	1	1	1
Q10	4	8	1
Q12	2	3	2
Q14	2	2	2

Fig. 3. Propertes of queries

6.2 Experimental Results

In the rewriting queries, the common subquery expressions are specified in a *WITH* clause. By this strategy, the result of these subquery are temporarily stored rather than compute several times. Hence the execution time will improved considerably.

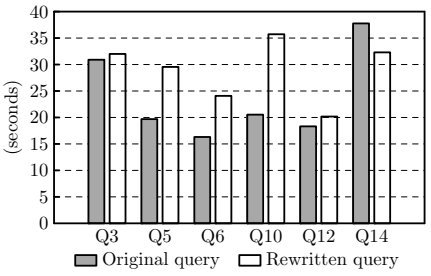


Fig. 4. Execution time of all queries

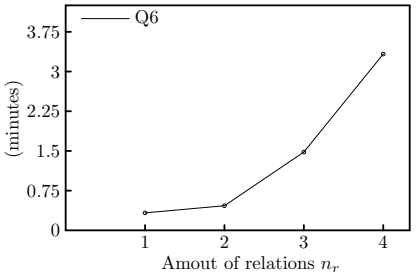


Fig. 5. Execution time of Q6 over n_r

In Figure 4, we first compare the performance of all queries and rewritings on a 1GB database and a *IC* set containing three integrity constraints, where $p = 10\%$, $= 3$, $n_r = 2$. Each integrity constraint have common attributes with each other. The increasing execution time ranges from 3.5%(Q3) to 61%(Q10) except Q14. We argue that the size of candidates which satisfy the original query condition with all grouping removed will impact the overhead of the rewritten queries. In the algorithm *ConsistentQuery*, we remove all the grouping of original query and take the result set as initial candidates for following computations. The size of candidates for *Q6* and *Q10* is 114,705 and 114,160 and the sizes of candidates for other queries are bellow 30 thousand. The reason for the exception of *Q14* is that there is *like* expression in *where* clause which takes high cost. The rewriting of *Q14* filters the inconsistent tuples to reduce the size of candidates for the *like* operation.

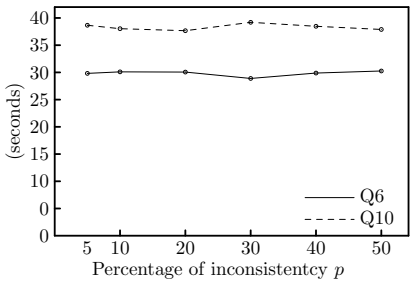


Fig. 6. Execution time of queries over p

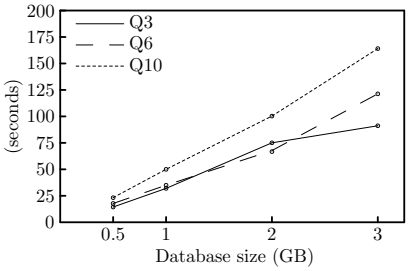


Fig. 7. Execution time of rewritten queries over database size

We also studied the effect of the amount of the relations in the IC Connected Graph (ICG), we post four different groups of integrity constraints. Each group has 4 ic*s* and different amount of relations involved. In Figure 5, We report results for one query (Q6) for 1GB database with 10% inconsistency since that Q6 involves just one relation. The result shows that more relations involved in a connected component of ICRG, the overhead of rewriting increases more fast. This is caused by the join operations between different relations. However, in most applications, the size of database relations involved in related integrity constraints is small. Hence this will not bring too much overhead. In Figure 6, we found that the percentage p has little influence on the execution time of the original queries and the rewriting.

Finally, the scalability of CQA based on repairing with *nulls* is shown in Figure 7. We explored databases of 0.5GB, 1GB, 2GB, 3GB with 10% inconsistency and an *ic* set of size 3. We report the execution time of the rewriting for queries Q3, Q6, Q10 and Q12. The result shows that the running time grow in linear time complexity with the size of the database.

7 Related Work

The notion of *consistent query answer* (CQA) is first defined in [4] based on provability in minimal logic. [1] proposes a query rewriting algorithm for *quantifier-free* first-order logic under binary universal integrity constraints and provides the proof of completeness, soundness and termination in theory. Since [1] there have been plenty of researches in computational complexity of CQA [7,14,13] and some consistent query system with powerful logic, such as [16]. However these programs focus on expressiveness of constraints and queries rather than scalability. Most of this system are only applicable for small size database due to the high computational complexity.

The database repairs considered so far have not explicitly consider that a database may have *nulls*. [3] used a different, more syntactic approach that simulates SQL nulls within a logic programming method to repair specifications. In our work, we provide a formal definition of repair semantics with *nulls* and a rewriting algorithm under denial integrity constraints.

ConQuer [10] is a CQA system using sql rewriting. It proposes a rewriting algorithm for C_{forest} queries in a limited context that each relation has at most one key function dependency. Hippo [8] is another system, which uses *conflict-hypergraph* to computes the consistent answers by a Java programm rather than existing database system. This method is infeasible for large-scale database due to high computational complexity and memory cost for constructing the hypergraph.

There are also considerable work in repairing the inconsistent database. [2] presented a approach to repair the values of attributes automatically, they are both based on certain cost functions and distance definitions to "guess" the suitable value of inconsistent attributes. [17] introduce a method to construct a nucleus: a single database to rectify errors within a tuple. However these methods are still with considerably high computational complexity.

8 Conclusion and Future Work

In this paper, we first present a repair semantics that can by applied to statistical analysis. Then we provide a rewriting algorithm for queries with respect to denial integrity constraints to fetch consistent answering. The experiment shows that our approach is feasible and efficient for various queries under different percentage of inconsistency and large-scale database by utilizing existing database system. We will continue to research on how to apply CQA in real application with properly repair semantics.

Acknowledgements. This work was partially supported by the National Natural Science Foundation of China (61202065), the National Grand Fundamental Research 973 Program of China (2009CB320704), the National Key Technology R&D Program (2012BAH05F02), and the National High-Tech Research and Development Plan of China (SS2012AA010104).

References

1. Arenas, M., Bertossi, L., Chomicki, J.: Consistent query answers in inconsistent database. In: ACM Symposium on Principles of Database System (PODS), pp. 68–79. ACM Press (1999)
2. Bohannon, P., Fan, W., Flaster, M., Rastogi, R.: A cost-based model and effective heuristic for repairing constraints by value modification. In: Proceedings of the ACM International Conference on Management of Data, pp. 143–154 (2005)
3. Bravo, L., Bertossi, L.: Semantically correct query answers in the presence of null values. In: Grust, T., Höpfner, H., Illarramendi, A., Jablonski, S., Fischer, F., Müller, S., Patranjan, P.-L., Sattler, K.-U., Spiliopoulou, M., Wijsen, J. (eds.) EDBT 2006. LNCS, vol. 4254, pp. 336–357. Springer, Heidelberg (2006)
4. Bry, F.: Query answering in information system with integrity constraints. In: Proceedings of the IFIP TC11 Working Group 11.5, First Working Conference on Integrity and Internal Control in Information Systems, pp. 113–130 (1997)
5. Chandel, A., Koudas, N., Pu, K.Q., Srivastava, D.: Fast identification of relational constraint violations. In: Proceedings of International Conference on Database Engineering, pp. 776–785 (2007)

6. Chomicki, J.: Consistent query answering: Five easy pieces. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 1–17. Springer, Heidelberg (2006)
7. Chomicki, J., Marcinkowski, J.: Minimal-change integrity maintenance using tuple deletions. Information and Computation, 90–121 (2005)
8. Chomicki, J., Marcinkowski, J., Staworko, S.: Computing consistent query answers using conflict hypergraphs. In: Proceedings of ACM International Conference on Information and Knowledge Management, pp. 417–426 (2004)
9. Flesca, S., Furfaro, F., Parisi, F.: Consistent query answers on numerical databases under aggregate constraints. In: Bierman, G., Koch, C. (eds.) DBPL 2005. LNCS, vol. 3774, pp. 279–294. Springer, Heidelberg (2005)
10. Fuxman, A., Fazli, E., Miller, R.J.: Conquer: Efficient management of inconsistent database. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 155–156 (2005)
11. Kuper, G., Paredaens, J., Libkin, L.: Constraint Databases. Spinger (1999)
12. Liu, J., Huang, F., Ye, D., Huang, T.: Efficient consistent query answering based on attribute deletions. In: Proceedings of International Symposium on Computer Science and its Applications (2008)
13. Lopatenko, A., Bertossi, L.: Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 179–193. Springer, Heidelberg (2006)
14. Pema, E., Kolaitis, P.G., Tan, W.-C.: On the tractability and intractability of consistent conjunctive query answering. In: Proceedings of the 2011 Joint EDBT/ICDT Ph.D. Workshop, pp. 38–44. ACM, New York (2011)
15. Rahm, E., Do, H.H.: Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin 24(3), 3–13 (2000)
16. Eiter, T., Fink, M., Greco, G., Lembo, D.: Efficient evaluation of logic programs for querying data integration systems. In: Palamidessi, C. (ed.) ICLP 2003. LNCS, vol. 2916, pp. 163–177. Springer, Heidelberg (2003)
17. Wijsen, J.: Database repairing using updates. ACM Transactions on Database System, 722–768 (2005)

On Efficient *k*-Skyband Query Processing over Incomplete Data

Xiaoye Miao[1], Yunjun Gao[1], Lu Chen[1], Gang Chen[1], Qing Li[2], and Tao Jiang[3]

[1] College of Computer Science, Zhejiang University, Hangzhou, China
{miaoxy,gaoyj,chenl,cg}@zju.edu.cn
[2] Department of Computer Science, City University of Hong Kong, Hong Kong, China
itqli@cityu.edu.hk
[3] College of Mathematics Physics and Information Engineering, Jiaxing University, China
jiangtao_guido@yahoo.com.cn

Abstract. The Skyline query and its variants have been extensively explored in the literature. Existing approaches, except one, assume that all dimensions are *available* for all data items. However, many practical applications such as sensor networks, decision making, and location-based services, may involve *incomplete* data items, i.e., some dimensional values are *missing*, due to the device failure or the privacy preservation. In this paper, for the first time, we study the problem of efficient *k-Skyband* (*k*SB) *query processing on incomplete data*, where multi-dimensional data items are missing some values of their dimensions. We formalize the problem, and then present several efficient algorithms for tackling it. Our methods employ some novel concepts/structures (e.g., *expired skyline*, *shadow skyline*, *thickness warehouse*, etc.) to improve the search performance. Extensive experiments with both real and synthetic data sets demonstrate the effectiveness and efficiency of our proposed algorithms.

1 Introduction

Skyline query has been shown as a powerful tool in many applications involving multi-criteria decision making. Given a set S of multi-dimensional data objects, a *skyline query* returns all the objects that are *not dominated* by any other object in S. Here, an object o *dominates* another object o' iff o is *not worse* than o' in all dimensions and strictly *better* than o' in at least one dimension. A classical example of hotel reservation system is depicted in Figure 1(a), where each point in a 2D space corresponds to a hotel record. The x-axis captures the distance (of every hotel) to the beach, while the y-axis represents its room price. In this case, hotel d dominates b since d is *cheaper* and *closer* to the beach. As hotels d, h, and j are not dominated by any other hotel, they constitute the *skyline* of the set $S = \{a, b, c, d, e, f, g, h, i, j\}$. All these skyline hotels offer more interesting and preferable choices for users.

The Skyline query and its variants have been extensively explored in the literature. Existing approaches, except one [11], assume that all dimensions are *available* for all data items. However, *incomplete* data items exist in many real applications such as sensor networks, decision making, and location-based services, i.e., some dimensional values are *missing*, due to the device failure or the privacy preservation. For instance, when people are invited to fill out a form (e.g., questionnaire), some ones are not willing to provide their privacy information (e.g., age, occupation, etc.). Hence, the

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 424–439, 2013.

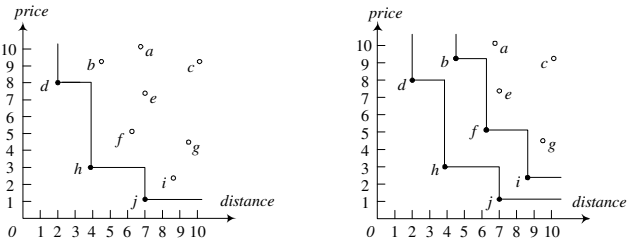


Fig. 1. Illustration of skyline and *k*-Skyband queries

data in the questionnaire might be *incomplete*. Another example is that, in a sensor network, sensor data cannot be captured at a certain timestamp because of the device failure. Thus, the sensor data may also be incomplete. Motivated by these, query processing over incomplete data has been paid attentions by the database community [1, 4, 10, 11, 15, 20]. In addition, skyline queries on incomplete data may not report all of the desirable objects, due to the *non-transitive* and *cyclic* of the dominance relationship (to be further discussed in Section 3).

In this paper, for the first time, we study the problem of *k-Skyband* (*k*SB) *query processing on incomplete data*, which retrieves the incomplete objects that are dominated by *at most k* other objects. Our problem is inherently *different* than the existing *uncertain* queries. First, the uncertainty is expressed in terms of probabilities. Second, the uncertain query assumes that the probabilities for some data items have been specified for the original dataset. However, in practical applications (e.g., Google Squared), it is usually not easy for each data item to set the probabilities. Consequently, the algorithms for processing uncertain queries are *not applicable* to the *k*SB query over incomplete data.

A naive solution is to use bucket to partition the whole dataset according to the bitmap, and then compute the *k*-Skyband of the dataset. Unfortunately, this solution is *inefficient* for a large data set due to the *exhaustive comparisons*. To address this, we propose (i) the *Virtual Point based algorithm* (VP), which uses the concepts/structures of *virtual point*, *expired skyline* (ES), and *shadow skyline* (SS); and (ii) the *k-iSkyband algorithm* (*k*ISB), a specific algorithm designed for the *k*SB query on incomplete data, which employs two novel concepts/structures, namely, *thickness warehouse* (TW) and ES, to further improve the search performance. In brief, the key contributions of this paper are summarized as follows:

- We identify the problem of *k*SB query processing in the context of incomplete data. To the best of our knowledge, there is *no prior work* on this problem.
- We formalize the *k*SB query over incomplete data, and analyze its properties.
- We utilize some novel concepts/structures (e.g., ES, SS, TW, etc.), and propose several efficient algorithms to compute *k*-Skyband objects.
- We conduct extensive experimental evaluation using both real and synthetic data sets to verify the effectiveness and efficiency of our proposed algorithms.

The rest of the paper is organized as follows. Section 2 briefly reviews the related work. Section 3 presents the problem formulation and its characteristics. Section 4 elaborates three algorithms for efficiently processing the *k*SB query on incomplete data, and

discusses their correctness and time complexities. Extensive experiments and our findings are reported in Section 5. Finally, Section 6 concludes the paper with some directions for future work.

2 Related Work

In this Section, we overview the existing work related to the *k*SB query on incomplete data, namely, traditional skyline queries, skyline queries over uncertain data, and queries on incomplete data.

Traditional Skyline Queries. Borzsonyi et al. [3] first introduce the skyline operator into the database community, and develop two algorithms, i.e., *Block Nested Loops* (BNL) and *Divided-and-Conquer* (D&C), for computing the skyline. Chomicki et al. [6] present the *Sort-Filter-Skyline* (SFS) algorithm that is an improved version of BNL. Bartolini et al. [2] propose the *Sort and Limit Skyline algorithm* (SaLSa) to avoid scanning the complete set of sorted data objects. Kossmann et al. [12] develop a *Nearest Neighbor* (NN) approach to obtain the skyline using an R-tree. Papadias et al. [16] improve the NN method, and propose a *Branch and Bound Skyline* (BBS) algorithm to retrieve the skyline by traversing the R-tree once in the best-first manner. Zhang et al. [23] investigate a dynamic indexing technique for skyline objects. Recently, many variations of the skyline operator have also been extensively explored, such as *k*-Skyband [16], reverse skyline [7], reverse *k*-Skyband [14], subspace skyline [18], metric skyline [5, 8], to name but a few. Unfortunately, none of all these algorithms is designed to aim at incomplete data.

Skyline Queries on Uncertain Data. Pei et al [17] first introduce the skyline query over uncertain data, and develop the *bounding-pruning-refining* techniques to handle it efficiently. Lian and Chen [13] investigate the probabilistic reverse skyline query in both monochromatic and bichromatic fashion. Zhang et al. [22] explore efficient skyline computation against sliding windows on uncertain data stream, using the specified probability thresholds. Recently, ranking with uncertain score functions [21] and probabilistic threshold query [19] are studied as well. Nonetheless, as pointed out in Section 1, these problems differ from ours because the uncertainty is expressed in terms of probabilities, while the incompleteness means some dimensional values of the data items are missing without the knowledge of probabilities. Therefore, the above algorithms cannot be used to solve our problem.

Queries on Incomplete Data. The most related work to ours is the problem of skyline query over incomplete data [11]. Specifically, Khalefa et al. [11] define the dominance relationship on incomplete data, and describe the *non-transitive* and *cyclic* properties. Then, three algorithms, namely, *Replacement*, *Bucket*, and *ISkyline*, are proposed, and the ISkyline algorithm performs the best. Other queries over incomplete data include ranking queries [20] and top-*k* queries [9]. In addition, Imielinski and Lipski [10] discuss the fundamental concept of *c*-table and its restrictions to simple tables with variables. Indexes on incomplete databases are studied in [4, 15]. Antova et al. [1] present the World-set Algebra language that can map from a complete database to an incomplete one. To our knowledge, this is the first attempt on the problem of *k*-Skyband query processing over incomplete data.

3 Problem Formulation

In this section, we present the dominance relationship over incomplete data, and then formally define the k-Skyband query on incomplete data.

For *complete* data, an object o dominates another object o' iff o is not worse than o' in all dimensions, and strictly better than o' in at least one dimension. For *incomplete* data, however, with the existence of incomplete dimensional values, the traditional definition of dominance relationship is not applicable any longer. Without loss of generality, we assume that the *smaller* value is *better*, and at least one dimensional value of any data object is known. Based on the two assumptions, the dominance relationship over incomplete data is defined below.

Definition 1 (Dominance Relationship on incomplete data) [11]. *Given any two D-dimensional objects o and o′, with missing values on some dimensions, and let o*.[*i*] *be the i-th dimensional value of o, object o dominates another object o′ if the following two conditions hold*: (1) *for all dimensions i, either o*.[*i*] *is unknown and/or o′*.[*i*] *is unknown, or o*.[*i*] ≤ *o′*.[*i*]; *and* (2) *there is at least one dimension j* (≠ *i*), *where both o*.[*j*] *and o′*.[*j*] *are known, and o*.[*j*] < *o′*.[*j*].

In the rest of paper, we use a dash "-" to represent a missing value. For example, a three-dimensional object o is denoted as (-, 5, 3), if its first dimensional value is missing and the last two dimensional values are 5 and 3. For example, object m = (-, 4, 5, -) is said to dominate object n = (4, 6, -, 8) since m.[2] (= 4) < n.[2] (= 6) satisfies. As another example, considering two objects p = (-, 6, 8, -) and q = (4, -, -, 7), they do not dominate each other as they do not have *common known dimension*.

Unfortunately, as pointed out in [11], the dominance relationship over incomplete data loses the *transitivity*, which is the basis of almost all previous skyline query processing algorithms. Furthermore, there may have a *cyclic* dominance relationship, meaning that *none* of the objects of a data set is considered as a skyline object. Consider, for instance, object l = (5, -, 4, 10) and the aforementioned objects m and n. According to Definition 1, n dominates l since both n.[1] < l.[1] and n.[4] < l.[4] hold, and l dominates m due to l.[3] < m.[3]. Nevertheless, as mentioned above, when comparing n with m, m dominates n (rather than n dominates m), indicating that the dominance relationship on incomplete data is *cyclic* (i.e., *non-transitive*). In this case of cyclic dominance, none of the three objects can be considered as a skyline object.

Take a data set $S = \{m, n, l\}$ as an example. If we assume that all the missing values are 3, namely, m = (3, 4, 5, 3), n = (4, 6, 3, 8), and l = (5, 3, 4, 10) in which the numbers with underlines represent the missing values, then we observe that all the three objects are skyline objects, since they are not dominated by each other. Thus, the results reported by the skyline query on incomplete data may ignore some *qualified* objects, because a desirable object may be discarded by the algorithm if an important and essential value is missing. Consequently, the skyline query over incomplete data can not offer all the desirable choices due to the non-transitive and cyclic dominance relationship. On the other hand, some objects in the result returned by the skyline query on incomplete data may also be the *unqualified* objects. For example, consider a data set $E = \{e_1, e_2, e_3\}$, where e_1 = (-, 4, -, 2), e_2 = (5, -, 3, -), e_3 = (3, 7, -, -). It is easy to say that e_1 is the skyline of the data set E. However, if e_1 = (6, 4, 5, 2). e_2 = (5, 2, 3, 1), and e_3 = (3, 7, 3, 2) in the real *complete* data set, objects e_2 and e_3 (instead of e_1) are the skyline of the data set.

In order to get the most desirable objects and provide more preferable choices for users, the k-Skyband query provides an opportunity, which can get more interesting objects. Motivated by this, in this paper, we investigate the problem of k-Skyband query over incomplete data, and formalize it as follows.

Definition 2 (k-Skyband query on incomplete data). *Given a D-dimensional object set S where every object is missing some value(s) of its dimension(s), but has at least one known dimension, a k-Skyband (kSB) query over S find the set of k-Skyband objects $S_{sb} \subseteq S$ such that each object $o \in S_{sb}$ is dominated by at most k objects in S, while every object $o' \in S - S_{sb}$ is dominated by more than k objects in S.*

Actually, the argument k is a parameter of *skyline thickness*. In particular, the skyline query is a conventional k-Skyband query when $k = 0$. For instance, Figures 1(a) and 1(b) illustrate the examples of 0-Skyband query (i.e., Skyline) and 2-Skyband query (on complete data), respectively. In Figure 1(b), the object b belongs to the result of the 2-Skyband query as only two objects d and h dominate the object b. Nevertheless, the object e does not belong to the result of the 2-Skyband query since it is dominated by three objects f, h, and j.

In addition, like [11], we also utilize the bitmap vector for ease of representation and computation. A D-dimensional incomplete object corresponds to a bitmap vector of D bits, in which 1 and 0 denote the complete dimension and incomplete dimension, respectively. As an example, the bitmaps of the previous objects m = (-, 4, 5, -), n = (4, 6, -, 8), and l = (5, -, 4, 10) are 0110, 1101, and 1011, respectively. Hence, the two objects are comparable iff the bitwise-and of their bitmaps is not zero. For example, objects m and n are comparable as 0110 AND 1101 is 0100.

4 Algorithms for kSB Queries on Incomplete Data

In this section, we present three algorithms, namely, *Baseline algorithm*, *Virtual Point based algorithm* (VP), and *k-iSkyband algorithm* (kISB), for answering kSB queries over incomplete data, and then, analyze the correctness and time complexities of our proposed algorithms, respectively.

Since the dominance relationship on incomplete data is *non-transitive* and *cyclic* (mentioned in Section 3), we cannot directly apply the existing traditional kSB query algorithm [16] to tackle our problem. Similar to the bucket algorithm [11], a naïve solution is to *partition* all input objects into different buckets and the objects in every bucket have the same bitmap, and then utilize the traditional kSB query algorithm to compute the k-Skyband for the objects in the same bucket by ignoring all incomplete dimensions. Here, we call a set of k-Skyband for each bucket as the *local k-Skyband set* (LkSB), all the local k-Skyband sets as the *candidate k-Skyband set* (CkSB), and the final result as the *global k-Skyband set* (GkSB).

Consider, for instance, Figure 2, where 40 objects are divided into 5 buckets A, B, C, D, and E, based on their bitmaps. First, we compute L2SB (k = 2) for every bucket, denoted by the shaded area in Figure 2. Overall, we have 27 local 2-Skyband objects in C2SB. Then, we conduct *exhaustive pairwise comparisons* in C2SB to discard the *unqualified* candidate 2-Skyband objects in C2SB. Finally, in order to refine the final result, for each remaining candidate object in C2SB, we compare it with the *local*

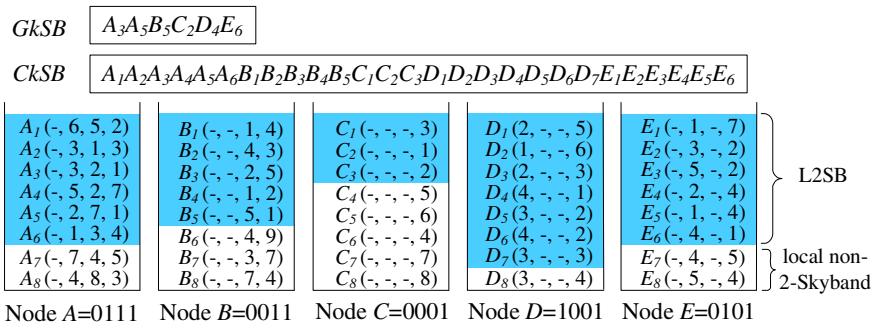


Fig. 2. Example of Baseline algorithm

non-2-Skyband objects of the comparable buckets, after which G2SB = {A_3, A_5, B_5, C_2, D_4, E_6} as the 2-Skyband of the specified 40 objects.

Recall that the example above, our *Baseline algorithm* has three phases: (1) **Phase 1:** Objects are partitioned into different buckets according to the bitmap, and LkSB is computed for every bucket. (2) **Phase 2:** In CkSB, objects are conducted exhaustive pairwise comparisons to prune unqualified candidates. (3) **Phase 3:** GkSB is obtained after comparing the remaining qualified objects in CkSB against all the *local non-kSB objects*. However, the Baseline algorithm is *very inefficient*, due to the *prohibitive comparisons* involved in Phase 2, especially for larger volume of CkSB.

Motivated by this, we develop two efficient algorithms, i.e., VP and kISB, for processing the kSB query over incomplete data. Our methods have *no* assumption of index availability and data pre-processing, and employ some novel techniques (e.g., *thickness warehouse*, *expire skyline*, etc.) to improve the search performance.

4.1 Virtual Point Based Algorithm

Given the fact that Baseline algorithm ignores the correlation among buckets, which may result in many objects in CkSB, we propose a new algorithm, namely, VP, which uses three concepts/structures, i.e., *virtual point*, *ES*, and *SS*, to reduce the number of comparisons. Before presenting VP, we briefly explain them.

The virtual point is employed to reduce the number of the objects in LkSB and thus the number of the objects in CkSB. The number of comparisons in CkSB significantly decreases accordingly. A virtual point is that when local kSB objects s, r in CkSB are from *different* buckets S, R, and s *dominates* r, the virtual point of s will be added to the bucket R for reducing the number of objects in the LkSB of S. As shown in Figure 2, A_2 and B_1 in L2SB are from different buckets, and A_2 dominates B_1. Hence, we insert the virtual object of A_2, denoted by A_{2v} (as shown in Figure 3), into the bucket B, after which B_3 is no longer a 2SB object in B since it is dominated by B_1, B_4, and A_{2v}. Here, A_{2v} = (-, -, 1, 3), which only maintains the values on the same dimensions as bucket B, and other dimensional values are represented as "-".

Similarly, we utilize the expired skyline (ES) to reduce the number of comparisons involved in CkSB. ES is to keep the discarded objects when performing comparisons in CkSB. As depicted in Figure 3, for example, A_4 is already dominated by A_2 and A_3

Algorithm 1. Virtual Point Based Algorithm (VP)

Input: an incomplete data set S, a parameter k
Output: the result set $GkSB$ of the kSB query on S
1: initialize sets $CkSB = ES = SS = GkSB = \emptyset$
2: **repeat**
3: read object o from S
4: node $N \leftarrow$ node corresponding to the bitmap of o
5: **if** N = NULL **then**
6: create and initialize node N with the bitmap of o
7: **end if**
8: *is-local* $\leftarrow$ Is-LkSB(o, N), or add o to SS if necessary
9: **if** *is-local* = TRUE **then**
10: Insert-CkSB(o), or add o to ES if necessary // using virtual point
11: **end if**
12: **until** the end of S
13: Insert-GkSB-with-ES($CkSB$, ES) // using ES
14: Insert-GkSB-with-SS($CkSB$, SS) // using SS
15: Get-GkSB($CkSB$) // using Lemma 1
16: **return** $GkSB$

in a bucket A. If we now evaluate A_4 and compare A_4 with C_2 in C2SB (k = 2), we identify that C_2 dominates A_4, and thus A_4 is not a 2-Skyband object any more, and it is pruned away and is preserved in ES to avoid unnecessary comparisons later. After finishing all the comparisons among the objects in CkSB, the remaining candidate objects are compared against the objects in ES to further refine the result.

For each bucket, we utilize the shadow skyline (SS) to store the objects which are dominated by more than k virtual objects, but are the real local skyline objects without taking into account *virtual points*. As an example, in Figure 3, an object D_2 is kept in the SS of a bucket D because, D_2 is dominated by virtual points A_{2v}, A_{3v}, A_{5v}, B_{4v}, and B_{5v}, and it is not dominated by any other object in D. In this way, after the candidate object A_3 compares with D_2, it is *unnecessary* for A_3 to compare against the objects D_6, D_7, and D_8 in the local non-kSB set of D, since there is *no* any object in D that can dominate A_3 yet. Here, it is worth noting that, the local non-kSB set is different from that in the baseline algorithm. In particular, it keeps the local non-kSB objects but not in SS with considering virtual points. Using SS, the candidate objects compare against SS before they compare with the local non-kSB set of every bucket, in order to avoid the comparisons with the local non-kSB set of *unnecessary buckets*. However, which is the unnecessary bucket? We answer it in Lemma 1 below.

Lemma 1. *Assume that a candidate k-Skyband object o has compared against the objects in the SS of a bucket B. If o is not dominated by any object in B, there is* no *object in B dominating o. Here, we call B is the* unnecessary bucket *w.r.t. o.*

Proof. Assume that there exists an object b in the bucket B that dominates o but have not compared with o yet, the object b must be one of three cases: (1) **Case 1:** b belongs to the LkSB of B. For a local k-Skyband object, b has been added to CkSB, and hence it must compare against o either in CkSB or in ES, which contradicts with the assumption. (2) **Case 2:** b is in the SS of B. Clearly, this case cannot happen. (3) **Case 3:** b is preserved in the local non-kSB set of B. Then, there must exist an object b' that belongs to the LkSB of B and dominates b. Due to Case 1, this case cannot take place. Consequently, the proof completes. □

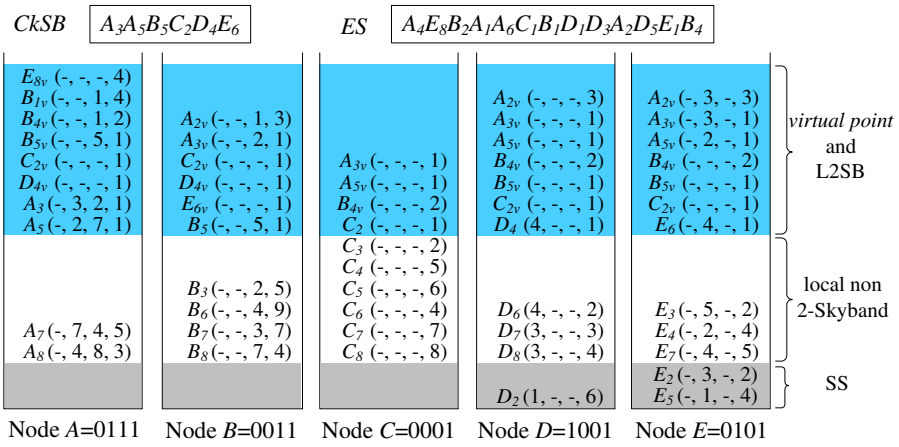


Fig. 3. Example of VP algorithm

The pseudo-code of VP algorithm is shown in Algorithm 1. It takes an incomplete data set S and a parameter k as inputs, and returns the result set *GkSB* of the kSB query over S. Initially, VP initializes sets *CkSB*, *ES*, *SS*, and *GkSB* (line 1). It then evaluates sequentially the objects in S (lines 2-12). For each object $o \in S$, VP gets the node N that corresponds to the bitmap of o, or creates N if it has not been created previously (lines 4-7). Next, the function Is-LkSB(o, N) is called to determine whether o is a local kSB object, or if necessary, add o to the set *SS* (line 8). If *yes*, VP invokes the function Is-CkSB(o) to decide whether it inserts o into the set *CkSB*, or store o in the set *ES* if necessary (lines 9-11). Also, the virtual point of o is added to the corresponding buckets if necessary. Thereafter, functions Insert-GkSB-with-ES(*CkSB*, *ES*), Insert-GkSB-with-SS(*CkSB*, *SS*), and Get-GkSB(*CkSB*) are used to refine *CkSB* and obtain *GkSB* by comparing all candidate objects in *CkSB* against all objects in *ES*, *SS*, and the local non-kSB set of necessary buckets (lines 13-15). Finally, the query result set *GkSB* is returned, and the algorithm terminates (line 16).

Example 1. Figure 3 depicts the example of VP algorithm, where the *deep* shaded area, the *light* shaded area, and the *white* area in each bucket represent virtual points and LkSB, SS, and the local non-kSB set, respectively. Also, the sets *CkSB* and *ES* are illustrated in Figure 3, after evaluating all objects. Note that, the number of objects in *CkSB* reduces significantly to 6 from 27, and the objects in *CkSB* are the actual result.

4.2 *k*-iSkyband Algorithm

It is worth pointing out that, in VP, virtual point may incur the redundant storage and comparisons, since the virtual point of a real data object might be inserted into many buckets and compare with the objects in those buckets. Motivated by this, we develop another novel algorithm, namely, kISB, which exploits the characteristic of k-Skyband to efficiently prune the unqualified objects in CkSB as early as possible, and employs the two concepts/structures, i.e., *TW* and *ES* (discussed in Section 4.1), to improve the efficiency of the search. We explain TW briefly before discussing kISB.

Algorithm 2. *k*-iSkyband algorithm (*k*ISB)

Input: an incomplete data set S, a parameter k
Output: the result set $GkSB$ of the kSB query on S
1: initialize sets $TW = ES = GkSB = \emptyset$
2: **repeat**
3: read object o from S
4: node $N \leftarrow$ node corresponding to the bitmap of o
5: **if** N = NULL **then**
6: create and initialize node N with the bitmap of o
7: **end if**
8: Is-LkSB(o, N)
9: **until** the end of S
10: hash the objects in all LkSBs into TW based on their dominated times
11: Get-Candidate(TW), or add o to ES if necessary
12: Insert-GkSB-with-ES(TW, ES) // using ES
13: Get-GkSB(TW) // using Lemma 2
14: **return** $GkSB$

In order to avoid the exhaustive comparisons in CkSB, kISB hashes the objects in all LkSBs into TW according to their dominated times. Also, a *special comparison criterion* is defined based on TW. The comparison criterion is that the objects with *more* dominated times are verified *earlier*, and they are compared against the objects in *ascending order* of the dominated times. The reason is that the objects with more dominated times can be easily validated unqualified ones, and the objects with *less* dominated times have *more* powerful dominance capacity. For example, as illustrated in Figure 4, TW consists of three heaps storing the objects whose dominated times are 0, 1, and 2, respectively, in which the hash key of each heap is the dominated times. Then, we first compare the objects in the heap of key = 2 with the objects in the heap of key = 0, instead of comparing object pairs in the heap of key = 0. In this way, the total number of comparisons is reduced significantly.

Like VP, ES is also utilized to improve the performance of kISB. For instance, as shown in Figure 2, since there is *no* object from other buckets that dominates B_5 yet if the comparisons between B_5 and the objects in ES are finished, it is *unnecessary* for B_5 to compare against the LkSBs of the other four buckets. Similar to VP, we derive Lemma 2 to avoid comparing the remaining objects in TW with the local non-kSB set in unnecessary buckets (involved in the last phase of the Baseline algorithm).

Lemma 2. *For any object o (from a bucket O) in TW, if there is* no *object o' (from a bucket $O' \neq O$) in TW or ES which dominates o, all the objects in O' do not dominate o, i.e., there exists no object in O' dominating o.*

Proof. Assume that there is an object s in the bucket O' which dominates the object o but not in TW or ES. According to kISB, the object s belongs to the local non-kSB set of O', as all local kSB objects have been inserted into TW or ES. Thus, s is dominated by more than k objects, meaning that there exists an object r in the LkSB of O' which dominates s, and r must be in TW or ES. Since s dominates o, r dominates o due to s and r sharing with the same bucket. There exists an object r in TW or ES that dominates o, which contradicts with the condition of Lemma 2. Consequently, the proof completes. □

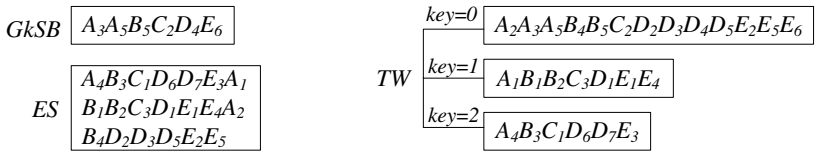


Fig. 4. Example of *k*ISB algorithm

The pseudo-code of *k*ISB algorithm is presented in Algorithm 2. In the first place, *k*ISB initializes sets *TW*, *ES*, and *GkSB* (line 1). Then, it evaluates sequentially the objects in an input incomplete data set *S* (lines 2-9). For each object $o \in S$, *k*ISB finds its corresponding node *N* using the bitmap of *o* (line 4), or create *N* if it does not exist currently (lines 5-7). Next, the function Is-L*k*SB(*o*, *N*) is invoked to insert the object *o* into the L*k*SB of the node *N* if *o* is a local *k*SB object (line 8). *k*ISB proceeds in the same manner until the end of *S*. Thereafter, *k*ISB hashes the objects in all L*k*SBs (i.e., C*k*SB) into *TW* based on their dominated times (line 10), calls Get-Candidate(*TW*) to conduct the comparisons in *TW* according to the aforementioned comparison criterion, or if necessary, add *o* to the set *ES* (line 11), uses Insert-G*k*SB-with-ES(*TW*, *ES*) to compare the objects in *TW* with the objects in *ES*, for refining the candidate objects (line 12). Finally, *k*ISB employs Get-G*k*SB(*TW*) to obtain and return the query result *GkSB*, by comparing the remaining candidate objects in *TW* against the local non-*k*SB objects of necessary buckets based on Lemma 2 (lines 13-14).

Example 2. Back to the running example illustrated in Figure 2, where the *deep* shaded area represents the L*k*SB of each bucket. Also, *GkSB*, *ES*, and *TW* involved in *k*ISB are depicted in Figure 4. As the thickness parameter *k* is 2, the *TW* contains three heaps based on the dominated times of every object. The objects in the same heap have the same key (i.e., the dominated times). For example, the objects A_2, A_3, and A_5 in the same heap have the same dominated times 0. According to the comparison criterion, the objects in the heap of *key* = 2 are firstly selected (e.g., C_1), and then, the object C_1 compares with some object in the heap of *key* = 0 (e.g., A_5). Since A_5 dominates C_1, the dominated times of C_1 increases 1, and thus, C_1 is inserted into *ES* and removed from *TW*. *k*ISB proceeds in the same fashion until all the objects in *TW* have been evaluated. In the end, the result set *GkSB* is returned after comparing the remaining objects in *TW* with the objects (needed to be compared) in *ES* and local non-*k*SB set of unnecessary buckets, and the algorithm terminates.

4.3 Discussion

In this subsection, we prove the correctness of our proposed algorithms (i.e., Baseline, VP, and *k*ISB), and then analyze their time complexities.

Theorem 1. *All the proposed algorithms find exactly the actual k-Skyband objects on incomplete data*, *i.e.*, *every algorithm has* no false negatives *and* no false positives.

Proof. First, no result is missed (i.e., no false negatives) as only *unqualified* objects are eliminated by the dominance relationship over incomplete data (see Definition 1), and only *unnecessary* buckets are pruned by Lemmas 1 and 2. Second, all the objects that

can be the k-Skyband objects are evaluated by comparing them against all other objects in order to ensure no false positive. □

Let $|S|$ be the cardinality of a dataset S, α be the average cost of inserting an object into its corresponding bucket, χ be the average number of objects needed to be compared in the sets *CkSB* (*TW*), and β be the average cost of refining the qualified objects in *CkSB* (*TW*) for obtaining the final result.

Theorem 2. *The time complexity of the algorithms Baseline*, *VP*, *and kISB is* $O(|S| \times \alpha + \chi^2 + \beta)$, $O(|S| \times \alpha_{VP} + \chi_{VP}^2 + \beta_{VP})$, *and* $O(|S| \times \alpha + \chi_{kISB}^2 + \beta_{kISB})$, *respectively*.

Proof. For the Baseline algorithm, it takes $O(|S| \times \alpha)$ to insert all of the objects into their corresponding buckets; $O(\chi^2)$ to conduct the comparisons in *CkSB*; and $O(\beta)$ to compare the remaining candidate objects in *CkSB* with the local non-kSB objects of their corresponding buckets. Hence, the total time complexity of Baseline is $O(|S| \times \alpha + \chi^2 + \beta)$. For the VP algorithm, it takes $O(|S| \times \alpha_{VP})$ to insert all of the objects into their corresponding buckets, where α_{VP} is *larger* than α due to virtual points; $O(\chi_{VP}^2)$ to perform the comparisons in *CkSB*, where χ_{VP} is *smaller* than χ as the number of candidates objects needed to be compared is reduced by using the virtual points; and $O(\beta_{VP})$ to obtain the final result by the comparisons between the remaining candidate objects and all the objects in *ES*, *SS*, and the local non-kSB objects of their corresponding buckets, where β_{VP} is *smaller* than β due to *ES*, *SS*, and Lemma 1. Thus, the total time complexity of VP is $O(|S| \times \alpha_{VP} + \chi_{VP}^2 + \beta_{VP})$. For the kISB algorithm, it takes $O(|S| \times \alpha)$ to insert all of the objects into their corresponding buckets (similar as Baseline); $O(\chi_{kISB}^2)$ to conduct the comparisons in *CkSB*, where χ_{kISB} is *smaller* than χ_{VP} as the number of candidates objects needed to be compared is further reduced by using TW. Also, $O(\beta_{kISB})$ is needed to compare the remaining candidate objects in *TW* against the objects in *ES* and the local non-kSB objects of the corresponding buckets. Here, β_{kISB} is *smaller* than β since kISB uses *ES* and Lemma 2 to reduce the unnecessary comparisons. Therefore, the total time complexity of kISB is $O(|S| \times \alpha + \chi_{kISB}^2 + \beta_{kISB})$.

5 Experimental Evaluation

In this section, we experimentally evaluate the performance of our algorithms, namely, Baseline, VP, and kISB. We describe the experimental settings, and then report the experimental results and our findings. All algorithms were implemented in C++, and all experiments are conducted on an Intel Core 4 Duo 2.8GHz PC with 4GB RAM, running Microsoft Windows XP Professional Edition.

5.1 Experimental Setup

In our experiments, we use both *real* and *synthetic* data sets to verify our algorithms. We employ two real datasets, i.e., *MovieLens* and *NBA*. (1) *MovieLens*: It contains 3900 movie records, where each object has 6040 dimensions representing the ratings of 6040 audiences. The rating values vary from 1 to 5, and the *higher* rating means

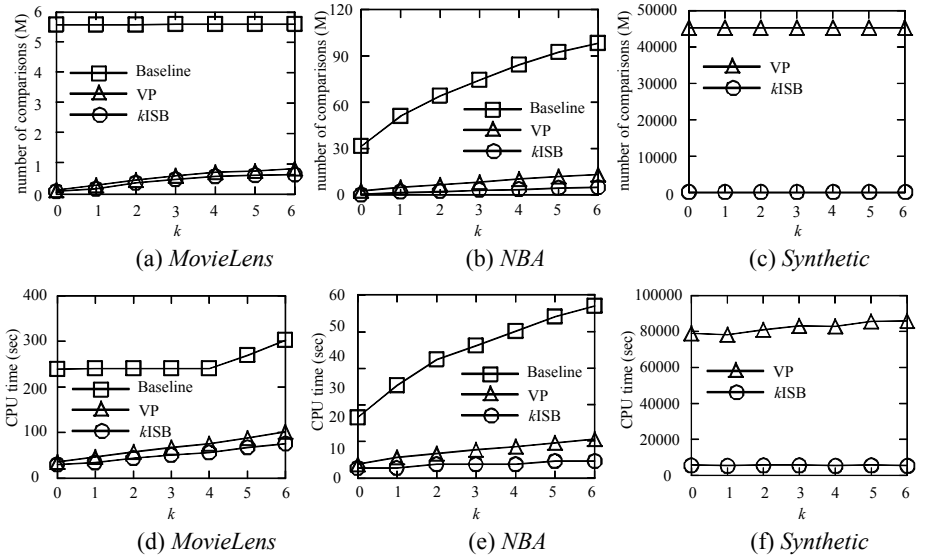


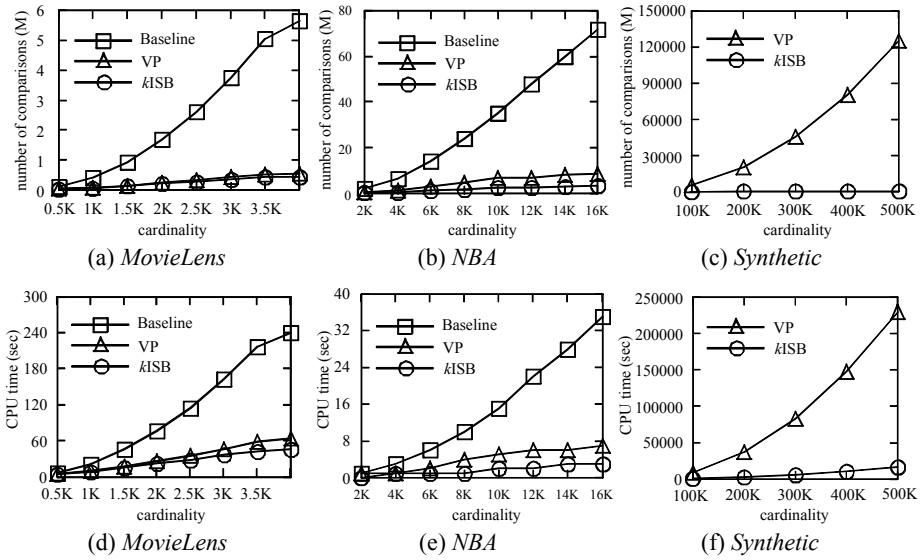
Fig. 5. kSB cost on incomplete data vs. *k*

the *better* recognition. The incomplete rate of *MovieLens* is 95%, i.e., only 5% of the ratings are available. (2) *NBA*: It includes the records for 16000 NBA players, where each record has 17 dimensions representing various technical statistics of every NBA player. Although the *NBA* data is *rather complete*, we explicitly remove some statistics in order to achieve the 20% *incomplete* rate for evaluating the performance of our algorithms. Following the common methodology in [11], we also create some 100-dimensional *Synthetic* datasets with cardinality in the range [100K, 500K] and incomplete rate in the range [5%, 80%].

We study the performance of our algorithms in terms of the number of comparisons and the CPU time. We investigate several factors, including the *k*-Skyband thickness *k*, cardinality *N*, dimensionality *dim*, incomplete rate *i*, and the rate of bucket number *r*. In each experiment, the *default* of *k*, *N*, *dim*, *i*, and *r* are 3, 300K, 60, 20%, and 20% respectively, and only one factor varies, whereas the others are fixed to their defaults.

5.2 Performance Study

The first set of experiments evaluates the effect of the *k*-Skyband thickness *k* on the efficiency of our algorithms. Figure 5 illustrates the number of comparisons and CPU time (in seconds) respectively with respect to *k* on real and synthetic datasets, varying *k* from 0 to 6. Clearly, in all cases, Baseline is *several orders of magnitude* worse than VP and *k*ISB, and *k*ISB performs the best. This is because the exhaustive pairwise comparisons involved in Baseline are *very costly*. In the sequel, we only focus on the experimental results of VP and *k*ISB over *Synthetic* datasets, since they always exceed Baseline in all cases. The CPU time increases with *k*. The reason is that, as *k* grows, more candidate objects need to be checked, resulting in more number of comparisons.



(a) *MovieLens* (b) *NBA* (c) *Synthetic*

(d) *MovieLens* (e) *NBA* (f) *Synthetic*

Fig. 6. kSB cost on incomplete data vs. N

The second set of experiments studies the impact of the dataset cardinality N on the performance of our algorithms. Figure 6 shows all the experimental results by varying N between 100K and 500K. Again, *k*ISB outperforms the other algorithms and the difference increases with N. This is because the number of the candidate k-Skyband objects grows as N ascends.

Then, we investigate the influence of the dimensionality *dim* on the efficiency of our algorithms. Figure 7 depicts all the experimental results on both real and synthetic datasets. As expected, *k*ISB also performs the best, and the cost increases with the growth of *dim*. The reason is that, as *dim* ascends, the dominance capacity of objects decreases, incurring more number of candidate k-Skyband objects.

Next, we inspect the impact of the incomplete rate i on the performance of our algorithms. Towards this, we employ *Synthetic* datasets, and change i in the range [5%, 80%]. Figure 8 plots the performance of VP and *k*ISB algorithms. Obviously, *k*ISB is better than VP in all cases. Note that, the number of comparisons and CPU time for the algorithms drop as i grows, especially for VP. This is because the comparable objects significantly decrease with the growth of incomplete rate.

Finally, we study the effect of the rate of bucket number r on the efficiency of our algorithms. Similarly, we use *Synthetic* datasets, and vary r from 5% to 80%. Figure 9 illustrates the cost of the algorithms as a function of r. Clearly, *k*ISB outperforms VP in all cases. Note that, the number of comparisons is not very sensitive to r, since the final query result is invariable no matter how large the r is. In addition, the CPU time of VP ascends gradually as r grows. The reason behind is that, the number of virtual points increases with r.

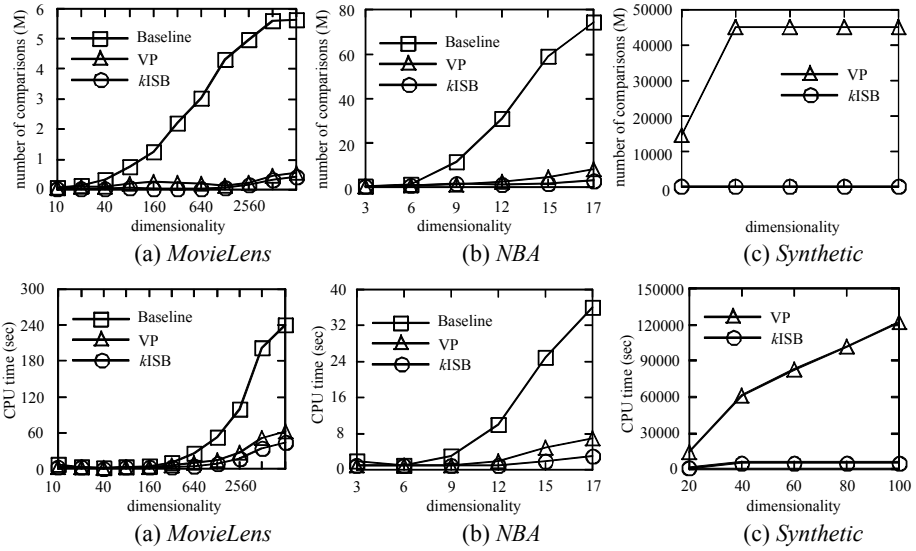


Fig. 7. kSB cost on incomplete data vs. *dim*

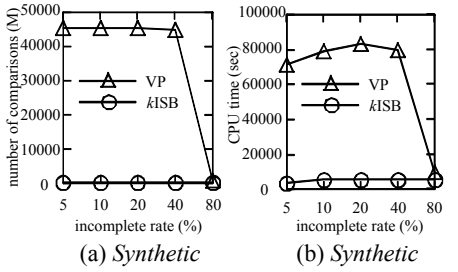


Fig. 8. kSB cost on incomplete data vs. *i*

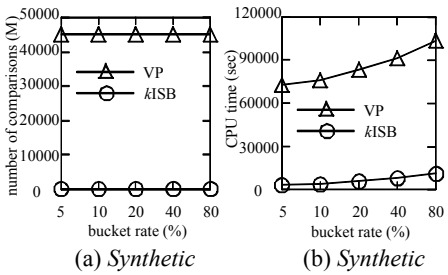


Fig. 9. *k*SB cost on incomplete data vs. *r*

6 Conclusions

This paper, for the first time, studies the problem of efficient *k-Skyband* (*k*SB) *query processing on incomplete data*, where multi-dimensional data items are missing some values of their dimensions. This problem is not only *interesting* from a research point of view, but also *useful* in many real applications such as decision making and location-based services. To efficiently answer the *k*SB query over incomplete data, we propose three algorithms, namely, Baseline, VP, and *k*ISB, which employ a series of novel concepts (e.g., ES, SS, TW, etc.) to improve the search performance. Extensive experimental evaluation using both real and synthetic datasets demonstrates that *k*ISB outperforms significantly the other algorithms in all cases. In the future, we intend to further improve the efficiency of our algorithms. Also, we plan to extend our methods to tackle other queries (e.g., Top-*k* Skyline query) over incomplete data.

Acknowledgements. This work was supported in part by NSFC 61003049, the Natural Science Foundation of Zhejiang Province of China under Grant LY12F02047, the Fundamental Research Funds for the Central Universities under Grant 2012QNA5018, the Key Project of Zhejiang University Excellent Young Teacher Fund (Zijin Plan), and the Bureau of Jiaxing City Science and Technology Project under Grant 2011AY1005.

References

1. Antova, L., Koch, C., Olteanu, D.: From Complete to Incomplete Information and Back. In: SIGMOD, pp. 713–724 (2007)
2. Bartolini, I., Ciaccia, P., Patella, M.: SaLSa: Computing the Skyline without Scanning the Whole Sky. In: CIKM, pp. 405–414 (2006)
3. Borzsonyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. In: ICDE, pp. 421–430 (2001)
4. Canahuate, G., Gibas, M., Ferhatosmanoglu, H.: Indexing Incomplete Databases. 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. 884–901. Springer, Heidelberg (2006)
5. Chen, L., Lian, X.: Efficient Processing of Metric Skyline Queries. IEEE Trans. Knowl. Data Eng. 21(3), 351–365 (2009)
6. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with Presorting. In: ICDE, pp. 717–719 (2003)
7. Dellis, E., Seeger, B.: Efficient Computation of Reverse Skyline Queries. In: VLDB, pp. 291–302 (2007)
8. Fuhry, D., Jin, R., Zhang, D.: Efficient Skyline Computation in Metirc Space. In: EDBT, pp. 1042–1051 (2009)
9. Haghani, P., Michel, S., Aberer, K.: Evaluating Top-*k* Queries over Incomplete Data Streams. In: CIKM, pp. 877–886 (2009)
10. Imielinski, T., Lipski, W.: Incomplete Information in Relational Databases. J. ACM 31(4), 761–791 (1984)
11. Khalefa, M.E., Mokbel, M.F., Levandoski, J.J.: Skyline Query Processing for Incomplete Data. In: ICDE, pp. 556–565 (2008)
12. Kossmann, D., Ramsak, F., Rost, S.: Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. In: VLDB, pp. 275–286 (2002)
13. Lian, X., Chen, L.: Monochromatic and Bichromatic Reverse Skyline Search over Uncertain Databases. In: SIGMOD, pp. 213–226 (2008)
14. Liu, Q., Gao, Y., Chen, G., Li, Q., Jiang, T.: On Efficient Reverse *k*-Skyband Query Processing. In: Lee, S.-G., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part I. LNCS, vol. 7238, pp. 544–559. Springer, Heidelberg (2012)
15. Ooi, B.C., Goh, C.H., Tan, K.-L.: Fast High-dimensional Data Search in Incomplete Databases. In: VLDB, pp. 357–367 (1998)
16. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive Skyline Computation in Database Systems. ACM Trans. Database Syst. 30(1), 41–82 (2005)
17. Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic Skylines on Uncertain Data. In: VLDB, pp. 15–26 (2007)
18. Pei, J., Yuan, Y., Lin, X., Jin, W., Ester, M., Liu, Q., Wang, W., Tao, Y., Yu, J.X., Zhang, Q.: Towards Multidimensional Subspace Skyline Analysis. ACM Trans. Database Systems 31(4), 1335–1381 (2006)

19. Qi, Y., Jain, R., Singh, S., Prabhakar, S.: Threshold Query Optimization for Uncertain Data. In: SIGMOD, pp. 315–326 (2010)
20. Soliman, M.A., Ilyas, I.F., David, S.-B.: Supporting Ranking Queries on Uncertain and Incomplete Data. In: VLDB, pp. 477–501 (2010)
21. Soliman, M.A., Ilyas, I.F., Martinenghi, D., Tagliasacchi, M.: Ranking with Uncertain Scoring Functions: Semantics and Sensitivity Measures. In: SIGMOD, pp. 805–816 (2011)
22. Zhang, W., Lin, X., Zhang, Y., Wang, W., Yu, J.: Probabilistic Skyline Operator over Sliding Window. In: ICDE, pp. 305–316 (2009)
23. Zhang, S., Mamoulis, N., Cheung, D.W.: Scalable Skyline Computation Using Object-based Space Partitioning. In: SIGMOD, pp. 483–494 (2009)

Mining Frequent Patterns from Uncertain Data with MapReduce for Big Data Analytics

Carson Kai-Sang Leung[1,⋆] and Yaroslav Hayduk[1,2]

[1] University of Manitoba, Winnipeg, MB, Canada
[2] Université de Neuchâtel, Neuchâtel, Switzerland
kleung@cs.umanitoba.ca

Abstract. Frequent pattern mining is commonly used in many real-life applications. Since its introduction, the mining of frequent patterns from *precise data* has drawn attention of many researchers. In recent years, more attention has been drawn on mining from *uncertain data*. Items in each transaction of these uncertain data are usually associated with existential probabilities, which express the likelihood of these items to be present in the transaction. When compared with mining from precise data, the search/solution space for mining from uncertain data is much larger due to presence of the existential probabilities. Moreover, we are living in the era of Big Data. In this paper, we propose a tree-based algorithm that uses MapReduce to mine frequent patterns from Big uncertain data. In addition, we also propose some enhancements to further improve its performance. Experimental results show the effectiveness of our algorithm and its enhancements in mining frequent patterns from uncertain data with MapReduce for Big Data analytics.

1 Introduction

Frequent pattern mining aims to discover implicit, previously unknown, and potentially useful knowledge—in the form of frequently occurring patterns—from large amounts of data. Since its introduction [2], the research problem of mining frequent patterns has drawn attention of many researchers. Over the past two decades, numerous methods have been proposed to mine and visualize frequent patterns [8, 20] as well as other related patterns [10, 14, 18, 24]. Examples of these methods include the classical Apriori algorithm [3] and the tree-based FP-growth algorithm [9]. Both algorithms mine frequent patterns from transaction databases of *precise* data.

However, there are situations in which users are uncertain about the presence or absence of some items or events [13, 15]. For example, a physician may highly suspect (but cannot guarantee) that a patient suffers from some specific diseases. The uncertainty of such suspicion can be expressed in terms of *existential probability*. As a concrete example, a physician may suspect that a patient has (i) a 75% likelihood of suffering from a flu and (ii) a $33\frac{1}{3}$% likelihood of suffering from

⋆ Corresponding author.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 440–455, 2013.

a cold (regardless of having or not having the flu). Here, in this uncertain dataset of patient records, each transaction represents a patient's visit to a physician's office. Note that a patient may suffer from multiple diseases at the same time (i.e., multiple items may appear together in the same transaction). Each item (representing a potential disease) in the transaction is associated with an existential probability expressing the likelihood of a patient having that disease in that visit. With this notion, each item in a transaction in traditional databases of precise data can be viewed as an item with a 100% likelihood of being present in the transaction.

Other examples of uncertain data include datasets of satellite images, where each item in a transaction expresses the likelihood of the presence of an object captured in an image. As the third example, each transaction in a dataset for an election expresses the likelihood of a collection of candidates chosen by (the secret ballot of) a voter. These are just a few examples of many real-life situations in which data are uncertain. Hence, efficient algorithms for mining uncertain data are in demand. Over the past few years, a few algorithms [1, 15–17] have been proposed to mine frequent patterns *serially* from static datasets of uncertain data. Note that the presence of existential probabilities in these uncertain datasets leads to a huge number of possible worlds [13] when using probabilistic-based mining of frequent patterns. In other words, the search space for frequent pattern mining from *uncertain data* can be much larger than that from *precise data*.

The situation has been worsen as we have moved into the era of *Big Data* [23]. When mining from vast amounts of Big Data, more efficient approaches (besides *serial* approach) are needed. To handle Big Data, some researchers proposed the use of *MapReduce*, which mines the search space with distributed or parallel computing. However, earlier works on MapReduce focused on data processing [7] or data mining tasks other than frequent pattern mining (e.g., clustering [5], outlier detection [11], structure mining [27]). Although two recent works [21, 25] were proposed to mine frequent patterns, both of them mine *precise* data (instead of *uncertain* data).

Hence, some natural questions to ask are: Can we use MapReduce to mine *uncertain* data? Can we use MapReduce to perform tree-based mining of uncertain data? How can we further speed up the mining process? In response to these questions, we propose a tree-based algorithm called **MR-growth**, which uses **M**ap**R**educe to mine frequent patterns from uncertain data in a pattern-**growth** fashion for Big Data analytics. Moreover, our additional *key contributions* of this paper include the three enhancements to the MR-growth algorithm.

This paper is organized as follows. The next section gives background and related work. In Section 3, we propose our MR-growth algorithm for mining frequent patterns from uncertain data using MapReduce. Section 4 discusses three enhancements to our MR-growth algorithm. Evaluation results and conclusions are presented in Sections 5 and 6, respectively.

2 Background and Related Work

In this section, we provide background on frequent pattern mining from uncertain data and on MapReduce. We also discuss some related works.

2.1 Mining Frequent Patterns from Uncertain Data

When using probabilistic-based mining [6, 15, 19] with the "possible world" interpretation [13], a pattern is considered *frequent* if its expected support is no less than the user-specified *minsup* threshold. When items within a pattern X are independent, the *expected support* of X in the database DB can be computed by summing (over all transactions $t_1, \ldots, t_{|DB|}$) the product (of existential probabilities within X):

$$expSup(X) = \sum_{i=1}^{|DB|} \left(\prod_{x \in X} P(x, t_i) \right), \tag{1}$$

where $P(x, t_i)$ is an existential probability of item x in transaction t_i. With this definition of expected support, the existing tree-based UF-growth algorithm mines frequent pattern from uncertain data as follows. The algorithm first scans the dataset once to compute the expected support of all domain items (i.e., singleton itemsets). Infrequent items are pruned as their extensions/supersets are guaranteed to be infrequent. The algorithm then scans the dataset the second time to insert all transactions (with only frequent items) into an UF-tree. Each node in the UF-tree captures (i) an item x, (ii) its existential probability $P(x, t_i)$, and (iii) its occurrence count. At each step during the mining process, the frequent patterns are expanded recursively.

2.2 The MapReduce Programming Model

MapReduce [7] is a high-level programming model for processing vast amounts of data. Usually, MapReduce uses parallel and distributed computing on clusters or grids of nodes (i.e., computers). The ideas behind MapReduce can be described as follows. As implied by its name, MapReduce involves two key functions: "map" and "reduce". The input data are read, divided into several partitions (subproblems), and assigned to different processors. Each processor executes the *map function* on each partition (subproblem). The map function takes a pair of $\langle key, value \rangle$ data and returns a list of $\langle key, value \rangle$ pairs as an intermediate result:

$$\text{map: } \langle key_1, value_1 \rangle \mapsto \text{list of } \langle key_2, value_2 \rangle,$$

where (i) key_1 & key_2 are keys in the same or different domains, and (ii) $value_1$ & $value_2$ are the corresponding values in some domains. Afterwards, these pairs are shuffled and sorted. Each processor then executes the *reduce function* on (i) a single key value from this intermediate result together with (ii) the list of all values that appear with this key in the intermediate result. The reduce function

"reduces"—by combining, aggregating, summarizing, filtering, or transforming—the list of values associated with a given key (for all k keys) and returns a list of k values:

$$\text{reduce: } \langle key_2, \text{list of } value_2 \rangle \mapsto \text{list of } value_3,$$

or returns a single (aggregated or summarized) value:

$$\text{reduce: } \langle key_2, \text{list of } value_2 \rangle \mapsto value_3,$$

where (i) key_2 is a key in some domains, and (ii) $value_2$ & $value_3$ are the corresponding values in some domains. Examples of MapReduce applications include the construction of an inverted index as well as the word counting of a document.

2.3 Related Work

Earlier works on MapReduce focused either on data processing [7] or on some data mining tasks other than frequent pattern mining (e.g., outlier detection [11], structure mining [27]). Recently, Lin et al. [21] proposed three Apriori-based algorithms called SPC, FPC and DPC to mine frequent patterns from *precise data*. Among them, SPC uses single-pass count to find frequent k-itemsets at the k-th pass of the database scan (for $k \geq 1$). FPC uses fixed-passes combined-counting to find all k-, $(k+1)$-, ..., $(k+m)$-itemsets in the same pass of database scan. On the one hand, this fix-passes technique fixes the number of required passes from K (where K is the maximum cardinality of all frequent itemsets that can be mined from the precise data) to a user-specified constant. On the other hand, due to combined-counting, the number of generated candidates is higher than that of SPC. In contrast, DPC uses dynamic-passes combined-counting, which takes the benefits of both SPC and FPC by taking into account the workloads of nodes when mining frequent itemsets with MapReduce. Like these three algorithms, our proposed MR-growth algorithm also uses MapReduce. However, unlike these three algorithms (which mine frequent itemsets from *precise data* using the *Apriori-based approach*), our proposed MR-growth algorithm mine frequent itemsets from *uncertain data* using a *tree-based approach*. Note that the search/solution space for frequent pattern mining from uncertain data is much larger than frequent pattern mining from precise data due to presence of the existential probabilities.

Riondato et al. [25] proposed a parallel randomized algorithm called PARMA for mining *approximations* to the top-k frequent itemsets and association rules from *precise data* using MapReduce. Although PARMA and our MR-growth algorithm both use MapReduce, one key difference between the two algorithms is that we aim to mine *truly frequent* (instead of approximately frequent) itemsets. Another key difference is that we mine from *uncertain data* (instead of precise data).

3 Our MR-Growth Algorithm for Mining Frequent Patterns from Uncertain Data with MapReduce

In this section, we propose our MR-growth algorithm, which uses MapReduce to mine frequent patterns from huge amounts of uncertain data in a tree-based pattern-growth fashion. The algorithm can be divided into multiple stages.

First, MR-growth reads a huge dataset of uncertain data. As each item in the dataset is associated with an existential probability, MR-growth aims to compute the expected support of all domain items (i.e., singleton itemsets) by using MapReduce. The expected support of any itemset can be computed by using Equation (1). Moreover, when computing singleton itemsets, such an equation can be simplified to become the following:

$$expSup(\{x\}) = \sum_{i=1}^{|DB|} P(x, t_i), \tag{2}$$

where $P(x, t_i)$ is an existential probability of item x in transaction t_i. Specifically, MR-growth divides the uncertain dataset into several partitions and assigns them to different processors. During the *mapping phase* of this stage, the mapper function receives ⟨transaction ID, content of that transaction⟩ as input. For every transaction t_i, the mapper function emits a $\langle key, value \rangle$ pair for each item $x \in t_i$.

What should be the emitted pair? A naive attempt is to emit $\langle x, 1 \rangle$ for each occurrence of $x \in t_i$. It would work well when mining *precise* data because each occurrence of x leads to an actual support of 1. In other words, occurrence of x is the same as the actual support of x when mining precise data. However, this is *not* the case when mining *uncertain* data. The occurrence of x can be different from the expected support of x when mining uncertain data. For instance, consider an item a with existential probability of 0.9 that appears only in transaction t_1. Its expected support may be higher than item b that appears seven times but with an existential probability of 0.1 in each appearance. Then, $expSup(\{a\}) = 0.9 > 0.7 = expSup(\{b\})$. Hence, instead of emitting $\langle x, 1 \rangle$ for each occurrence of $x \in t_i$, MR-growth emits $\langle x, P(x, t_i) \rangle$ for each occurrence of $x \in t_i$. In other words, the *mapper* function can be specified as follows:

For each transaction $t_i \in$ partition of the uncertain dataset **do**
  for each item $x \in t_i$ **do**
    emit $\langle x, P(x, t_i) \rangle$.

This results in a list of $\langle x, P(x, t_i) \rangle$ pairs for many different x and $P(x, t_i)$. Afterwards, these pairs are shuffled and sorted. Each processor then executes the reduce function on the shuffled and sorted pairs to obtain the expected support of x. In other words, the *reducer* function can be specified as follows:

Set $expSup(x) = 0$;
For each $x \in \langle x,$ list of $P(x, t_i) \rangle$ **do**
  for each $P(x, t_i) \in \langle x,$ list of $P(x, t_i) \rangle$ **do**
    $expSup(x) = expSup(x) + P(x, t_i)$.
  emit $\langle x, expSup(x) \rangle$.

Table 1. A sample transaction dataset of uncertain data

TID	Itemsets
t_1	{a:0.5, b:0.5, c:1.0, d:1.0, u:0.5}
t_2	{a:0.5, b:0.5, p:0.5}

Example 1. Let us consider an uncertain dataset as shown in Table 1 with *minsup*=1.0 For the first transaction t_1, the mapper function outputs $\langle a, 0.5\rangle$, $\langle b, 0.5\rangle$, $\langle c, 1.0\rangle$, $\langle d, 1.0\rangle$, $\langle u, 0.5\rangle$. Similarly, for the second transaction t_2, the mapper function outputs $\langle a, 0.5\rangle, \langle b, 0.5\rangle, \langle p, 0.5\rangle$. These pairs are then shuffled and sorted. Afterwards, the reducer reads $\langle a, [0.5, 0.5]\rangle$, $\langle b, [0.5, 0.5]\rangle$, $\langle c, [1.0]\rangle$, $\langle d, [1.0]\rangle, \langle p, [0.5]\rangle, \langle u, [0.5]\rangle$ and outputs $\langle a, 1.0\rangle, \langle b, 1.0\rangle, \langle c, 1.0\rangle, \langle d, 1.0\rangle, \langle p, 0.5\rangle$, $\langle u, 0.5\rangle$ (i.e., items and their corresponding expected support). □

Next, MR-growth reads the singleton items and their associated existential supports, and prunes the infrequent items. Then, it splits the list containing frequent singletons into distinct groups and assigns a unique ID to each group. The new list containing group-to-singleton mappings is called a group list (G-list). To summarize, this stage identifies which conditional trees should be mined together on one computing node.

Example 2. Let us continue with our example. At this stage, MR-growth prunes items u and p because their existential support equals to $0.5 < 1.0$=*minsup*. Given that we want to split the remaining (frequent) items a, b, c and d into two groups (the number of items which are mapped to a given group can be determined automatically depending on the number of computing nodes), this stage yields $\langle Group_1\text{: } a, b\rangle$ and $\langle Group_2\text{: } c, d\rangle$. □

The next stage is an important and computationally intensive stage. Here, MR-growth identifies all group-dependent transactions. First, on each machine executing the mapper functions, MR-growth loads the G-list into main memory, and creates a reverse map that maps singletons to their corresponding group ID. Then, each mapper receives $\langle key = groupID, value = DB(groupID)\rangle$ as input. For every transaction t_i in $DB(groupID)$, MR-growth substitutes all transaction items with their corresponding group IDs from the reverse map, creating a new list I of the same size as the transaction size. For each groupID in I, MR-growth locates the rightmost appearance L of $groupID$ in I, and emits a new (truncated) transaction, in the form of $\langle key' = groupID, value' = t_i[1]t_i[2]\dots t_i[L]\rangle$.

Afterwards, MR-growth receives group-dependent transactions in the form of $\langle key = groupID, value = \{t_1 \dots t_n\}\rangle$ and inserts them into a tree, creating a compressed tree-based representation of these group-dependent transactions. MR-growth then collects the group-dependent UF-trees and merges them into one single tree, from which frequent patterns can be mined.

Example 3. Let us continue with our example. After replacing the transaction items with their corresponding group IDs, we get the two lists as summarized in Table 2.

Table 2. Group lists

Group List	groupIDs
1	⟨1, 1, 2, 2⟩
2	⟨1, 1⟩

For the first transaction t_1, the rightmost appearance of groupID 1 is 2 (indicating the appearance of a, b ends in the 2nd position of transaction t_1). So, ⟨1, [a:0.5, b:0.5]⟩ is emitted. Similarly, the rightmost appearance of groupID 2 is 4 (indicating the appearance of c, d ends in the 4th position of transaction t_1). So, ⟨2, [a:0.5, b:0.5, c:1.0, d:1.0]⟩ is emitted. In a similar fashion, for the second transaction t_2, the rightmost appearance of groupID 1 is 2 (indicating the appearance of a, b ends in the 2nd position of transaction t_2). So, ⟨1, [a:0.5, b:0.5]⟩ is emitted.

MR-growth then merges both group-dependent transactions in the UF-tree. Notice that we can merge the two received group-dependent transactions into one branch of the UF-tree for $Group_1$: (a:0.5):2 and (b:0.5):2 when using the (*item*:*probability*):*count* notation. For $Group_2$, MR-growth receives ⟨2, [a:0.5, b:0.5, c:1.0, d:1.0]⟩ as input and inserts the single group-dependent transaction into a new UF-tree. To summarize, the algorithm emits ⟨1, [(a:0.5):2, (b:0.5):2]⟩ and ⟨2, [(a:0.5):1,(b:0.5):1,(c:1.0):1,(d:1.0):1]⟩.

Afterwards, for $Group_1$, the reducer function receives one UF-tree in the ⟨1, [(a:0.5):2,(b:0.5):2]⟩ format, mines patterns having items a and b from that tree, but it does not discover any *frequent* patterns. Similarly, for $Group_2$, the reducer function receives ⟨2, [(a:0.5):1, (b:0.5):1, (c:1.0):1, (d:1.0):1]⟩ as input, mines that tree for patterns having items c and d, and emits ⟨$\{c, d\}$:1.0⟩ as the only frequent pattern. □

4 Enhancements to MR-Growth

While our proposed MR-growth algorithm efficiently mines frequent patterns from uncertain data, we propose three enhancements in this section to further speed up the mining process.

4.1 Enhancement #1: Multi-core Processors in the ForkJoin Framework

To increase the mining speed, we exploit machines having multi-core processors by using the *ForkJoin framework* [12]. The main goal of the ForkJoin framework is to split computationally intensive tasks into multiple pieces, which can then be performed in parallel, to minimize the execution time of the algorithm. Unlike MapReduce (in which the developer does *not* need to explicitly control the work distribution process), ForkJoin requires the developer to explicitly control the work distribution process. In the Java programming language, the ForkJoin

framework also uses the concept of *work stealing*, where the thread (which completes the work assigned to it) can *steal* tasks from other threads and assign the tasks to idle threads as efficiently as possible.

Each thread maintains a task queue and repeatedly takes the next available task from the head of its queue until the task queue becomes empty. Each time when a thread does not have any pending work to complete (i.e., its queue is empty), the thread becomes a *thief*. It selects a different thread at random, and tries to *steal* a task from the tail of the queue of the chosen thread. Once the task is completed, this process is repeated (i.e., steal a task from the tail of the queue of some random thread, which can be the previously selected one).

To summarize, we enhance our MR-growth algorithm by taking the following steps:

1. detect the number N_{cores} of processing cores on a multi-core processor;
2. divide the list of group-dependent items G_i into approximately equal parts such that each thread (running on a different processing core) is responsible for mining $\frac{|G_i|}{N_{cores}}$ items, and insert these items into the queue of each thread;
3. when any given thread finishes constructing and mining conditional trees for all its assigned singletons, the algorithm attempts to steal a singleton from the queue of a random thread.

This enhancement is particularly beneficial in MapReduce infrastructures having a limited number of computing nodes. Each group ID is mapped to many singleton items. Hence, each computing node is responsible for the construction and the mining of conditional trees for a number of domain items (i.e., singleton itemsets).

4.2 Enhancement #2: Efficient Conditional Tree Construction

In this section, we discuss the next enhancement, which allows us to construct conditional trees without constructing projected trees first. Recall that to construct a conditional tree, the MR-growth algorithm first constructs a projected tree and then prunes the locally-infrequent nodes from it to create a conditional tree.

Our enhancement for conditional tree construction to our proposed MR-growth algorithm can be described as follows. The conditional tree is constructed using two traversals of the main tree. The first bottom-up traversal accumulates the counts of all encountered items on the path, flagging all visited nodes. Then, by traversing the same path again but *top-down*—which can be accomplished by recursively visiting only the flagged child nodes, the second scan traverses the main tree in a depth-first manner to build a conditional tree. Specifically, the algorithm performs the following steps:

1. For each item x in the header table, traverse the tree bottom-up and count the occurrence of each encountered item on the tree path;
2. If the parent node of the current node has more than one child, flag the current node with item x (i.e., `childNode.flag`=x);

Table 3. A sample transaction database of precise data

TID	Itemsets	(Ordered) Frequent Itemsets
t_1	$\{a, b, c, d, u\}$	$\{a, b, c, d\}$
t_2	$\{a, b, p\}$	$\{a, b\}$
t_3	$\{a, j, b, c, i, d\}$	$\{a, b, c, d\}$
t_4	$\{g, a, n, b\}$	$\{a, b\}$
t_5	$\{l, a, b, m, c\}$	$\{a, b, c\}$
t_6	$\{a, c, q, t, u, g, w, d\}$	$\{a, c, q, t, u, w, d\}$
t_7	$\{h, a, q, t, u, w\}$	$\{a, q, t, u, w\}$
t_8	$\{b, c, q, u, t, w, k\}$	$\{b, c, q, t, u, w\}$

3. Determine which items are locally frequent, and insert them into the new header table, which will be associated with the conditional tree;
4. Traverse the tree again in a top-down fashion. When a node with multiple children is encountered, visit each of the children and flag it with item x;
5. For each visited node, check if it is frequent. Add the frequent nodes to the new conditional tree; and
6. Stop traversing the current branch if all children of the current node are guaranteed to be infrequent.

Example 4. Let us consider the dataset shown in Table 3. Without loss of generality, when building the $\{d\}$-conditional tree, MR-growth with Enhancement #2 first traverses each $\{d\}$-link (circled nodes denote $\{d\}$-link nodes) bottom-up. Then, it accumulates the count of the encountered items in the header table. For nodes having multiple children, it flags each child node with $\{d\}$. Fig. 1 demonstrates the process of building a $\{d\}$-conditional tree. Nodes visited during the first traversal are surrounded by squares, and nodes visited during the second traversal are bolded. During the second traversal, we can stop the traversal of both branches early (e.g., after visiting a node with item c) because, thanks to the information collected during the first traversal of the tree, we know that any items after item c in *any* path are guaranteed to be infrequent.

Using the tree in Fig. 1, let us compute the amount of allocated memory and calculate the number of visited node required by the original version of MR-growth vs. the version enhanced by this efficient conditional tree construction. During the first bottom-up traversal, MR-growth (w/ Enhancement #2) visited 9 nodes in the main tree and did not allocate any new nodes. Then, MR-growth (w/ Enhancement #2) traversed the tree once again in the top-down fashion. It visited 4 nodes in the main tree and allocated 2 nodes for the new conditional tree. To summarize, MR-growth (w/ Enhancement #2) traversed 13 nodes and allocated 2 new nodes; it did not perform any memory deallocations. In contrast, the original version of MR-growth visited 9 nodes (in the main tree) + 8 nodes (in the projected tree) = 17 nodes as well as allocated 9 new nodes for the projected tree, 7 of which needed to be deallocated in the conditional tree. □

As observed from the above example, the benefits of employing Enhancement #2 include the following:

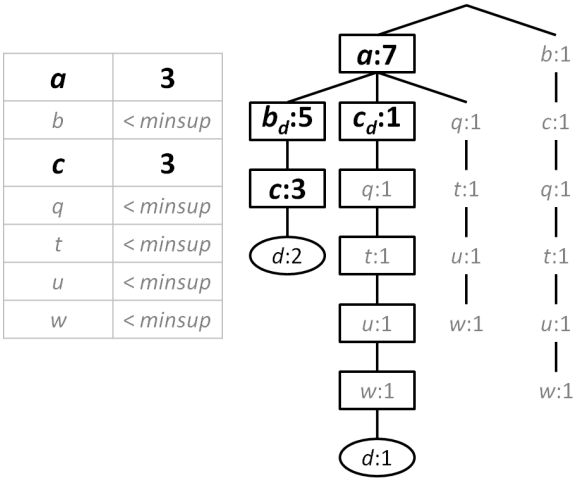


Fig. 1. The efficient construction of a $\{p\}$-conditional tree

1. its efficient tree node allocation, which avoids the need of (i) allocating memory for infrequent nodes in a projected tree and (ii) freeing it when pruning a projected tree or a conditional tree;
2. its efficient tree traversal, which visits all of the tree nodes only twice in the worst case.

4.3 Enhancement #3: Sampling

Sampling is a commonly used technique in many data mining tasks, especially when trying to find an *approximate* solution (instead of an accurate one). It was observed [26] that the UF-tree is usually bigger than the FP-tree because the former captures both items and their existential probabilities from the uncertain datasets whereas the latter captures only the items from the precise databases. As the tree gets bigger, it takes longer to build and traverse. Consequently, mining with UF-trees usually takes longer than mining with FP-trees.

The idea of our Enhancement #3 is that, instead of building a UF-tree during the mining process, we adopt the Concatenating Sample method [4]. For every $\langle x, P(x, t_i)\rangle$-pair in each transaction t_i, we generate a random real number r in the range (0,1]. If r is $\geq P(x, t_i)$, then we include x in the current sample. Otherwise (i.e., when $r < P(x, t_i)$, we omit x from the current sample. At the end of this sampling process, we obtain a "possible" world instance for the uncertain dataset. For example, $t_1 = \{a, b, d\}$ and $t_2 = \{b, p\}$ can be one such "possible" world instance. We can then mine frequent patterns from such an instance in the same way that we mine frequent patterns from precise data (e.g., using FP-growth [9]).

As one sample is subject to bias, we repeat the above process to obtain a few samples and mine frequent patterns from each of these samples. Given these

samples, we apply the enhanced version of MR-growth to mine frequent patterns. As we are dealing with "possible" world instances in the same way that we mine precise data, we need to modify the mapper and reducer function for this enhancement. For example, instead of letting mapper emit $\langle x, P(x, t_i)\rangle$, we modify the mapper to emit $\langle x, 1\rangle$.

As this mining process gives an approximate solution, a post-processing step (which requires one more—i.e., the third—scan of the dataset). So, we also need to modify the reducers to execute this post-processing step.

5 Experimental Results

In this section, we evaluate our proposed MR-growth algorithm and its enhancements. Experiments were run using either a single machine or the Amazon EC2 cluster. Specifically, some experiments were executed on a machine with an Intel Core i7 4-core processor (1.73 GHz) and 8 GB of main memory, running a 64-bit Windows 7 operation system. All versions of the algorithm were implemented in the Java programming language. The stock version of Apache Hadoop 0.20.0 was used. As for the datasets for experiments, we used those benchmarks (e.g., accidents, connect4 and mushroom) from the UCI Machine Learning Repository (`http://mlearn.ics.uci.edu/MLRepository.html`) and the FIMI repository (`http://fimi.ua.ac.be`). Some other experiments were run on the Amazon EC2 cluster—specifically, 11 `m2.xlarge` computing nodes (`http://aws.amazon.com/ec2`). As Calders et al. [4] suggested that two samples per transaction were sufficient to approximate (i.e., with less than 0.02% error) the expected supports of the mined patterns for most datasets, we also used two samples per transaction in the experiments.

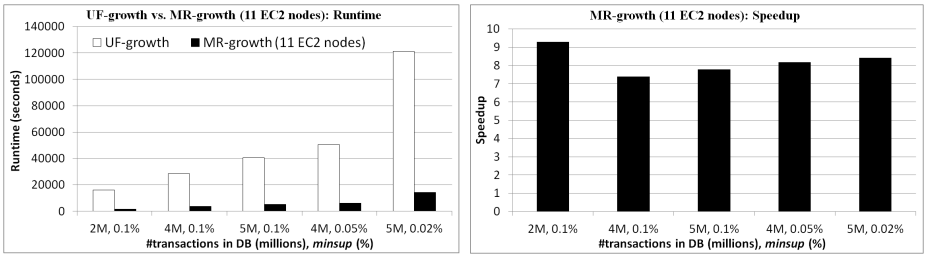
In addition to the above real-life benchmark datasets, we also generated three new synthetic datasets using the IBM Quest Dataset Generator [3] for our evaluations. The generated data ranges from 2M to 5M transactions with an average transaction length of 10 items from a domain of 1K items. As these datasets originally contained only precise data, we assigned to each item contained in every transaction an existential probability from the range (0,1].

5.1 Evaluation of MR-Growth

In this experiment, we executed our MR-growth algorithm in the MapReduce environment with 11 nodes. Fig. 2(a) shows that, while the sequential version of the UF-growth algorithm took more than 120,000 seconds to execute, its corresponding version required less than 20,000 seconds in the MapReduce environment.

Observed from Fig. 2(b), when the total execution time of the MR-growth algorithm was low, speedup of 7 to 8 times over its sequential version was achieved. When we increased the dataset size, the algorithm achieved a speedup of approximately 8.5 times on 11 nodes.

In terms of accuracy, as an exact algorithm, our MR-growth algorithm found the same sets of truly frequent patterns as those returned by UF-growth [15].



(a) Runtime (b) Speedup

Fig. 2. UF-growth vs. MR-growth

5.2 Evaluation of Enhancement #1: MR-Growth with ForkJoin

This experiment demonstrates the effect of employing multiple threads for mining frequent patterns with our MR-growth algorithm. As the tests were executed on a 4-core machine, we varied the number of threads from 1 to 4.

For the accidents dataset, Fig. 3(a) shows that, when the execution time was short, the algorithm took longer on multiple threads than on a single thread. The reason is that, as there is not enough work to do in the parallel, the algorithm cannot take advantage of multiple cores. When the algorithm executed for a longer period of time (e.g., more than 100 seconds), sub-linear speedup was achieved as shown in Fig. 3(b). Fig. 3 also shows the experimental results for other datasets (e.g., connect4 and mushroom).

As this enhancement aims to speed up the mining process, it does not change the accuracy of the mining results.

5.3 Evaluation of Enhancement #2: Efficient Conditional Tree Construction

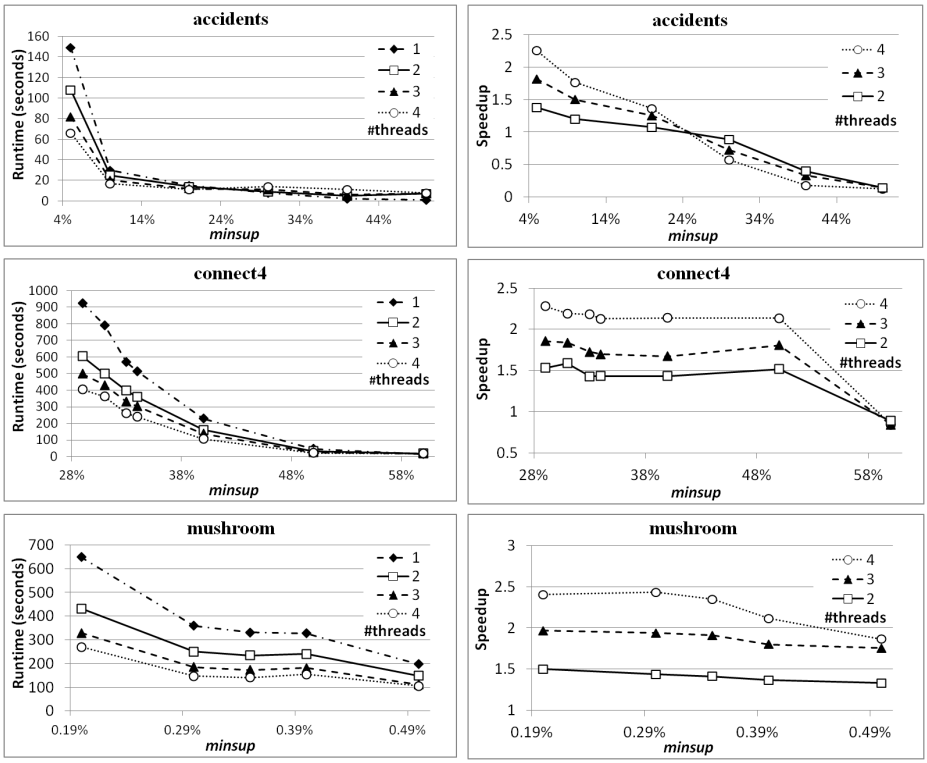
In this experiment, we compared the execution time of the original version of MR-growth with the version enhanced with the efficient conditional tree construction as discussed in Section 4.2.

Fig. 4 shows that, for small *minsup* values, both versions yielded the final result in less than 200 seconds for the accidents dataset. The performance differences became apparent only when *minsup* was lowered to 30%. As for the connect4 and mushroom datasets, the enhanced version of the MR-growth algorithm outperformed the original version. Fig. 4 also highlights that the enhanced MR-growth algorithm performed better (e.g., the difference were more than 300 seconds in some cases).

In terms of accuracy, frequent patterns mined by MR-growth with Enhancement #2 were identical to those mined by MR-growth without this enhancement.

5.4 Evaluation of Enhancement #3: MR-Growth with Sampling

We evaluated Enhancement #3 by comparing the execution times of the original version of MR-growth with the version enhanced with sampling. Fig. 5(a)



(a) Runtime (b) Speedup

Fig. 3. MR-growth with ForkJoin

illustrates that MR-growth (w/ sampling) consistently yielded the final result quicker than the original version of MR-growth.

Moreover, to test the capacity of the MR-growth algorithm to further offload work to different processor cores by using the ForkJoin framework (Enhancement #1), we compared the execution times of MR-growth with one thread on ForkJoin vs. MR-growth with two threads on ForkJoin. We observed from Fig. 5(b) that, while the execution time of MR-growth on two threads was lower, the overall benefits of distributing work to multiple threads was not too significant. This behaviour is expected because Amazon uses virtualization to expose *virtual processing cores* on shared hardware resources, which degrades potential speedup. Significant performance improvements could be observed in MapReduce environments built from high-performance machines [22].

As MR-growth with Enhancement #3 approximated expected supports, the mined frequent patterns were not identical to those mined by MR-growth without this enhancement. However, the differences were not too significant. In other words, the algorithm produced an acceptable approximation to truly frequent patterns.

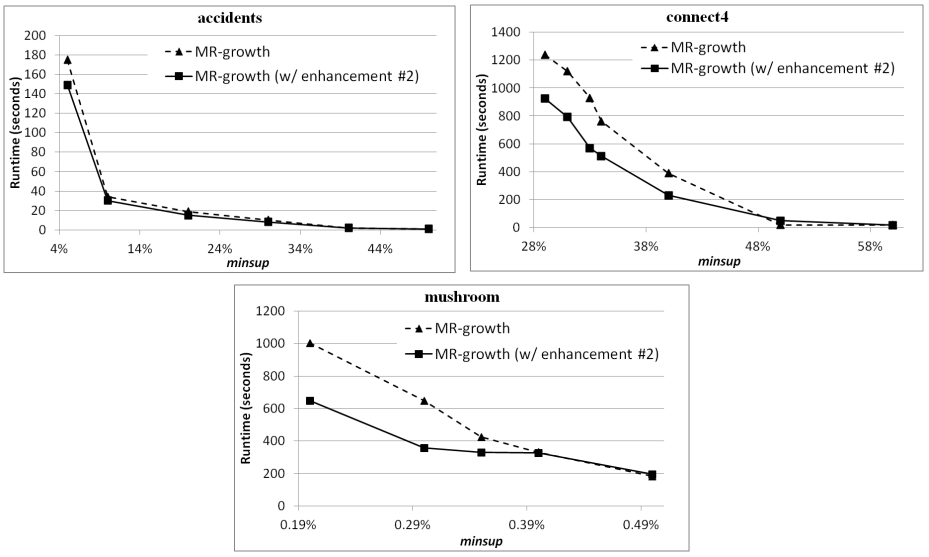
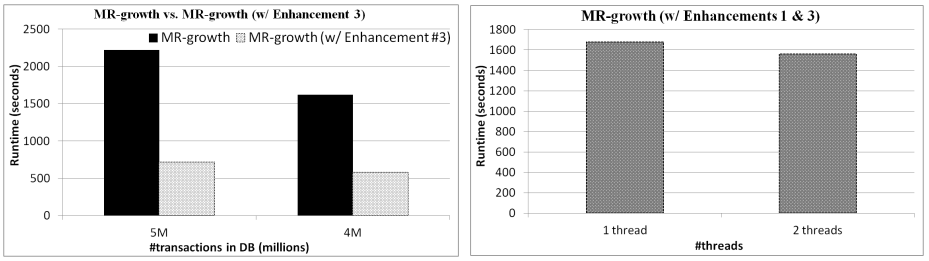


Fig. 4. MR-growth (without vs. with Efficient Conditional Tree Construction)



(a) MR-growth: With Enhancement #3 (b) With Enhancements #1 & #3

Fig. 5. MR-growth with Sampling (Enhancement #3) + 1 or 2 threads in ForkJoun (Enhancement #1)

6 Conclusions

There are many real-life situations in which we observe uncertain data (e.g., in temperature and wind speed readings, patient diagnosis, and satellite imaging). Given the probabilistic nature of these data, it may take a long time and more resources to mine frequent patterns from uncertain data. Currently, many state-of-the-art algorithms for mining frequent patterns from uncertain data may not provide superb performance because most of them are not crafted to execute in parallel. In this paper, we introduced our MR-growth algorithm, which provides the possibility to construct and mine smaller-sized UF-trees on distributed machines. Our experimental results demonstrate the effectiveness of employing the

MapReduce programming model for mining frequent patterns from uncertain data for Big data analytics. Moreover, the use of MapReduce yields significant speedups to our MR-growth algorithm. As for the use of the ForkJoin framework, it is beneficial for networks where computing nodes contain multi-core processors. As ongoing and future work, we plan to conduct more extensive experiments (e.g., evaluate the effect of the number of nodes in the MapReduce environment on the runtime). We also target at finding a framework, possibly an extension of MapReduce, which would allow us to recursively build sub-trees and schedule their mining on available computation resources.

Acknowledgements. This project is partially supported by NSERC (Canada) and University of Manitoba.

References

1. Aggarwal, C.C., Li, Y., Wang, J., Wang, J.: Frequent pattern mining with uncertain data. In: ACM KDD 2009, pp. 29–38 (2009)
2. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: ACM SIGMOD 1993, pp. 207–216 (1993)
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB 1994, pp. 487–499 (1994)
4. Calders, T., Garboni, C., Goethals, B.: Efficient pattern mining of uncertain data with sampling. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part I. LNCS (LNAI), vol. 6118, pp. 480–487. Springer, Heidelberg (2010)
5. Cordeiro, R.L.F., Traina Jr., C., Traina, A.J.M., López, J., Kang, U., Faloutsos, C.: Clustering very large multi-dimensional datasets with MapReduce. In: ACM KDD 2011, pp. 690–698 (2011)
6. Chui, C.-K., Kao, B., Hung, E.: Mining frequent itemsets from uncertain data. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 47–58. Springer, Heidelberg (2007)
7. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. CACM 51(1), 107–113 (2008)
8. Eavis, T., Zheng, X.: Multi-level frequent pattern mining. In: Zhou, X., Yokota, H., Deng, K., Liu, Q. (eds.) DASFAA 2009. LNCS, vol. 5463, pp. 369–383. Springer, Heidelberg (2009)
9. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD 2000, pp. 1–12 (2000)
10. Kiran, R.U., Reddy, P.K.: An alternative interestingness measure for mining periodic-frequent patterns. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part I. LNCS, vol. 6587, pp. 183–192. Springer, Heidelberg (2011)
11. Koufakou, A., Secretan, J., Reeder, J., Cardona, K., Georgiopoulos, M.: Fast parallel outlier detection for categorical datasets using MapReduce. In: IEEE IJCNN 2008, pp. 3298–3304 (2008)
12. Lea, D.: A Java fork/join framework. In: ACM Java 2000, pp. 36–43 (2000)
13. Leung, C.K.-S.: Mining uncertain data. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1(4), 316–329 (2011)
14. Leung, C.K.-S., Jiang, F., Sun, L., Wang, Y.: A constrained frequent pattern mining system for handling aggregate constraints. In: IDEAS 2012, pp. 14–23 (2012)

15. Leung, C.K.-S., Mateo, M.A.F., Brajczuk, D.A.: A tree-based approach for frequent pattern mining from uncertain data. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 653–661. Springer, Heidelberg (2008)
16. Leung, C.K.-S., Sun, L.: Equivalence class transformation based mining of frequent itemsets from uncertain data. In: ACM SAC 2011, pp. 983–984 (2011)
17. Leung, C.K.-S., Tanbeer, S.K.: Fast tree-based mining of frequent itemsets from uncertain data. In: Lee, S.-g., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part I. LNCS, vol. 7238, pp. 272–287. Springer, Heidelberg (2012)
18. Leung, C.K.-S., Tanbeer, S.K.: Mining popular patterns from transactional databases. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 291–302. Springer, Heidelberg (2012)
19. Leung, C.K.-S., Tanbeer, S.K., Budhia, B.P., Zacharias, L.C.: Mining probabilistic datasets vertically. In: IDEAS 2012, pp. 99–204 (2012)
20. Leung, C.K.-S., Jiang, F.: RadialViz: An orientation-free frequent pattern visualizer. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part II. LNCS (LNAI), vol. 7302, pp. 322–334. Springer, Heidelberg (2012)
21. Lin, M.-Y., Lee, P.-Y., Hsueh, S.-C.: Apriori-based frequent itemset mining algorithms on MapReduce. In: ICUIMC 2012, art. 76 (2012)
22. Lloyd, W., Shrideep, P., Olaf, D., Lyon, J., Mazdak, A., Ken, R.: Migration of multi-tier applications to infrastructure-as-a-service clouds: an investigation using Kernel-based virtual machines. In: IEEE/ACM GRID 2011, pp. 137–144 (2011)
23. Madden, S.: From databases to big data. IEEE Internet Computing 16(3), 4–6 (2012)
24. Rashid, M. M., Karim, M. R., Jeong, B.-S., Choi, H.-J.: Efficient mining regularly frequent patterns in transactional databases. In: Lee, S.-g., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part I. LNCS, vol. 7238, pp. 258–271. Springer, Heidelberg (2012)
25. Riondato, M., DeBrabant, J., Fonseca, R., Upfal, E.: PARMA: a parallel randomized algorithm for approximate association rules mining in MapReduce. In: ACM CIKM 2012, pp. 85–94 (2012)
26. Tong, Y., Chen, L., Cheng, Y., Yu, P.S.: Mining frequent itemsets over uncertain databases. In: PVLDB, vol. 5(11), pp. 1650–1661 (2012)
27. Yang, S., Wang, B., Zhao, H., Wu, B.: Efficient dense structure mining using MapReduce. In: IEEE ICDM Workshops 2009, pp. 332–337 (2009)

Efficient Probabilistic Reverse k-Nearest Neighbors Query Processing on Uncertain Data

Jiajia Li, Botao Wang, and Guoren Wang

College of Information Science & Engineering, Northeastern University, P.R. China
jiajia4487@gmail.com, {wangbotao,wangguoren}@ise.neu.edu.cn

Abstract. A reverse k-nearest neighbors ($RkNN$) query returns all the objects that take the query object q as their k nearest neighbors. However, data are often uncertain in numerous applications. In this paper, we focus on the problem of processing $RkNN$ on uncertain data. A probabilistic $RkNN$ ($PRkNN$) query retrieves all the objects that have higher probabilities than a user-specified threshold to be the $RkNN$ of q. The previous work for answering $PRNN$ query are mainly based on the distance relationship between uncertain objects, and are inapplicable for $PRkNN$ when $k > 1$. In this paper, we design a novel algorithm for $PRkNN$ query to support arbitrary values of k on the basis of two pruning strategies, namely spatial pruning and probabilistic pruning. The spatial pruning rule is defined on both the distances and the angle ranges between uncertain objects. And an efficient upper bound of probability is estimated by the probabilistic pruning algorithm. Extensive experiments are conducted to study the performance of the proposed approach. The results show that our proposed algorithm has a better performance and scalability than the existing solution regarding the growth of k.

1 Introduction

A reverse k-nearest neighbors ($RkNN$) query returns all the objects taking the query object as their k nearest neighbors. This type of query is very practical and important because of its applications in decision support system, profile-based marketing and maintaining document repositories, and received plenty of research interests in recent years [1–3]. However, data are often uncertain in numerous applications due to limitations of measuring equipment, delayed data updates, or privacy protection.

In this paper, we investigate the problem of probabilistic $RkNN$ ($PRkNN$) query which is to find the probable $RkNN$s with probability higher than a user-specified threshold. $PRkNN$ query has many important applications. For example, a restaurant may want to find the residents that consider it as their k-nearest neighbors. For privacy reasons, only the residential blocks where they live are available, but the exact positions of their houses are not clear. Additionally, people may have more than one activity place, such as house, supermarket, shopping mall and so on. Therefore, each block or place is assigned a probability. An example is shown in Fig. 1, the appearance probability of the place a_1 is 0.7.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 456–471, 2013.

Since the location of the object is uncertain, the distance between each pair of the uncertain objects is not a fixed value but a number of possible values with probabilities. Therefore, the previous approaches for answering $RkNN$ query on certain data are not directly applicable to uncertain data, and it is not trivial to process $RkNN$ query on such uncertain data.

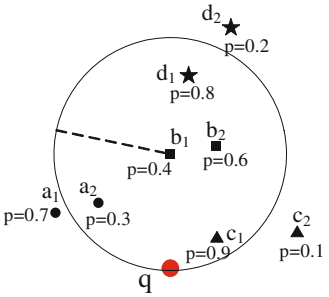


Fig. 1. An example of $R2NN$ on uncertain data

To the best of our knowledge, there are three solutions [4–6] to solve the problem of $PRkNN$ query on uncertain data when $k = 1$. However, the algorithms in [4,5] cannot be applied to the case when $k > 1$. Although the algorithm in [6] can be extended to this case, it is not very efficient with the increment of k and finding an optimal depth value *depth* for its pruning technique is not a trivial task. Motivated by this, we propose a new approach for $PRkNN$ which supports arbitrary value of k following the general framework proposed in [6]. Our contributions in this paper are summarized as follows:

1. A novel spatial pruning algorithm called $CPAI$ (angle interval based on conceptual partitioning) is designed based both on the distances between uncertain objects and the angle ranges of objects w.r.t. the query object.
2. An efficient probabilistic pruning technique called $UBPrunig$ (upper bound pruning) is proposed which utilizes the results of $CPAI$. Further, an optimized strategy is proposed to improve the pruning.
3. Extensive experiments are devised to study the performance of the proposed approach. The experimental results show that our proposed algorithm outperforms the existing solution.

The rest of this paper is organized as follows. Section 2 briefly reviews the related work on RNN query on certain and uncertain data respectively. Section 3 formally defines the $PRkNN$ query on uncertain database. Section 4 presents the techniques of spatial pruning, probabilistic pruning and verification for processing $PRkNN$ query efficiently. Extensive experiments are conducted in Section 5 to evaluate the efficiency of the methods. Finally, Section 6 concludes the paper and sketches some future works.

2 Related Work

In recent years, a number of studies have been focused on query processing on certain [1, 2, 7] and uncertain data [8–10]. Our work concentrate on the probabilistic $RkNN$ query on uncertain data, in this section, the existing work on $RkNN$ query on certain and uncertain data are briefly reviewed.

Since Korn et al. [1] introduced the concept of RNN query which is aimed to find all the objects that take the query object as their nearest neighbors, many efficient algorithms about RNN query and its variations have been proposed. The nearest neighbor for every object p was pre-computed in [1]. Yang et al. [11] designed a new index $RkNN$-tree which can efficiently support both nearest neighbor and reverse nearest neighbor query. Stanoi et al. [12] proposed a method (denoted as $60°$-*pruning*) without any pre-computation based on some interesting propositions. Xia et al. [13] extended $60°$-*pruning* to monitor the continuous RNN query on moving objects. Wu et al. [14] provided a new continuous verification method called *CRang-k* similarly based on the $60°$-*pruning* idea.

Through the above-mentioned work, it can be found that the methods on the basis of the $60°$-*pruning* idea still perform better even if the objects are moving or the k value is greater than 1. In this paper, we exactly exploit the idea to process $RkNN$ query on uncertain data for the first time.

To the best of our knowledge, there are currently three algorithms for answering $PRNN$ query on uncertain data, but all have some limitations for extending to $PRkNN$. Cheema et al. [5] (denoted as $CLWZP$) designed novel pruning rules each generated a pruning region and presented several optimizations for $PRNN$. $CLWZP$ has a very complex spatial pruning technique when computing the pruning region, which makes the method inapplicable to the $PRkNN$ where $k > 1$. Lian et al. [4] (denoted as LC) also provided techniques to solve $PRNN$ query. They focused on the case where the uncertain objects are represented by continuous probabilistic density function and approximate each uncertain object by a sphere. For the $PRkNN$ ($k > 1$), the pruning rules lose efficacy. In addition, since the probabilistic pruning sphere has to be pre-computed using τ, it is not possible to change τ during query processing. Lately, Bernecker et al. [6] proposed a general framework and developed an efficient algorithm (denoted as HP) for $PRNN$ query. They adopted a new pruning mechanism and decomposed the uncertain objects to get a more tighter bound in the probabilistic pruning phase. Though HP can be extended to $PRkNN$ ($k > 1$) as demonstrated in [6], the efficiency is dissatisfactory. Besides, finding an optimal depth value *depth* which represents the extent of decomposition is not a trivial task.

The above methods are all proposed based on the spatial relationship between uncertain objects, while our algorithm takes the angle range into account, which can improve the pruning efficiency.

3 Problem Definitions

In this section, first the definition of conventional reverse k-nearest neighbors query is reviewed; then, the uncertainty model used in this paper is presented;

Table 1. Notations

Notation	Definition
U	a set of uncertain objects
u, U_i	an uncertain object or i^{th} uncertain object in U
u_j	j^{th} instance of an uncertain object
p_{u_j}	the appearance probability of the instance u_j
λ	the user-specified probability threshold

finally, the $PRkNN$ problem in uncertain databases is formally defined. For references, the notations and their definitions used throughout the paper are summarized in Table 1.

Definition 1 (conventional reverse k-nearest neighbors, $RkNN$). *Given a set of objects P, and q is the query object, a conventional reverse k nearest neighbors query of q ($RkNN(q)$) returns all the objects $p \in P$ which take q as their k nearest neighbors. Formally, $RkNN(q) = \{p | q \in kNN(p)\}$.*

Uncertainty Model. There are two major ways in describing an uncertain object, either *continuously* which uses a probability density function (*pdf*) [8, 15] or *discretely* which uses a discrete set of alternative values associated with assigned probablities [16–20]. In this paper, we follow the discrete one. It's worth mentioning that a continuous *pdf* can be converted to a discrete one by sampling methods. Given a set of uncertain objects $U = \{U_1, ..., U_n\}$. In the discrete cases, each uncertain object U_i is represented by a set of points (instances) $u_1, ..., u_m$, and each instance u_j is assigned with an appearance probability denoted by p_{u_j}. Assuming that the probability distribution of every object is independent and the probability of each instance is independent of other instances as well. Then the equation $\sum_{j=1}^{m} p_{u_j} = 1$ holds.

Definition 2 (probabilistic reverse k-nearest neighbors, $PRkNN$). *Given a set of uncertain objects $U = \{U_1, ..., U_n\}$, a query object q, and a user-specified probability threshold $\lambda \in (0, 1]$, a $PRkNN$ query of q returns all the objects $u \in U$ that being $RkNNs$ of q with probabilities higher than λ, that is,*

$$PRkNN(q) = \{u | P_{RkNN(q,u)} \geq \lambda\} = \{u | \sum_{u_j \in u} p_{u_j} \cdot P_{RkNN(q,u_j)} \geq \lambda\} \quad (1)$$

$P_{RkNN(q,u_j)}$ denotes the probability of an instance $u_j \in u$ being the $RkNN$ of q, it can be computed as

$$P_{RkNN(q,u_j)} = \sum_{W \in \Omega} (\prod_{j'=1}^{n-1} P_{u_{j'}} \cdot \delta) \quad (2)$$

where Ω is the set of all possible worlds, and W is a possible world of Ω. $W = \{u_1, ..., u_{n-1}\}$ is a set of $n-1$ instances from different uncertain objects in U excluding the object that contains u_j. The value of δ is 1 if q is the $kNNs$ of u_j in the selected W and 0 otherwise.

Consider the example of Fig. 1 which shows the uncertain objects A, B, C, and D. Table 2 demonstrates the process of calculating $P_{R2NN(q,b_1)}$. The sum of the values (is equal to 0.202) is the probability of b_1 to be the $PR2NN$ of q. The value of $P_{R2NN(q,b_2)}$ can be computed in the same way. Finally, the probability of uncertain object B being the $P2NN$ is the sum of the two results.

Table 2. Illustration of calculating $P_{RkNN(q,b_1)}$

Possible world	Appearance probability	δ	Value
$\{a_1, c_1, d_1\}$	$0.7 \times 0.9 \times 0.8$	0	0
$\{a_1, c_1, d_2\}$	$0.7 \times 0.9 \times 0.2$	1	0.126
$\{a_1, c_2, d_1\}$	$0.7 \times 0.1 \times 0.8$	1	0.056
$\{a_1, c_2, d_2\}$	$0.7 \times 0.1 \times 0.2$	1	0.014
$\{a_2, c_1, d_1\}$	$0.3 \times 0.9 \times 0.8$	0	0
$\{a_2, c_1, d_2\}$	$0.3 \times 0.9 \times 0.2$	0	0
$\{a_2, c_2, d_1\}$	$0.3 \times 0.1 \times 0.8$	0	0
$\{a_2, c_2, d_2\}$	$0.3 \times 0.1 \times 0.2$	1	0.006
			0.202

4 *PRkNN* Processing

In the following subsections, we introduce the details of our algorithms for efficient $PRkNN$ processing following the framework proposed in [6] using the totally different techniques.

4.1 Spatial Pruning

The objective of the spatial pruning phase is to prune the objects as many as possible just utilizing their spatial locations without considering the probability distributions. In this subsection, the important insights for the $PRkNN$ problems are identified and a proposition is proposed. Next, based on this proposition, a novel spatial pruning algorithm is designed. It utilizes both the distances and the angle ranges (Definition 3) between objects, and adopts the conceptual space partitioning method proposed by Mouratidis et al. [7] as the underlying structure. At last, to further reduce the space visited, an optimized strategy is proposed.

Definition 3 (angle range). *Given an uncertain object U_i and a query q, the angle range of U_i w.r.t. q is an interval [$MinA$,$MaxA$], where MinA is the minimum angle between q and all the instances of U_i whereas MaxA is the maximum one.*

a) Proposition. As introduced in Section 2, Stanoi et al. [12] proposed the 60°-*pruning technique* to solve the RNN problem on certain data. It divides the space into 6 equal disjoint pie regions S_1 to S_6 around q. It can be proved that only the NN results of q in each region can possibly be the RNNs of q. However, the observation cannot be applied to the uncertain scenario directly. As shown in

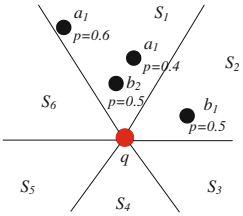


Fig. 2. 60°-*pruning* on uncertain object

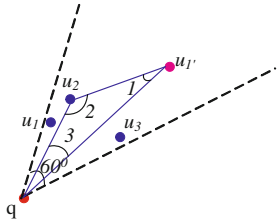


Fig. 3. Proof for Proposition 1

Fig. 2, there does not exist any object which is the $NN(q)$ in S_1 definitely. Next, the insights of the $RkNN$ query on uncertain objects are investigated and the new proposition is developed. For the convenience of discussion, we reference the spatial domination concept proposed in [21] and redefine it for $PRkNN$ query.

Definition 4 (dominance relation). *Given an uncertain object U_i, an instance $u_{j'}$ of another uncertain object and a query object q, U_i dominate $u_{j'}$ w.r.t. q iff for all instances $u_j \in U_i$ it holds that $dist(u_j, u_{j'}) < dist(q, u_{j'})$ and denoted as $U_i \prec_q u_{j'}$.*

Obviously, if k uncertain objects can be found that dominate $u_{j'}$ w.r.t. q, it can be said that the probability of $u_{j'}$ to be the $RkNN$ of q is 0. So it can be pruned in the spatial pruning phase.

Proposition 1. *Given an uncertain object U_i, an instance $u_{j'}$ of another uncertain object and a query object q, $U_i \prec_q u_{j'}$ can be inferred if they satisfy the following conditions: i) the angle range of the U_i w.r.t. q (later we omit "w.r.t.q" for shortness) is no greater than 60 degree. ii) for each instance $u_j \in U_i$, $dist(u_j, q) < dist(u_{j'}, q)$. iii) the angle formed by $u_{j'}$ and q is within the angle range of the U_i w.r.t. q.*

Proof: To show Prop.1 holds, assume w.l.o.g. that there is an uncertain object (has three instances, u_1, u_2 and u_3), and an instance ($u_{1'}$) from another object which satisfies the three conditions shown in the Fig. 3. Considering the triangle $\triangle u_2 q u_{1'}$, it is known that $dist(u_2, q) < dist(u_{1'}, q)$, so $\angle 1 < \angle 2$. According to the condition i), it can be deduced that $\angle 3 < 60° \Longrightarrow \angle 1 + \angle 2 > 120° \Longrightarrow \angle 2 + \angle 2 > 120° \Longrightarrow \angle 2 > 60° \Longrightarrow \angle 3 < \angle 2$, and therefore $dist(u_2, u_{1'}) < dist(q, u_{1'})$. The inequalities $dist(u_1, u_{1'}) < dist(q, u_{1'})$ and $dist(u_3, u_{1'}) < dist(q, u_{1'})$ both can be obtained in the similar way. Consequently, it can be concluded that $U_1 \prec_q u_{1'}$ based on the Definition 4. Hence, the proposition holds.

b) The Spatial Pruning Algorithm. The above proposition states that each uncertain object whose angle range is less than 60° can produce a pruning region. Fig. 4 shows an example, the shade regions are generated by U_1, U_2 and U_3. The spatial pruning algorithm has three steps corresponding to the three conditions in Prop.1: (1) Adopt the conceptual space partitioning method to traverse the objects from the nearest to furthest w.r.t. q. (2) Store the pruning regions in

the $AIList$ (angle interval list) in the form of three tuples $(MinA, MaxA, level)$ where $MaxA - MinA \leq 60°$. (3) Prune the instance whose angle is covered by $AIList$ k times. The details of these three steps are presented as below.

1. The conceptual space partitioning method organizes the cells around q into conceptual rectangles as illustrated in Fig. 5. Each rectangle is defined by a direction (up-U, down-D, left-L and right-R) and a level number. Our spatial pruning step access objects in ascending level order and it guarantees that the instances are visited in the order of nearest to furthest. Consequently, condition ii) in Prop.1 can be satisfied.

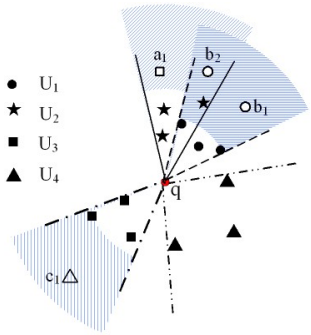


Fig. 4. An example of spatial pruning

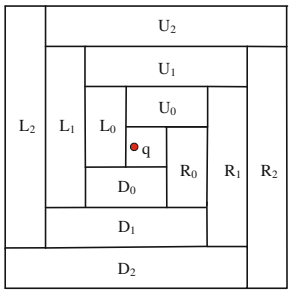


Fig. 5. The conceptual space partitioning around q

2. $AIList$ (actually is an array which consists of k lists) is maintained for $PRkNN$ query. If the angle range of an uncertain object is less than 60°, its angle range and the corresponding level number are inserted into $AIList$ in the form of $(MinA, MaxA, level)$. And if the angle intervals in $AIList$ can full cover q (the combined interval covers $[-\pi, \pi]$) for k times, the spatial phase can be terminated (according to Definition 4 and Prop.1).
3. Given an instance u_j, its level number and angle denoted by $u_j.level$ and $u_j.angle$ respectively. If there exists any k three tuples $(MinA, MaxA, level)$ in $AIList$ which satisfies the two conditions: $MinA < u_j.angle < MaxA$ and $u_j.level > level$, it can be inferred that u_j must be in the pruning regions and can be pruned safely. Or else, u_j is not in the pruning regions by now and should be inserted into the $CandidateSet$. As shown in Fig. 4, the instances a_1, b_1, b_2 and c_1 can be pruned when $k = 1$, while only the instance b_2 can be pruned if $k = 2$.

Based on the above analysis and discussion, an efficient spatial pruning algorithm is proposed for $PRkNN$, namely $CPAI$ (angle interval based on conceptual partitioning). The pseudo code of $CPAI$ is shown in Algorithm 1.

The output of $CPAI$ includes two sets, pruned set $pndSet$ and candidate set $cndSet$. Particularly, $pndSet$ contains the pruned instances which may be used to prune other instances in the next step, while set $cndSet$ contains all the instances that cannot be pruned by $CPAI$ and needs further checked.

The algorithm mainly consists of two parts, searching the angle intervals (lines 5-11) and pruning the instances based on Prop. 1 (lines 13-16). For the first part, when all the instances of an uncertain object U_i have been visited, its angle interval is inserted into $AIList$ if its angle range is less than 60°. After updating $AIList$, line 11 checks $AIList$ whether it full covers 360° k times (function $Iffullcovered$), and $CPAI$ can be terminated if it returns $true$. For the second part, if the angle of an instance is covered by $AIList$ (function $Ifcovered$), it can be safely pruned based on Proposition 1. But it is removed to the set $pndSet$ instead of being dropped for the reason that the pruned instances are likely to be used in the probabilistic pruning phase.

Since $CPAI$ terminates when $AIList$ full covers 360° k times, the performance of the spatial pruning phase is affected by the value of k and the data distribution. Meanwhile, it can be seen that the growth of k only rises the times used to decide whether 360° is covered but does not obviously increase the difficulty in computation.

Algorithm 1. CPAI(k, q, λ)

input : Value k for $PRkNN$, query object q
output: The pruned set $pndSet$ and the candidate set $cndSet$

```
1  initialize the heap H by pushing all the instances in the cell containing q into
   the heap;
2  int currentLeverl = 0;
3  while H!= ∅ do
4      for each instance u_j in H do
5          compute the angle of u_j, u_j.angle;
6          if u_j is the last instance of U_i then
7              compute the MinA and MaxA of uncertain object U_i;
8              if (MaxA-MinA) < 60° then
9                  AI(MinA, MaxA, currentLeverl);
10                 InsertAI(AI, *AIList);
11                 if Iffullcovered(*AIList)==true then
12                     return;
13         if Ifcovered(u_j.angle, AIList[i])==true then
14             insert u_j into pndSet;
15         else
16             insert u_j into cndSet
17     currentLeverl ++;
18     push all the instances whose level is equal to currentLeverl into H;
19 return;
```

c) The Optimizing Strategy. It is worth mentioning that when $AIList$ full covers a certain quadrant, the cells with larger level numbers in this quadrant can be pruned. That's because the angle range of those cells must be covered

by $AIList$. Furthermore, when the angle range of a cell is covered by $AIList$, the cell can be pruned in the same way. As shown in Fig. 6, the cells in the shaded region are not going to be visited. This optimizing strategy can help $CPAI$ further reduce the number of uncertain objects to be visited.

4.2 Probabilistic Pruning

The probabilistic pruning phase is performed for the uncertain objects whose instances cannot be pruned in the spatial pruning phase. In this subsection, an efficient probabilistic pruning algorithm (namely $UBPruing$) is designed which is based on algorithm $CPAI$. It utilizes the upper bound of the probabilities (denoted as UB) to prune the objects whose UBs are less than the probabilistic threshold λ. Moreover, an optimizing strategy is proposed to improve the pruning efficiency by sorting the objects according to their appearance probabilities in descending order.

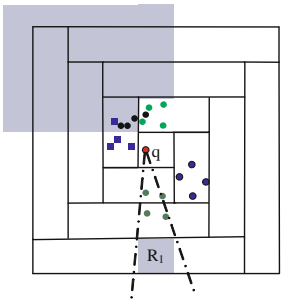


Fig. 6. The cells in the shaded region can be pruned

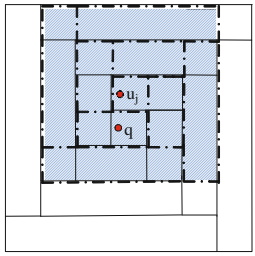


Fig. 7. The instances in $InfluSet(u_j)$ come from the shaded region

a) The Probabilistic Pruning Algorithm. Considering an uncertain object U_i, the pruning algorithm mainly consists of three part, finding its influence set (the instances having effect on its probability being the result), computing its UB and deciding whether it can be pruned or not.

Assuming u_j one of the instances in $cndSet$, what is concerned is the influence set of u_j. These influential instances are denoted as $InfluSet(u_j)$. Obviously, only the instances located much closer to u_j than q might be contained in $InfluSet(u_j)$. As introduced previously, the level number of a cell indicates the number of rectangles between q and itself, certainly, also indicates the rough distance between q and itself. If the cells around u_j are partitioned into conceptual rectangles in the similar way as q, the level number of the instances in $InfluSet(u_j)$ w.r.t. u_j must be no larger than the lever number of u_j w.r.t. q. An example is shown in Fig. 7, the instances in the shaded region formed the $InfluSet(u_j)$. Fortunately, the majority of the instances are stored in $pndSet$ and $cndSet$. So, for each uncertain object in $cndSet$, $UBPruing$ first finds its $InfluSet(u_j)$ from the two sets, and then computes its UB, finally prunes the uncertain objects whose UBs are less than the threshold λ.

Now, the formulas to compute the UB of an instance and an uncertain object are presented in Equation 3 and Equation 4, respectively. Assuming the number of uncertain objects in $InfluSet(u_j)$ is $m+1$. In Equation 3, $W = u_1, ..., u_m$ is a set of m instances from m different uncertain objects in $InfluSet(u_j)$ excepting U_i which contains u_j, and Ω denotes the set of all possible worlds among the m uncertain objects. $P_{u_{j'}}$ is the appearance probability of the instance $u_{j'}$, and $\prod_{n=1}^{m} P_{u_{j'}}$ denotes the appearance probability of one possible world in Ω. It is known that the appearance of any possible world contributes to the probability of u_j can not be the $PRkNN$ result. Hence, the sum of the appearance probabilities of all the possible worlds is the probability that u_j can not be the $PRkNN$ result. Noting that $InfluSet(u_{i_j})$ is produced just by $pndSet$ and $cndSet$ rather than the whole conceptual rectangles. Therefore, when the sum is subtracted from 1, what can be obtained is the upper bound of probability of u_j rather than the exact probability.

$$UB(u_j) = 1 - \sum_{W \in \Omega} \prod_{n=1}^{m} P_{u_{j'}} \tag{3}$$

In Equation 4, s is the number of existing instances of U_i in $cndSet$. The UB of an uncertain object is the sum of all its unpruned instances' appearance probabilities multiplies their corresponding UB.

$$UB(U_i) = \sum_{j=1}^{s} P_{u_j} \cdot UB(u_j) \tag{4}$$

Algorithm 2 illustrates the probabilistic pruning procedure discussed above. Lines 2-4 compute $UB(u_j)$ according to Equation 3, and line 5 computes $UB(U_i)$ according to Equation 4. If a certain $UB(U_i)$ is less than the probabilistic threshold λ, it is inserted to the refinement set $rfnSet$ (line 7). At last, the algorithm $UBPruning$ returns the set $rfnSet$ that contains the unpruned uncertain objects. Furthermore, they will be verified by the *verification phase*.

Algorithm 2. UBPruning($pndSet$,$cndSet$, λ)

input : Two sets $pndSet$ and $cndSet$, a probabilistic threshold λ.
output: The refinement set $rfnSet$

```
1 for each uncertain object U_i in cndSet do
2     for each instance u_j belongs to U_i in cndSet do
3         find the InfluSet(u_j) from pndSet and cndSet;
4         compute UB(u_j);
5     compute UB(U_i);
6     if UB(U_i) < λ then
7         insert it to rfnSet;
8 return rfnSet;
```

The cost of $UBPruing$ depends mainly on the computing the UB for each uncertain object in $cndSet$, therefore, the decrement of the number of uncertain objects in $cndSet$ can improve the efficiency of pruning.

b) The Optimizing Strategy. In addition, an optimal strategy is proposed for computing $UB(u_j)$ (line 4). by sorting the objects according to their appearance probabilities in descending order.

According to the Equation 3, it can be easily deduced that the more possible worlds are, the smaller $UB(u_j)$ is, and the larger $P_{u_{j'}}$ is, the smaller $UB(u_j)$ is. As the computing cost may be very expensive with the increasing number of possible worlds, $UB(u_j)$ can be relaxed by reducing the number of possible worlds being considered to improve the probabilistic pruning. The optimizing strategy gives priority to the instances having larger appearance probabilities. First it uses these instances with higher priorities to computer a loose $UB(u_j)$, and further gets the loose $UB(U_i)$ (denoted as $RUB(U_i)$. $RUB(U_i) > UB(U_i)$), then compare $RUB(U_i)$ with threshold λ. If $RUB(U_i) < \lambda$, U_i can be safely pruned right now, otherwise more possible worlds should be considered to shrink the $RUB(U_i)$.

4.3 Verification

In the verification phase, the algorithm computes, for each remaining uncertain object in $rfnSet$, the exact probability being the result of $PRkNN$ of q. An uncertain object U_i for example, the algorithm can issue a probabilistic k-nearest neighbors ($PkNN$) query of it using the existing method [22]. Obviously, the probability of q being the result of $PkNN$ of U_i is equal to the probability of U_i being the result of q. If the value is larger than the threshold λ, it is returned, otherwise it is discarded. It can be seen that the number of uncertain objects in $rfnSet$ directly determines the cost of verification phase. Therefore the larger the value of λ is, the better the performance of verification will be.

5 Experimental Evaluation

5.1 Experiment Setup

We compare our algorithm (named $AIUB$) with the state-of-the-art $PRkNN$ query algorithm (named HP) proposed in [6]. HP uses two $R*$-trees, a global one is for organizing the uncertain object approximations and a local one is to index the instances of an uncertain object. We set the page size 1024 bytes for the global tree, to a maximal capacity of three entries for the local trees, and set the *depth* (representing the extent of partition) to 2. And the above settings are recommended in the original paper. All the algorithms evaluated are implemented in C++ with STL library support and compiled with GNU GCC. The hardware platform is one IBM X3500 sever with 2 Quad Core 1333MHz CPUs and 16G bytes memory under linux (Red Hat 4.1.2-42).

For the experiments, synthetic uncertain objects under the space with size (16000×16000) are generated as follows: first, the centers (Gaussian or uniform distribution) of the uncertain objects are created, then the instances of each object are generated within their corresponding rectangles following uniform distribution. The length of the rectangles is set based on the whole space and the percentage varies from 1 to 4. Additionally, the appearance probabilities of

Table 3. Specifications of Parameters

Parameter	Value range	Default value
# uncertain objects	2000, 4000, 6000, 8000	6000
Maximum # instances per object	100, 200, 400,800	200
Probability threshold λ	0.1, 0.3, 0.5, 0.7, 0.9	0.5
Value of k	5, 10, 15, 20, 25	15
Grid division	100^2, 200^2, 400^2, 800^2	200^2
Maximum length of rectangle	1%, 2%, 3%, 4%, 5%	2%
Distribution of object centers	Gaussian, uniform	uniform

instances are also generated following either Gaussian or uniform distribution. Analogously, the query objects follow the same distribution as the underlying data set. Table 3 summarizes the parameters used in our experiments which may have a potential impact on the performance. We evaluate the performance under various parameters and data settings and all the parameters use the default values unless specified. For each setting, we issue 100 queries and display the average total cost (I/O and CPU time).

5.2 Evaluation Results

Fig. 8 shows the different numbers of candidate objects remained by the spatial pruning steps of the two methods under the variation of the k value. It can be seen that when $k < 10$, HP and $AIUB$ have the similar pruning power. Nevertheless, $AIUB$ performs better than HP as the value of k increases. That's because HP only considers the spatial relationship between objects, and the pruning rule cannot work well when k is too large. While $AIUB$ utilizes both the distance and the angle range to prune objects, the only job $AIUB$ need to do is to find more 360° full covers when k increases.

Fig. 9 reports the whole query time against the value of k. As expected, Fig. 8 and Fig. 9 have a similar trend. Compared to HP, the query speed of $AIUB$

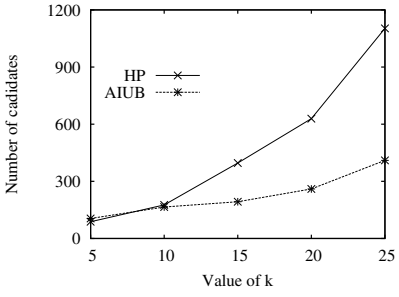


Fig. 8. Number of candidates with different values of k

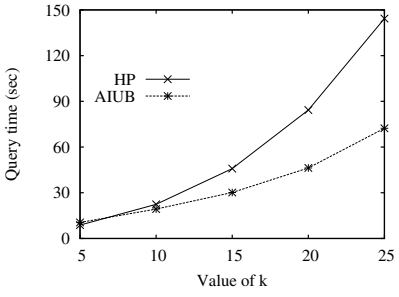


Fig. 9. Query time with different values of k

is almost double faster when k is larger than 15. This is mainly because that the larger number of candidate objects returns, the more complex the probability pruning and verification phases are. Thus it can be seen that the spatial pruning power is very significant for the whole processing.

Fig. 10 evaluates the performances of HP and $AIUB$ with increasing number of uncertain objects. It is not surprised to say that query time increases as the number of objects gets larger. The computation cost of them both increase mainly due to the increased verification cost caused by the returned larger number of objects (and in effect instances). Nevertheless, the scalability of $AIUB$ is better than that of HP regarding to the growth of k.

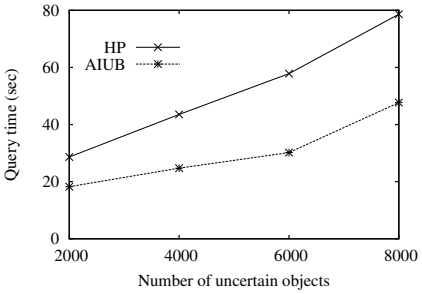


Fig. 10. Query time with different numbers of uncertain objects

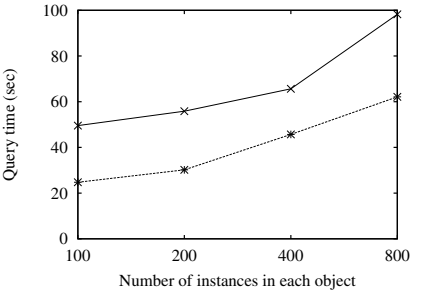


Fig. 11. Query time with different numbers of instances in each object

Fig. 11 shows the effect of the maximum number of instances in each object. The performance degrades as the number of instances increases. That is because the verification phase becomes more expensive if each object has larger number of instances.

In Fig. 12, we increase the probability threshold λ from 0.1 to 0.9. The query time of both the algorithms decreases as the probability threshold increases. The rationale is that with a greater value of λ, a majority of objects are pruned and not requires to be verified which reduces the computation expense.

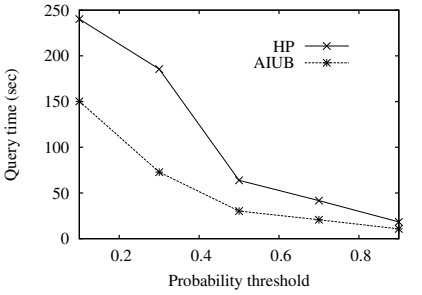


Fig. 12. Query time with different probability thresholds

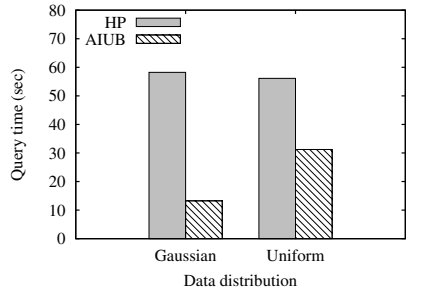


Fig. 13. Query time with different data distributions

Fig. 13 illustrates the query time of HP and $AIUB$ under different data distributions. HP has the similar performance with the two different data distributions, while $AIUB$ has a clear advantage on Gaussian distribution. The reason why $AIUB$ on Gaussian data performs better than the uniform data is that $AIUB$ prunes instances level by level around q in the spatial pruning phase, and it is more efficient under Gaussian distribution.

Fig. 14 investigates the performances of HP and $AIUB$ with different maximum length of the rectangle which represent the region of the instances of an uncertain object. The performance of both algorithms degrades against the growth of length. For $AIUB$, the smaller rectangle means that the angle range of the uncertain object is more likely less than 60°, which can enhance the spatial pruning power. When the length of rectangle is set to 4% or more of the whole space, the performance degrades smoothly. This is because too much overlap of objects invalids a majority of objects, and a few number of objects require verification.

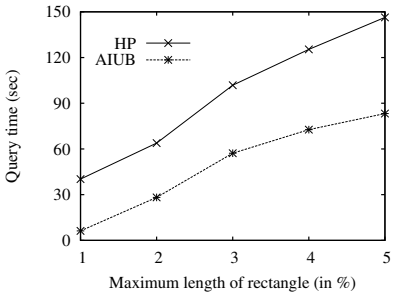


Fig. 14. Query time with different maximum length of rectangle

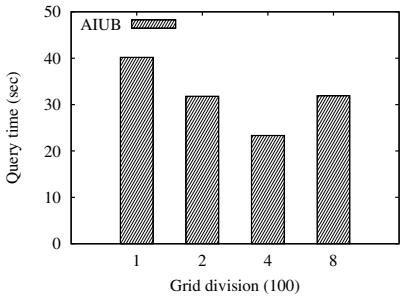


Fig. 15. Query time with different grid divisions

Fig. 15 shows the running time of each phase of $AIUB$ with the different grid granularity. It can be seen that when we use grid index of 400×400 cells, $AIUB$ performs the best. This is mainly due to that $AIUB$ prunes objects utilizing the objects in the lower level, too small division decreases the number of objects in lower levels and weaken the pruning power of $AIUB$. As the grid division increases, the performance degrades again. That is because too many cells makes it costly to manage them.

6 Conclusions and Future work

In this paper, we discussed the problem of reverse k-nearest neighbors on uncertain data and proposed a pipeline algorithm. We designed a spatial pruning technique utilizing the distances between objects and the angle ranges of objects w.r.t. query object. Based on the spatial pruning, an efficient probabilistic pruning algorithm was proposed followed by an optimizing strategy. The extensive experiments verified the pruning power and the efficiency. The pruning power

and the efficiency were evaluated in a simulated environment, the results show that our proposed algorithm has a better performance and scalability than the existing solution when $k > 1$.

In the future, we will focus on the extension of our method to $PRkNN$ query on moving objects, and adapt it to the case that the query object is also uncertain.

Acknowledgement. This research was partially supported by the National Natural Science Foundation of China under Grant Nos.61173030, 61025007 and 61272182; the 863 Program under Grant No.2012AA011004.

References

1. Korn, F., Muthukrishnan, S.: Influence sets based on reverse nearest neighbor queries. In: SIGMOD, pp. 201–212. ACM (2000)
2. Stanoi, I., Riedewald, M., Agrawal, D., El Abbadi, A.: Discovery of influence sets in frequently updated databases. In: VLDB, pp. 99–108 (2001)
3. Tao, Y., Papadias, D., Lian, X.: Reverse knn search in arbitrary dimensionality. In: VLDB, pp. 744–755. VLDB Endowment (2004)
4. Lian, X., Chen, L.: Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data. The VLDB Journal 18(3), 787–808 (2009)
5. Cheema, M., Lin, X., Wang, W., Zhang, W., Pei, J.: Probabilistic reverse nearest neighbor queries on uncertain data. TKDE 22(4), 550–564 (2010)
6. Bernecker, T., Emrich, T., Kriegel, H., Renz, M., Zankl, S., Züfle, A.: Efficient probabilistic reverse nearest neighbor query processing on uncertain data. VLDB 4(10), 669–680 (2011)
7. Mouratidis, K., Papadias, D., Hadjieleftheriou, M.: Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring. In: SIGMOD, pp. 634–645. ACM (2005)
8. Beskales, G., Soliman, M., IIyas, I.: Efficient search for the top-k probable nearest neighbors in uncertain databases. VLDB 1(1), 326–339 (2008)
9. Tong, Y., Chen, L., Ding, B.: Discovering threshold-based frequent closed itemsets over probabilistic data. In: ICDE, pp. 270–281. IEEE (2012)
10. Tong, Y., Chen, L., Cheng, Y., Yu, P.: Mining frequent itemsets over uncertain databases. In: VLDB, pp. 1650–1661 (2012)
11. Yang, C., Lin, K.: An index structure for efficient reverse nearest neighbor queries. In: ICDE, pp. 485–492. IEEE (2001)
12. Stanoi, I., Agrawal, D., Abbadi, A.: Reverse nearest neighbor queries for dynamic databases. In: SIGMOD, pp. 44–53 (2000)
13. Xia, T., Zhang, D.: Continuous reverse nearest neighbor monitoring. In: ICDE, p. 77. IEEE (2006)
14. Wu, W., Yang, F., Chan, C., Tan, K.: Continuous reverse k-nearest-neighbor monitoring. In: MDM, pp. 132–139. IEEE (2008)
15. Cheng, R., Chen, J., Mokbel, M., Chow, C.: Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In: ICDE, pp. 973–982. IEEE (2008)
16. Hua, M., Pei, J., Zhang, W., Lin, X.: Ranking queries on uncertain data: A probabilistic threshold approach. In: SIGMOD, pp. 673–686. ACM (2008)
17. Widom, J.: Trio: A system for integrated management of data, accuracy, and lineage. Technical Report (2004)

18. Kriegel, H.-P., Kunath, P., Renz, M.: Probabilistic nearest-neighbor query on uncertain objects. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 337–348. Springer, Heidelberg (2007)
19. Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: VLDB, pp. 15–26. VLDB Endowment (2007)
20. Soliman, M., Ilyas, I., Chen-Chuan Chang, K.: Top-k query processing in uncertain databases. In: ICDE, pp. 896–905. IEEE (2007)
21. Emrich, T., Kriegel, H., Kröger, P., Renz, M., Züfle, A.: Boosting spatial pruning: on optimal pruning of mbrs. In: SIGMOD, pp. 39–50. ACM (2010)
22. Cheng, R., Chen, L., Chen, J., Xie, X.: Evaluating probability threshold k-nearest-neighbor queries over uncertain data. In: EDBT, pp. 672–683. ACM (2009)

Efficient Querying of Correlated Uncertain Data with Cached Results

Jinchuan Chen[1], Min Zhang[2], Xike Xie[3], and Xiaoyong Du[1,2]

[1] Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University of China), MOE, China
{jcchen,duyong}@ruc.edu.cn
[2] School of Information, Renmin University of China
zhangmin0453@ruc.edu.cn
[3] Department of Computer Science, Aalborg University, Denmark
xkxie@cs.aau.dk

Abstract. Although there have been many efforts for management of uncertain data, evaluating probabilistic inference queries, a known NP-hard problem, is still a big challenge, especially for querying data with highly correlations. The state-of-art exact algorithms for accelerating the evaluation of inference queries are based on special indices. Besides, with the observation of the existence of many frequent queries, some researchers try to improve efficiency by reusing previously queried results. Indexing depends on the static properties like data distributions, whereas caching is in favor of the dynamic features like query workload. In this paper we propose a new approach for speeding up the evaluation of inference queries by caching frequent results in a junction tree-based hierarchical index. To the best of our knowledge, this is the first effort on utilizing both the static (data) and dynamic (query workload) properties to efficiently evaluate probabilistic inference queries. Moreover, according to our experience, different caching strategies may significantly affect the query performance. Basically a good caching strategy needs to have high cache hit ratio with limited space budget.Based on these considerations, we propose a novel caching approach, called *FVEC*, and present corresponding algorithms for efficiently querying correlated uncertain data. We further conduct a series of extensive experiments on large uncertain datasets in order to illustrate the effectiveness and efficiency of our proposed approaches. As illustrated by the results, compared with previous solutions, our method could greatly improve the query performance.

1 Introduction

Recently data uncertainty becomes an intrinsic property in many applications. For example, in an information integration system [6], the automatically generated schema mappings could be imprecise. Hence the data in the integrated database would be uncertain. As another example, when extracting information from unstructured data like web pages, top-k possible results would be generated and the database obtained would be a probabilistic database [18]. Due to

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 472–486, 2013.

its high applicability, the management of uncertain data is attracting more and more research interests during the past decade.

One of the core problems for managing uncertain data is the evaluation of probabilistic queries, a known #P-complete problem [5]. Generally, a probabilistic query is to compute the probabilities of some complex random events based on the joint distribution of all the uncertain data items in a dataset. In probabilistic theory, it is exactly the process of *probabilistic inference*. Specifically, we only discuss the queries of extracting the joint distribution of a set of variables from the whole joint distribution. But our methods could be applicable for general cases. In this paper, we use the two terms *probabilistic query* and *inference query* interchangeably.

Many research works are proposed for solving this problem. Most of these approaches are based on the assumption that uncertain data are either fairly independent or partitioned in independent groups. Kanagal et al. propose a novel index called INDSEP to efficiently query correlated uncertain data [8]. The main idea of INDSEP is to cluster uncertain data into partitions and organize them in a hierarchical structure. When evaluating probabilistic queries, INDSEP can then prune many computation efforts with the hierarchical structure and the pre-computed *shortcut* potentials.

The INDSEP structure is designed as a disk-based index. The main part of the index is stored in disk and will be loaded into main memory in need. The size of each index node is limited by the size of a disk page. However, as found in many projects [3,5,9], the major bottleneck of answering probabilistic queries is not I/O but CPU. Nowadays, it is not that expensive to buy a machine with several gigabytes RAM, but the response time could still be quite long even if the whole INDSEP structure has been loaded into the main memory.

On the other hand, in reality the *Pareto principle* is a common phenomena in many applications. This principle, also known as the 80-20 rule, states that roughly 80% of the queries focus on 20% data items [1]. With this observation, and the availabity of large main memory, an intuitive solution is to cache frequent results and use them for answering queries. In [15], the frequent intermediate results are stored in the main memory in order to reduce response time.

The indecis are built based on the static properties like data distributions, e.g. two variables X and Y would be probably placed in the same partition if they have strong correlations. On the contrary, when deciding which items should be cached, the major concern is the dynamic features like query workload, e.g. we may consider to store the joint distribution of $\{X, Y, Z\}$ if it is contained by many previous queries. Intuitively, we can further improve the query performance by benefiting from both the static (data) and dynamic (workload) features. Our work is motivated by this observation. In brief, our basic idea is to adapt the INDSEP structure to accommodate the frequent accessed results. We also revise the original algorithm for evaluating inference queries on INDSEP so that the cached results could be utilized.

However, this approach seems not to work well if we directly adopt the caching strategy in [15], i.e. storing frequent intermediate results. According to our

experimental results, this caching method, called *SubQuery* in this paper, can only improve the query performance by about 10% compared with original INDSEP. The reasons are two folds. Firstly, there are huge number of possible intermediate queries and the cache hit ratio of *SubQuery* cannot be satisfactorily high. Note that an intermediate query is exactly to extract the joint distribution of a set of random variables. The number of possible intermediate queries, or combinations of variables, would be extremely huge when the data set is large. The second reason lies in its high space consumption. Storing a joint distribution requires to store all the entries of this distribution, whose volume will increase exponentially along with the enlargement of the number of variables contained in the distribution. The *SubQuery* approach usually requires to cache sub-queries with relatively large cardinality and consumes lots of space. We will revisit this issue in Sec. 3.1.

With this concern, we propose a new caching technique, called *Frequent-Variables*, which only store the joint distribution of frequent single variables with their corresponding separators. Separators are special variables in junction-tree which are required to *connect* the partial results from different partitions. More details about separators and junction-tree would be discussed in Section 2. Compared with *SubQuery*, the *Frequent-Variables* technique could control the size of cached joint distributions because usually there are only few separators. In this way, we could reduce the memory consumption. The cache hit ratio would also be improved since we now able to reserve more frequent variables.

More importantly, different from [15], we are not only trying to match the intermediate sub-queries with cached results, but also pruning unnecessary intermediate sub-queries and consequentially shortening the evaluation path. Specifically, we find that we can use some cached results for acceleration even if the cached variables are NOT included in a query. Based on this observation, we propose the so-called *Express-Channels* technique which could further save lots of subqueries. Our proposed method integrates these two techniques. Hence we call it *FVEC (Frequent Variables with Express Channels)*.

Our major contributions in this paper are summarized as follows:

- An approach for efficiently querying large volume of uncertain data with high correlations based on cached results.
- Novel caching approaches for storing the requent results.
- Efficient algorithms for evaluating inference queries based on the cached information.
- Extensive experiments on large datasets for verifying the effectiveness and efficiency of our proposed approaches.

Next we will review some issues which our work is based on, such as uncertain data model, the junction tree and the INDSEP structures in Section 2. After that, we will propose our approaches for caching frequent variables in Section 3, and a series of techniques to make good use of cached results to prune subqueries in Section 4. The experimental study will be illustrated in Section 5. Section 6 will discuss related works. We then conclude this paper in Section 7.

2 Background Knowledge

In this section, we briefly review several important issues related to our work. We will first introduce a data model for representing uncertain data.Secondly, we will show the basic idea of junction tree [10], a data structure for efficient processing of inference queries, and explain the structure of INDSEP [8], an index for querying correlated uncertain data.

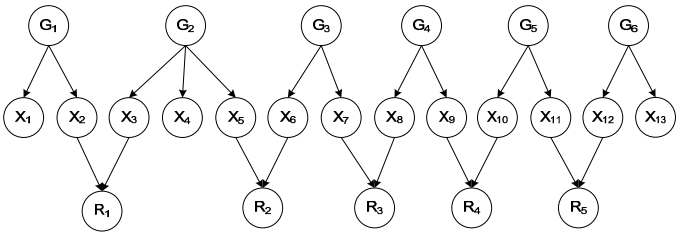


Fig. 1. GRN Representation for a Probabilistic Database

2.1 GRN Model

There have been several models for representation of uncertain data such as attribute uncertainty and tuple uncertainty [14]. But most models are only applicable for data with no correlations. Recently Chen et al. propose a model called *GRN* (**G**enerator-**R**ecognizer **N**etwork) for expressing correlated uncertain data [2]. As our work focuses on querying highly correlated uncertain data, we decide to adopt the GRN model.

The GRN model is a special Bayesian Network. Figure 1 illustrates a simple example for representing a probabilistic database in GRN. Each node is labeled with the random variable to which it is corresponded. Each arrow depicts a dependency among variables and is attached with a conditional probability table (CPT). The nodes labeled by $X_1 \ldots X_{13}$ are tuple variables. Basically each tuple in the probabilistic database will have a corresponding node in GRN. The nodes $G_1 \ldots G_6$ are called *generators*, which are inserted to express the correlations among tuple variables. For example, with G_1 and the attached CPTs, X_1 and X_2 should be mutually exclusive. Finally, the nodes $R_1 \ldots R_5$ are recognizers for representing the correlations among X-tuples, e.g. two groups $\{X_1, X_2\}$ and $\{X_3, X_4, X_5\}$ are correlated with the existence of R_1. .

2.2 Junction Tree and INDSEP

Intuitively, the evaluation efficiency would be much better if we can perform inference on a small part of the whole distribution, which leads to the introduction of junction tree [10]. The nodes in the Bayesian network would be clustered into *cliques*. Each clique has the property that all nodes inside it are pairwise linked. A junction tree will be obtained then by applying an elimination sequence on the triangulated graph.

In a junction tree, there are two types of nodes, *clique nodes* and *separator nodes*. The clique nodes in the junction tree correspond to the maximal cliques in GRN and the separator nodes correspond to the vertex sets that separate the maximal cliques. Suppose χ is the set of all random variables, C_i is denoted as a clique and s_j is denoted as a separator, the overall joint distribution represented by a junction tree can be computed as follows.

$$P(\chi) = \frac{\prod_{C_i} P(C_i)}{\prod_{s_j} P(s_j)} \tag{1}$$

After construction and calibration, the potential of C_i (or s_j) is exactly the marginal distribution of C_i (or s_j).

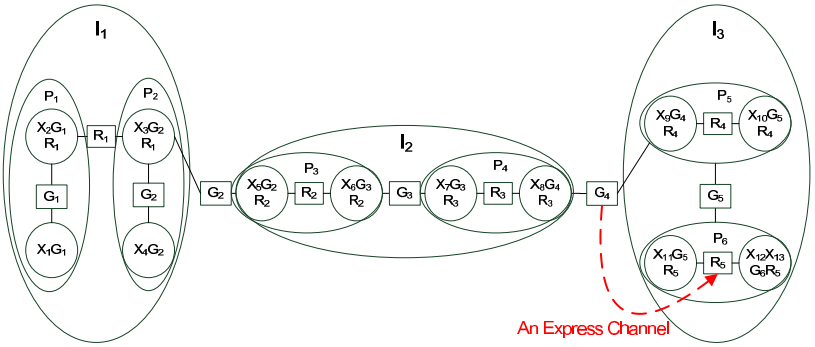


Fig. 2. The Partition Result for the Junction Tree Builit on Figure 1

Kanagal et al. find that we could avoid going into some of these intermediate nodes if the joint distributions of some special variables are pre-computed. With this observation, they propose the so-called INDSEP structure [8], which is to hierarchically partition a junction tree into connected subtrees and subsequently construct the index. Figure 2 shows a hierarchical partition of the junction tree built on the above GRN. Each index node in INDSEP corresponds to a connected subtree of the junction tree, and it has a set of separators which separate this node from its siblings. A separator is basically a set of random variables. The joint distribution of all the separators of a node is called *shortcut* potential and can be used for speeding up query processing.

For completeness, we now briefly summarize the main idea of answering inference queries on INDSEP. When receiving an inference query, INDSEP will first search for the children nodes of the root which contain the queried variables. Next, a so-called Steiner tree [7] will be constructed which connect all the target nodes. For each node in the Steiner tree, a subquery will be issued on it which includes not only the corresponding query variables, but also separators of this node. For the cases where a subquery contains separators only, INDSEP just takes the node's shortcut potential into computation. After obtaining the results

of all subqueries, the final result will be obtained by computing the joint distribution of all subquery results and intermediate separators, and marginaling out the redundant variables. Note that this process is recursive, i.e. the evaluation of each subquery will follow the same process before going down to the leaf level. Imagine in an INDSEP containing thousands of nodes, the query performance could be quite low when the queried variables are far away from each other. The performance could still be improved greatly if the results of some queried variables have been cached.

3 Caching Frequent Query Results

In this section, we will illustrate our approaches for caching frequent query results, i.e. *SubQuery*, and *FVEC*. We also give a detailed discussions to explain why they have different performance.

Table 1. Results Cached in Each Node in Example 1

Node	Cache(SubQuery)	Cache(FVEC)
I_1	$\{X_1, X_2, X_4, G_2\}$	$\{X_1, G_2\}, \{X_2, G_2\}, \{X_4, G_2\}$, $\{R_1, G_1, G_2\}$
I_2	$\{X_5, X_6, X_8, G_2, G_4\}$	$\{X_5, G_2, G_4\}$,$\{X_6, G_2, G_4\}$,$\{X_8, G_2, G_4\}$, $\{R_2, G_2, G_4\}$, $\{R_3, G_2, G_4\}$
I_3	$\{X_{12}, G_4\}$	$\{X_{12}, G_4\}$, $\{R_5, G_4\}$
P_1	$\{X_1, X_2, R_1\}$	$\{X_1, R_1\}$, $\{X_2, R_1\}$
P_2	$\{X_4, R_1, G_2\}$	$\{X_4, R_1, G_2\}$
P_3	$\{X_5, X_6, G_2, G_3\}$	$\{X_5, G_2, G_3\}$, $\{X_6, G_2, G_3\}$
P_4	$\{X_8, G_3, G_4\}$	$\{X_8, G_3, G_4\}$
P_6	$\{X_{12}, G_5\}$	$\{X_{12}, G_5\}$

Example 1. For ease of illustration, we now give an example before touching the detailed cache approaches. Suppose we now need to obtain the marginal distribution $P(X_1, X_2, X_4, X_5, X_6, X_8, X_{12})$ over the INDSEP structure shown in Figure 2. The query is first performed on the root node. Since these variables locate in I_1, I_2 and I_3 respectively. The Steiner tree for connecting the two nodes would be $I_1 - I_2 - I_3$. Three sub-queries are then generated for these three nodes, i.e. (X_1, X_2, X_4, G_2) for I_1, $(X_5, X_6, X_8, G_2, G_4)$ for I_2 and (X_{12}, G_4) for I_3. Note that when a children node v is queried, its separator S_v must be included in the query. Otherwise we cannot assembly the final result with the partial ones [8]. This process will be repeated for each of these sub-queries until either the result of a sub-query is found in one shortcut or a leaf node is met.

3.1 The SubQuery Approach

As shown in Example 1, when a query is evaluated over an INDSEP tree, many sub-queries would be generated. Each sub-query may also be broken down into several new children sub-queries. Following [15], the *SubQuery* approach will store the results of frequent sub-queries at the corresponding nodes. For example, since the sub-query (X_1, X_2, X_4, G_2) is queried over I_1, we then store the result $P(X_1, X_2, X_4, G_2)$ at node I_1. Similarly, we will store $P(X_5, X_6, X_8, G_2, G_4)$,

$P(X_{12}, G_4)$ at I_2 and I_3 respectively. In this way, when processing new queries, we can avoid re-evaluate some intermediate queries if they are found in the cache. Table 1 lists the cached data at each node after evaluating the query in Example 1 according to the *Subquery* approach. Moreover, we attach a memory limit on each node, and adopt the classical LRU (Least Recently Used) algorithm for updating a cache when it is full.

Discussions. According to the experimental results, the *SubQuery* approach could really save the computation of many sub-queries and reduce the response time. But the improvement is only about 10% compared with the original IND-SEP without any cache. After careful analysis, we find an interesting phenomenon that most cache hittings appear in low-level nodes. Due to the recursive evaluation process, a sub-query generated in a high-level node usually results in much more sub-queries than one generated in a low-level node. Hence it is more valuable to kill sub-queries in high-level nodes. In our experiments, most killed sub-queries by the *Subquery* approach locate in 4-5 levels (with tree height equals to 5). That is why the performance is only slightly improved.

Why it is so hard to hit high-level caches? The reason lies in the number of sub-queries. Note that a sub-query is exactly a combination of variables, and the number of possible sub-queries would increase very fast when its cardinality enlarges. On average, the cardinality of sub-queries appearing in high levels would be larger than that of sub-queries in low levels. Hence there are large number of possible sub-queries for high-level nodes and the chances of finding a new query in the cache would be very low.

Another shortcoming of the *Subquery* approach is its high memory cost. The data stored in cache are some joint distributions of several variables. The number of entries in a joint distribution would increase exponentially with the number of variables contained in it. For sub-queries with big cardinality, the memory budget would be quickly run out, and we have to perform the cache updating very often. Again, this problem will become much more serious in high-level nodes where sub-queries are quite large. Frequently cache updating causes not only extra time cost, but also low cache hit ratio.

With these observations, we decide to decompose the results of sub-queries into small fragments, i.e. results of single variables. In this way, we are able to accommodate more distributions within the same cache size, and to identify frequent variables. Furthermore, we find that by leveraging some special variables we could build *express channels* between the nodes in high-level and low-level. With these channels, we could skip large parts of the evaluation path and reduce the overall evaluation time. This approach, called *Frequent Variable with Express Channel* (FVEC in short) will be illustrated in the next part.

3.2 The FVEC Approach

The FVEC approach is based on the idea of caching the distributions of frequent variables. However, it is no use to cache the marginal distribution of any single variables. For example, we cannot obtain the joint distribution of $P(X_1, X_2)$

from the cached results $P(X_1)$ and $P(X_2)$ because they are not independent. Specifically, we need to choose some *bridge* variables for a frequent variable according to the *d-separation* theorem [2] and store the joint distribution of the frequent variable and its bridges. In INDSEP, these bridge variables are usually the separators of the node containing the frequent variable. For example, at node I_1 in Fig. 2, we should store $P(X_1, G_2)$ at I_1 when X_1 is found to be frequent since G_2 is the separator of I_1. Similarly, we store $P(X_5, G_2, G_4)$ at node I_2. Therefore, if (X_1, X_5) is queried in the future we could derive it by $P(X_1, G_2) * P(X_5, G_2)/P(G_2)$. Here $P(X_5, G_2)$ could be computed from $P(X_5, G_2, G_4)$.

Definition 1. Frequent-Variable. *Suppose a variable X is found to be frequent in a node v, the Frequent-Variable scheme will store $P(X, S_v)$ at this node. Here S_v is the set of separators of v.*

Now we explain the idea of *express channel*. In Example 1, in order to evaluate the sub-query (X_{12}, G_4), we need to build a long evaluation path from the root node to the leaf node containing X_{12}. During this process, we must process a set of sub-queries, i.e. (G_4, G_5), (G_5, R_5) and (R_5, X_{12}), and obtain the final result based on Equation 1. Suppose next time we want to query X_{13}. We have to go through the same evaluation path and evaluate almost the same set of sub-queries except that the sub-query (R_5, X_{12}) is replaced by (R_5, X_{13}).

The evaluation of X_{13} could be accelerated if we cache the result $P(G_4, R_5)$. In this case, we just have two sub-queries (G_4, R_5) and (R_5, X_{13}). As shown by the dotted red arrow in Fig. 2, this cache scheme is like to build a channel from I_2 to the leaf node. Hence we call it *express channel*. Note that in practical there could be hundreds of thousands of variables and the evaluation path would be much longer. Therefore this method could help us cut off many partial evaluation path and skip many sub-queries.

Definition 2. Express-Channel. *If a node v is accessed, we would store the joint distributions of S_v and S'_v at each ancestor node v' of v.*

The major advantage of the express-channel scheme is the ability of killing many sub-queries in high-level nodes. As aforementioned, with express channels, we can directly pull the variables from leaf nodes. Moreover, note that with this scheme we can accelerate queries with no cached variables, e.g. the above query X_{13}. This is because these channels are linked with separators and these separators will be shared by different queries.

Finally, the **FVEC approach** is to store the results according to both frequent-variable scheme and express-channel scheme. The cached data according to FVEC approach in Example 1 are listed in Table 1. The items cached due to the express-channel scheme are marked as red color.

4 Processing Inference Queries with Cached Results

We are now ready to illustrate the process of evaluating probabilistic inference queries with cached results.

4.1 Querying Algorithm

As explained in Sec. 3, the frequent results are cached in some nodes of INDSEP. The original algorithm of processing inference queries over INDSEP, as proposed in [8], needs to be revised in order to make use of cached results. Algorithm 1 lists the main steps of the revised querying algorithm, called *queryWithCache*. The revised parts are highlighted in red color.

```
1  if v is a leaf node then
2      return extract(v.potential,q);
3  if config.option = 'FVEC' then
4      v' ← findExpressChannel(v,q);
5      if v' is NOT null then
6          return queryByChannel(v,v',q);
7  construct a steiner tree T for q over the children nodes of v;
8  Φ ← ∅;
9  foreach node v' in T do
10     q' ← S_v' ∪ (all variables inside v' ∩ q) ;
11     if q' is a sub set of any shortcut sc of v or v' then
12         Φ ← Φ∪ extract(sc,q');
13         continue;
14     r' ← searchInCache(q',v');
15     if r' is null then
16         r' ← queryWithCache(v',q');
17     Φ ← Φ ∪ {r'} ;
18 r ← (∏_{φ∈Φ} φ) / (∏_{s∈T} P(s));
19 return r;
```

Algorithm 1. queryWithCache(Node v, Query q)

This algorithm works in a recursive manner. When a query q is evaluated, we invoke *queryWithCache* and set the root node as its input. Steps 1 and 2 are to check whether the input node v is on the leaf level. If the answer is YES, we can directly obtain the result by marginalizing q from v's potential. The function $extract(Factor, Query)$ is to extract the marginal distribution of a set of variables from a factor[1]. Steps 4 to 6 are only executed if the programme works in the 'FVEC' mode, i.e. we can use the *Express Channel* to skip some parts of the evaluation path. The details of leveraging express channels will be discussed in Sec. 4.2. If a query cannot be accelerated by express channel, we then build a Steiner tree at step 7, a path containing all variables in q. Step 8 is to initialize

[1] In this paper, we will use the terms "factor" and "distribution" interchangeably and ignore their difference in the probabilistic graphical model literatures.

an empty set for storing results of sub-queries. Next we evaluate each sub-query for every node v' contained by the Steiner tree (steps 9-17). Firstly, we build a sub-query (step 10) which contains the variables in v''s separators, or in the intersection of the variables in v' and in q. Steps 11-13 try to find the result in any short-cut of v or v'. If not found in short-cut, we then check whether the result could be found in cache (steps 14-16). In step 17, we insert the result of this sub-query into Φ. Finally, after collecting the results of each sub-query, we obtain the result by Equation 1.

The $searchInCache(q, v')$ function (step 14) tries to find a match of q in node v'. Here q *matches* a cached factor means that q is a subset of the variables contained in this factor.

4.2 Leveraging Express Channels

The use of *express channels* to accelerate query evaluation consists of two major steps. Firstly, we need to find whether there exists a channel and, if there are multiple ones, choose the longest one. This task is solved by the $findExpressChannel(v, q)$ function. A channel is represented by a children node of v, which means the item $P(S_v, S'_v)$ is contained in the cache of v. Therefore, the path from v to v' could be replaced by this channel factor. After that, we invoke the function $queryByChannel(v, v', q)$ to obtain the result of q which would cut off the path from v to v'.

Algorithm 2 lists the process of finding express channels. At the first step, we retrieve the lower common ancestor (LCA) of all variables in q because a channel must be shared by all the queried variables. Hence each common ancestor of all queried variables is a candidate to construct a channel, with the LCA generating the longest one. Steps 3 is to check whether the path $< v, v' >$ exists, i.e. the joint distribution of $S_v \cup S'_v$ is cached. If not, we then set v' as its parent node and repeat this process. The loop ends if we find v' becomes v, which means we could not find a children node of v which can build a channel. A *null* value will be returned for this case.

The $queryByChannel(v, v', q)$ function is not complicated. As listed in Algorithm 3, the input query q is decomposed into two parts. The first part, i.e. (v', q) is obtained by invoking the $queryWithCache$ function, which is not costly since v' usually locates in the levels very close to the leaf node. The second part, i.e. (v, v'), can be loaded from cache. The final result is then obtained by combining these two parts with Equation 1 (step 3).

5 Experimental Study

We now report our experimental results. All programs are implemented in C++ under gcc 4.1.2, and run on a machine with six-core 2.4G Hz Intel(R) Xeon(R) CPU, 16G RAM and Linux 2.6.18. For comparison, we implement the INDSEP index [8] based on an open source C++ library libDAI[2]. Without otherwise specifications, each point in the following figures is an average of 200 runs.

[2] `http://cs.ru.nl/~jorism/libDAI/`

```
1 v' ← getLCA(q) ;
2 while v' != v do
3     if P(S_v ∪ S'_v) can be found in cache then
4         return v';
5     v' ← the parent node of v';
6 return null;
```

Algorithm 2. findExpressChannel(Node v, Query q)

```
1 r_1 ← queryWithCache(v', q ∪ S_{v'}) ;
2 r_2 ← searchInCache(S_v ∪ S_{v'}, v) ;
3 r ← r_1 r_2 / ∏_{s∈S_{v'}} P(s) ;
4 return r;
```

Algorithm 3. queryByChannel (Node v, Node v',Query q)

Datasets and Queries. We generate a probabilistic database with one relation and 100,000 tuples, which is represented in GRN model. The uncertain data is generated in the way that in GRN, each generator or recognizer is linked to 2-4 tuple variables. For comparison, we also generate probabilistic relations with 10K, 20K, and 50K tuples respectively[3]. The inference queries are all about extracting the joint distribution of a set of random variables. The number of variables contained in each query varies from six to ten, and on average a query covers more than 60% range of the whole INDSEP. The queries are generated by following the *Pareto* pattern, which means that among all queries 80% queried variables come from a set containing 20% of the whole variables.

5.1 Parameter Tuning

Firstly, we illustrate our results for choosing appropriate node size, i.e. the maximum number of cliques contained in each INDSEP node. Note that our proposed approach is main-memory based. It seems as if the larger node size, the shorter INDSEP tree, and the better query performance.We test different node sizes on querying INDSEP on the 100K dataset. As illustrated by Figure 3, the performance does not keep increasing as the node size enlarges. The reason is due to the high computation cost of handling node potentials during query evaluation. This cost will increase exponentially when the node size enlarges and may totally counteract the benefits obtained from the reduction of tree height. Based on this observation, we set the node size as eight, the optimal point in Figure 3.

[3] By default, the dataset contains 100K tuples.

We further run a series of experiments to detect the optimal value of cache size, i.e. the maximum memory consumption permitted to store all the cached factors. From Fig. 4, we can see that when the cache size increases, the response time of both *SubQuery* and *FVEC* will drop in the initial phase, then increase a little bit and finally become stable. When cache size is small, a larger cache will accommodate more factors and help reduce the response time. But if the cache size is big enough to store most frequent results, the performance cannot be increased more. On the contrast, the response time may be longer since there are many infrequent results reserved in cache and more efforts are needed to search the cache. As implied by the results, we set the cache size as 200M for the following experiments. We also compare the cache hit ratio, i.e. percent of intermediate sub-queries which are skipped due to matched factors in the cache. We can observe from Fig.5 that the cache hit ratio of *FVEC* is much higher than that of *SubQuery* which is because the advantages of using smaller factors and express channels. Also, the ratios will be become stable for both methods when the cache size is big enough, which again explains why the performance will not increase constantly when enlarging cache sizes.

5.2 Performance Comparison

Overall Performance. Figure 6 shows the comparison of three approaches, i.e. INDSEP, *Subquery* and *FVEC* on the 100K data set. The x-axis shows the sequence number of queries. Basically a sequence is a group of 200 queries. The y-axis shows the average response time for each sequence. Clearly, the *FVEC* outperforms the other two approaches. The relative improvement with respect to INDSEP is more than 50 percents. The *Subquery* approach performs worse than *FVEC* but better than INDSEP. This result is coincident with our estimation. Our proposed approach will increase the cache hit ratio as discussed in Section 4 and it wins the best. The *Subquery* approach is based on the INDSEP and tries to utilize cached results for answering queries, so it defeats INDSEP.

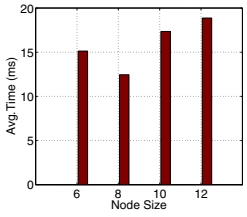


Fig. 3. Time vs. NodeSize

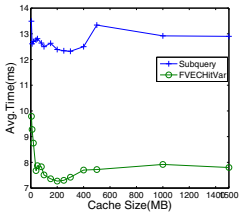


Fig. 4. Time vs. Cache

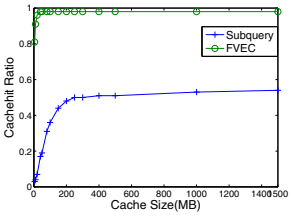


Fig. 5. HitRatio vs. Cache

In order to illustrate the fire up phase of collecting frequent variables, we partially enlarge Figure 6 and show performance of processing the first 1,000 queries in Figure 7. Each point of Figure 7 is an average of 20 runs. From this figure, we can see that in the beginning, the performance of the three approaches are roughly the same. During the processing of queries, the response time of *Subquery* and *FVEC* will decrease quickly, while the performance of INDSEP

has no obvious changes. This because both *Subquery* and *FVEC* will cache some results and use them for answering new queries. Hence their performance will become better when experiencing more queries.

Finally, we test the average response time of the three approaches on four datasets for processing 10,000 queries, with sizes of 10K, 20K, 50K and 100K respectively. Figure 8 illustrates the result. We can find that *FVEC* always performs the best among the three approaches, and the increasing rate is acceptable.

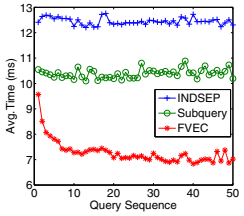


Fig. 6. Comparison of Overall Performance

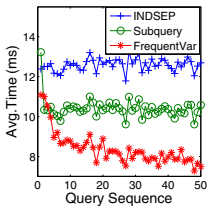


Fig. 7. Fire Up Phase of Figure 6

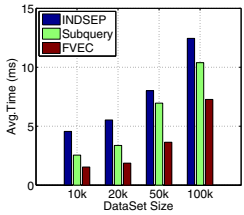


Fig. 8. Response Time vs. Data Set Size

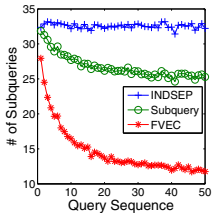


Fig. 9. # of Subqueries

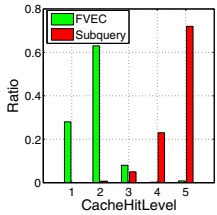


Fig. 10. Cache Hit Levels

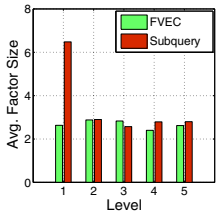


Fig. 11. Avg. Factor Size

Breakdown Analysis. We now illustrate why *FVEC* could perform the best through several breakdown analysis. Firstly, we want to mention that during a query evaluation the number of extended subqueries is one of the most crucial factors which affect the response time. We compare the number of required subqueries for the three approaches in Figure 9. Again *FVEC* performs the best. Recall that *FVEC* can reduce the number of subqueries by completely matching subqueries with cached result, and can skip many subqueries by the express channels based on Algorithm 3.

As discussed before, a subquery may be again broken into a set of children subqueries if it is issued on an non-leaf node. Hence on average a cache hitting in an upper level could save much more subqueries than hittings in lower levels. To illustrate this, we further compare the cache hit levels of the two caching approaches in Figure 10. Here a cache hit happens if and only if a subquery terminates by reusing some cached information. The y-axis of Figure 10 presents the percents of cache hits at each level. From Figure 10, we can see that more than 90% cache hittings of *FVEC* happens in the first two levels, while most cache hittings of *Subquery* appear in the fourth and fifth levels. That explains why *FVEC* performs much better than *Subquery*.

Finally, we compare the average size of factors, i.e. the number of variables in a factor. Note that the *FVEC* approach utilizes smaller factors to accommodate more cached items 3.2. This motivation is verified by the results in Figure 11. At the first level, the average factor size of *Subquery* is much larger than that of *FVEC*. We also find that this effect is not that evident in lower levels. Note that a sub-query evaluated in the first level would be decomposed into several smaller sub-queries to the second level. Hence the size of sub-queries in lower levels would be quite small (usually one or two) and this is why it is hard to see the difference in these levels. Remember that the first level would be the most important place to kill sub-queries. Thus *FVEC* could benefit a lot from the larger capacity in the first level.

6 Related Works

Management of Uncertain Data. During the past decade, along with the quick development of applications like mobile data management, sensor network, data streams and data integration, processing uncertain data attract many research interests from the database community. These works are mainly targeted on uncertain data models [2], probabilistic queries [4,11], uncertain data index [17,8], and uncertain data mining [16].

Querying Uncertain Data. Evaluating probabilistic queries is known to be #-complete [5]. Many solutions are proposed for improving the performance, including Monte-Carlo sampling [4], verification and refinement approaches [19], and uncertain data index [17] etc.. All of these works assume that there are no correlations or only locally correlations, i.e. each uncertain data is only correlated with several data items, and independent of others. Hence they are not applicable for the applications where uncertain data are highly correlated with each other.

Works on querying correlated uncertain data usually represent uncertain data by probabilistic graphical models (PGM) and regard this problem as performing inference on PGM. Sen et al. utilize the shared correlations among uncertain data to simplify the original PGM and reduce the query evaluation time [13]. In [8], the INDSEP structure is proposed for indexing correlated uncertain data, which is adopted by this work. In [12], probabilistic inference is transformed to SAT and existing solutions for SAT could be adopted for solving inference. None of these works consider the use of cached results for accelerating future query evaluation. The work most related to ours is [15]. Instead of storing intermediate results as [15] does, we cache the results of each frequent variable in order to increase the cache hit ratio. Also, we try to cut off unnecessary path from the evaluation path with the *express channel* technique.

7 Conclusion and Future Works

In this paper, we address the problem of answering probabilistic inference queries over large volume of uncertain data with high correlations. In order to solve this problem, we propose to cache the result of each frequent single variable in

INDSEP and design a series of novel techniques for efficient processing inference queries with cached information. According to our experimental results, our proposed approach is both effectiveness and efficient compared with the state-of-the-art works. About the future work, we plan to study how to further improve query performance with some approximates and/ or cloud computing techniques.

Acknowledgments. This work is funded by the National Science Foundation of China under Grant No. 61003086, and National Basic Research Program of China (973 Program) No. 2012CB316205.

References

1. The pareto principle, `http://en.wikipedia.org/wiki/Pareto_principle`
2. Chen, R., Mao, Y., Kiringa, I.: Grn model of probabilistic databases: construction, transition and querying. In: SIGMOD 2010, pp. 291–302 (2010)
3. Cheng, R., Chen, J., Mokbel, M.F., Chow, C.-Y.: Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In: ICDE 2008, pp. 973–982 (2008)
4. Cheng, R., Kalashnikov, D.V., Prabhakar, S.: Evaluating probabilistic queries over imprecise data. In: SIGMOD 2003, pp. 551–562 (2003)
5. Dalvi, N.N., Suciu, D.: Efficient query evaluation on probabilistic databases. In: VLDB 2004, pp. 864–875 (2004)
6. Das Sarma, A., Dong, X., Halevy, A.: Bootstrapping pay-as-you-go data integration systems. In: SIGMOD 2008, pp. 861–874 (2008)
7. Hwang, F.K., Richards, D.S.: Steiner tree problems. Networks 22(1), 55–89 (1992)
8. Kanagal, B., Deshpande, A.: Indexing correlated probabilistic databases. In: SIGMOD 2009, pp. 455–468 (2009)
9. Kanagal, B., Deshpande, A.: Lineage processing over correlated probabilistic databases. In: SIGMOD 2010, pp. 675–686 (2010)
10. Koller, D., Friedman, N.: Probabilistic Graphical Models Principles and Techniques. MIT Press, London (2009)
11. Lian, X., Chen, L.: Efficient query answering in probabilistic rdf graphs. In: SIGMOD 2011, pp. 157–168 (2011)
12. Sang, T., Bearne, P., Kautz, H.: Performing bayesian inference by weighted model counting. In: AAAI 2005, pp. 475–481. AAAI Press (2005)
13. Sen, P., Deshpande, A., Getoor, L.: Exploiting shared correlations in probabilistic databases. In: VLDB 2008 (2008)
14. Singh, S., Mayfield, C., Prabhakar, S., Shah, R., Hambrusch, S.E.: Indexing uncertain categorical data. In: ICDE 2007, pp. 616–625 (2007)
15. Song, S., Chen, L., Yu, J.X.: Answering frequent probabilistic inference queries in databases. IEEE Trans. on Knowledge and Data Engineering 23, 512–526 (2011)
16. Sun, L., Cheng, R., Cheung, D.W., Cheng, J.: Mining uncertain data with probabilistic guarantees. In: KDD 2010, pp. 273–282 (2010)
17. Tao, Y., Cheng, R., Xiao, X., Ngai, W.K., Kao, B., Prabhakar, S.: Indexing multi-dimensional uncertain data with arbitrary probability density functions. In: VLDB 2005, pp. 922–933. VLDB Endowment (2005)
18. Wang, D.Z., Franklin, M.J., Garofalakis, M., Hellerstein, J.M.: Querying probabilistic information extraction. In: PVLDB, vol. 3(1-2), pp. 1057–1067 (September 2010)
19. Zhang, W., Lin, X., Zhang, Y., Pei, J., Wang, W.: Threshold-based probabilistic top-k dominating queries. The VLDB Journal 19(2), 283–305 (2010)

Author Index