Jeffrey Xu Yu
Myoung Ho Kim
Rainer Unland (Eds.)

LNCS 6588

Database Systems for Advanced Applications

16th International Conference, DASFAA 2011
Hong Kong, China, April 2011
Proceedings, Part II

2 Part II

Springer

Jeffrey Xu Yu   Myoung Ho Kim
Rainer Unland (Eds.)

Database Systems for Advanced Applications

16th International Conference, DASFAA 2011
Hong Kong, China, April 22-25, 2011
Proceedings, Part II

Springer

Volume Editors

Jeffrey Xu Yu
The Chinese University of Hong Kong
Department of Systems Engineering and Engineering Management
Shatin, N.T., Hong Kong, China
E-mail: yu@se.cuhk.edu.hk

Myoung Ho Kim
Korea Advanced Institute of Science and Technology (KAIST)
Department of Computer Science
291 Daehak-ro (373-1 Guseong-don), Yuseong-gu, Daejeon 305-701, Korea
E-mail: mhkim@dbserver.kaist.ac.kr

Rainer Unland
University of Duisburg-Essen
Institute for Computer Science and Business Information Systems (ICB)
Schützenbahn 70, 45117 Essen, Germany
E-mail: rainer.unland@icb.uni-due.de

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-20151-6 e-ISBN 978-3-642-20152-3
DOI 10.1007/978-3-642-20152-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011923553

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

DASFAA is an annual international database conference, which showcases state-of-the-art R&D activities in database systems and their applications. It provides a forum for technical presentations and discussions among database researchers, developers and users from academia, business and industry. It is our great pleasure to present you the proceedings of the 16th International Conference on Database Systems for Advanced Applications (DASFAA 2011), which was held in Hong Kong, China, during April 22-25, 2011.

DASFAA 2011 received 225 research paper submissions from 32 countries / regions (based on the affiliation of the first author). After a thorough review process for each submission by the Program Committee and specialists recommended by Program Committee members, DASFAA 2011 accepted 53 full research papers and 12 short research papers (the acceptance rates were 24% and 5%, respectively). Other papers in this volume include four industrial papers selected by a committee chaired by Takahiro Hara (Osaka University), Tengjiao Wang (Peking University), and Xing Xie (Microsoft Research China), and eight demo papers selected by a committee chaired by Jidong Chen (EMC Research China), Lei Chen (The Hong Kong University of Science and Technology), and Kyoung-Gu Woo (Samsung Electronics).

This volume also includes two invited keynote papers, presented by leading experts in database research and advanced applications at DASFAA 2011, Josephine M. Cheng (IBM Research Almaden Lab) and Divy Agrawal (University of California at Santa Barbara), on the topics of "Smarter Planet: Empower People with Information Insights" and "Database Scalability, Elasticity, and Autonomy in the Could," respectively; one extended abstract for the DASFAA 2011 ten-year best paper on "What Have We Learnt from Deductive Object-Oriented Database Research?" by Mengchi Liu (Carleton University), Gillian Dobbie (University of Auckland), and Tok Wang Ling (National University of Singapore); three tutorial abstracts, selected by Tutorial Co-chairs Reynold Cheng (The University of Hong Kong), Ken Lee (University of Massachusetts Dartmouth), and Ee-Peng Lim (Singapore Management University), "Managing Social Image Tags: Methods and Applications" by Aixin Sun and Sourav S. Bhowmick, "Searching, Analyzing and Exploring Databases" by Yi Chen, Wei Wang and Ziyang Liu, and "Web Search and Browse Log Mining: Challenges, Methods, and Applications" by Daxin Jiang; and one panel abstract selected by Panel Co-chairs, Haibo Hu (Hong Kong Baptist University), Haixun Wang (Microsoft Research China), and Baihua Zheng (Singapore Management University). The conference program boasts conference proceedings that span two volumes in Springer's *Lecture Notes in Computer Science* series.

Beyond the main conference, six workshops, held in conjunction with DASFAA 2011, were selected by Workshop Co-chairs Jianliang Xu (Hong Kong

Baptist University), Ge Yu (Northeastern University), and Shuigeng Zhou (Fudan University). They are the First International Workshop on Graph-structured Data Bases (GDB 2011), the First International Workshop on Spatial Information Modeling, Management and Mining (SIM3), the International Workshop on Flash-Based Database Systems (FlashDB), the Second International Workshop on Social Networks and Social Media Mining on the Web (SNSMW), the First International Workshop on Data Management for Emerging Network Infrastructures (DaMEN), and the 4th International Workshop on Data Quality in Integration Systems (DQIS). The workshop papers are included in a separate volume of proceedings also published by Springer in its *Lecture Notes in Computer Science* series.

DASFAA 2011 was jointly organized by The Chinese University of Hong Kong, The Hong Kong University of Science and Technology, Hong Kong Baptist University, The University of Hong Kong, City University of Hong Kong, and The Hong Kong Polytechnic University. It received in-cooperation sponsorship from the China Computer Federation Database Technical Committee. We are grateful to the sponsors who contributed generously to making DASFAA 2011 successful. They are the Department of Systems Engineering and Engineering Management of The Chinese University of Hong Kong, Oracle, IBM, K.C. Wong Education Foundation, and Hong Kong Pei Hua Education Foundation.

The conference would not have been possible without the support of many colleagues. We would like to express our special thanks to Honorary Conference Co-chairs, Xingui He (Peking University), Shan Wang (Renmin University of China), and Kyu-Young Whang (KAIST) for their valuable advice on all aspects of organizing the conference. We thank Organizing Committee Chair Kam-Fai Wong (The Chinese University of Hong Kong), Publicity Co-chairs, Raymond Wong (The Hong Kong University of Science and Technology), Xiaochun Yang (Northeastern University), and Xiaofang Zhou (University of Queensland), Publication Chair Rainer Unland (University of Duisburg-Essen), Finance Chair Vincent Ng (The Hong Kong Polytechnic University), Local Arrangements Chair Hong-va Leong (The Hong Kong Polytechnic University), Sponsor Chair Joseph Ng (Hong Kong Baptist University), Best Award Committee Co-chairs Ming-Syan Chen (Academia Sinica, Taiwan and National Taiwan University) and Aoying Zhou (East China Normal University), and Demo Award Committee Co-chairs Ben Kao (The University of Hong Kong) and Lizhu Zhou (Tsinghua University). Our thanks go to all the committee members and other individuals involved in putting it all together, and all authors who submitted their papers to this conference.

July 2010

Dik Lun Lee
Wang-Chien Lee
Kamal Karlapalem
Jeffrey Xu Yu
Myoung Ho Kim

Organization

Honorary Conference Co-chairs

Xingui He	Peking University, China
Shan Wang	Renmin University of China, China
Kyu-Young Whang	Korea Advanced Institute of Science and Technology (KAIST), Korea

Conference General Co-chairs

Dik Lun Lee	The Hong Kong University of Science and Technology, China
Wang-Chien Lee	Penn State University, USA
Kamal Karlapalem	IIIT-Hyderabad, India

Program Committee Co-chairs

Jeffrey Xu Yu	The Chinese University of Hong Kong, China
Myoung Ho Kim	Korea Advanced Institute of Science and Technology (KAIST), Korea

Organizing Committee Chair

Kam-Fai Wong	The Chinese University of Hong Kong, China

Workshop Co-chairs

Jianliang Xu	Hong Kong Baptist University, China
Ge Yu	Northeastern University, China
Shuigeng Zhou	Fudan University, China

Industrial Co-chairs

Takahiro Hara	Osaka University, Japan
Tengjiao Wang	Peking University, China
Xing Xie	Microsoft Research China, China

Tutorial Co-chairs

Reynold Cheng	The University of Hong Kong, China
Ken Lee	University of Massachusetts Dartmouth, USA
Ee-Peng Lim	Singapore Management University, Singapore

Panel Co-chairs

Haibo Hu	Hong Kong Baptist University, China
Haixun Wang	Microsoft Research China, China
Baihua Zheng	Singapore Management University, Singapore

Demo Co-chairs

Jidong Chen	EMC Research China, China
Lei Chen	The Hong Kong University of Science and Technology, China
Kyoung-Gu Woo	Samsung Electronics, Korea

Publicity Co-chairs

Raymond Wong	The Hong Kong University of Science and Technology, China
Xiaochun Yang	Northeastern University, China
Xiaofang Zhou	University of Queensland, Australia

Local Arrangements Chair

Hong-va Leong	The Hong Kong Polytechnic University, China

Finance Chair

Vincent Ng	The Hong Kong Polytechnic University, China

Publication Chair

Rainer Unland	University of Duisburg-Essen, Germany

Web Chair

Hong Cheng	The Chinese University of Hong Kong, China

Demo Award Committee Co-chairs

Ben Kao	The University of Hong Kong, China
Lizhu Zhou	Tsinghua University, China

Best Paper Committee Co-chairs

Ming-Syan Chen	Academia Sinica, Taiwan and National Taiwan University, Taiwan
Aoying Zhou	East China Normal University, China

Steering Committee Liaison

Qing Li	City University of Hong Kong, China

Sponsor Chair

Joseph Ng	Hong Kong Baptist University, China

CCF DBTC Liaison

Xiaofeng Meng	Renmin University of China, China

DASFAA Awards Committee

Tok Wang Ling (Chair)	National University of Singapore, Singapore
Jianzhong Li	Harbin Institute of Technology, China
Krithi Ramamirtham	Indian Institute of Technology at Bombay, India
Kian-Lee Tan	National University Singapore, Singapore
Katsumi Tanaka	Kyoto University, Japan
Kyu-Young Whang	Korea Advanced Institute of Science and Technology (KAIST), Korea
Jeffrey Xu Yu	The Chinese University of Hong Kong, China

DASFAA Steering Committee

Katsumi Tanaka (Chair)	Kyoto University, Japan
Ramamohanarao Kotagiri (Vice Chair)	University of Melbourne, Australia
Kyu-Young Whang (Advisor)	Korea Advanced Institute of Science and Technology (KAIST), Korea
Yoshihiko Imai (Treasurer)	Matsushita Electric Industrial Co., Ltd., Japan
Kian Lee Tan (Secretary)	National University of Singapore (NUS), Singapore
Yoon Joon Lee	Korea Advanced Institute of Science and Technology (KAIST), Korea
Qing Li	City University of Hong Kong, China
Krithi Ramamritham	Indian Institute of Technology at Bombay, India
Ming-Syan Chen	National Taiwan University, Taiwan
Eui Kyeong Hong	Univerity of Seoul, Korea
Hiroyuki Kitagawa	University of Tsukuba, Japan
Li-Zhu Zhou	Tsinghua University, China
Jianzhong Li	Harbin Institute of Technology, China
BongHee Hong	Pusan National University, Korea

Program Committees

Research Track

Toshiyuki Amagasa	University of Tsukuba, Japan
Masayoshi Aritsugi	Kumamoto University, Japan
James Bailey	University of Melbourne, Australia
Ladjel Bellatreche	Poitiers University, France
Boualem Benatallah	University of New South Wales, Australia
Sourav S. Bhowmick	Nanyang Technological University, Singapore
Athman Bouguettaya	CSIRO, Australia
Chee Yong Chan	National University Singapore, Singapore
Jae Woo Chang	Chonbuk National University, Korea
Lei Chen	The Hong Kong University of Science and Technology, China
Ming-Syan Chen	National Taiwan University, Taiwan
Reynold Cheng	The University of Hong Kong, China
Hong Cheng	The Chinese University of Hong Kong, China
James Cheng	Nanyang Technological University, Singapore
Byron Choi	Hong Kong Baptist University, China
Yon Dohn Chung	Korea University, Korea
Gao Cong	Nanyang Technological University, Singapore
Bin Cui	Peking University, China
Alfredo Cuzzocrea	ICAR-CNR / University of Calabria, Italy
Gill Dobbie	University of Auckland, New Zealand
Xiaoyong Du	Renmin University of China, China
Jianhua Feng	Tsinghua University, China
Ling Feng	Tsinghua University, China
Sumit Ganguly	IIT Kanpur, India
Yunjun Gao	Zhejiang University, China
Vivek Gopalkrishnan	Nanyang Technological University, Singapore
Wook-Shin Han	Kyungpook National University, Korea
Takahiro Hara	Osaka University, Japan
Bingsheng He	Nanyang Technological University, Singapore
Wynne Hsu	National University Singapore, Singapore
Haibo Hu	Hong Kong Baptist University, China
Seung-won Hwang	POSTECH, Korea
Yoshiharu Ishikawa	Nagoya University, Japan
Mizuho Iwaihara	Waseda University, Japan
Adam Jatowt	Kyoto University, Japan,
Ruoming Jin	Kent State University, USA
Jaewoo Kang	Korea University, Korea

Vincent S. Tseng	National Cheng Kung University, Taiwan
Vasilis Vassalos	Athens University of Economics and Business, Greece
John Wang	Griffith University, Australia
Jianyong Wang	Tsinghua University, China
Guoren Wang	Northeastern University, China
Wei Wang	University of New South Wales, Australia
Raymond Wong	The Hong Kong University of Science and Technology, China
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, Korea
Ge Yu	Northeastern University, China
Aidong Zhang	University of Buffalo, SUNY, USA
Rui Zhang	University of Melbourne, Australia
Yanchun Zhang	Victoria University, Australia
Baihua Zheng	Singapore Management University, Singapore
Aoying Zhou	East China Normal University, China
Xiaofang Zhou	University of Queensland, Australia

Industrial Track

Wolf-Tilo Balke	University of Hannover, Germany
Edward Chang	Google, China and University of California Santa Barbara, USA
Bin Cui	Peking University, China
Dimitrios Georgakopoulos	CSIRO, Australia
Seung-Won Hwang	POSTECH, Korea
Marek Kowalkiewicz	SAP, Australia
Sanjay Kumar Madria	Missouri University of Science and Technology, USA
Mukesh Mohania	IBM Research India, India
Makoto Onizuka	NTT Corporation, Japan
Jilei Tian	Nokia Research China, China
Masashi Tsuchida	Hitach, Ltd., Japan
Jianyong Wang	Tsinghua University, China
Wei Wang	Fudan University, China
Yu Zheng	Microsoft Research Asia, China

Demo Track

Ilaria Bartolini	University of Bologna, Italy
Bin Cui	Peking University, China
Heasoo Hwang	Samsung Electronics, Korea

Jin-ho Kim	Kangwon National University, Korea
Changkyu Kim	Intel Labs, USA
Guoqiong Liao	Jiangxi University of Finance and Economics, China
Hongrae Lee	Google Research, USA
Jiaheng Lu	Renmin University of China, China
Peng Wang	Fudan University, China
Feng Yaokai	Kyushu University, Japan

External Reviewers

Eunus Ali
Parvin Asadzadeh
He Bai
Moshe Barukh
Seyed-Mehdi-Reza Beheshti
Arnab Bhattacharya
Nick Bozovic
Xin Cao
Wing Kwan Chan
Lijun Chang
Muhammad Aamir Cheema
Jinchuan Chen
Jiefeng Cheng
Taewon Cho
Jaehoon Choi
Shumo Chu
Ke Deng
Wei Feng
Shen Ge
Haris Georgiadis
Kazuo Goda
Jian Gong
Reza Hemayati
He Hu
Hai Huang
Jun Huang
Stéphane Jean
Bin Jiang
Lili Jiang
Yifan Jin
Akimitsu Kanzaki
Hideyuki Kawashima
Selma Khouri
Henning Koehler
Neila Ben Lakhal
Dong-Ho Lee
Injoon Lee
Jongwuk Lee
Kyubum Lee
Mu-Woong Lee
Sanghoon Lee
Sunwon Lee
Guoliang Li
Jianxin Li
Jing Li
Jiang Li
Lin Li
Xian Li
Xiang Li
Xiang Lian
Wenxin Liang
Lian Liu
Wenting Liu
Xingjie Liu
Jiangang Ma
Hossein Maserrat
Takeshi Mishima
Surya Nepal
Bo Ning
Wee Siong Ng
Junho Oh
Sai Tung On
Jin-woo Park
Yu Peng
Jianzhong Qi
Kun Qian
Miao Qiao
Hongda Ren
Jong-Won Roh
Seung Ryu
Sherif Sakr
Shuo Shang
Jie Shao
Mohamed A. Sharaf
Gao Shen
Wei Shen
Zhitao Shen
Wanita Sherchan
Reza Sherkat
Lei Shi
Chihwan Song
Ha-Joo Song
Shaoxu Song
Yehia Taher
Takayuki Tamura
Guanting Tang
Yuan Tian
Guoping Wang
Puwei Wang
Yi Wang
Yousuke Watanabe
Ingo Weber
Chuan Xiao
Hairuo Xie
Kexin Xie
Lin Xin
Jiajie Xu
Zhiqiang Xu
Kefeng Xuan
Yuan Xue

Table of Contents – Part II

Table of Contents – Part I

XML

Efficient Histogram-Based Similarity Search in Ultra-High Dimensional Space

Jiajun Liu[1], Zi Huang[1,2], Heng Tao Shen[1], and Xiaofang Zhou[1,2]

[1] School of ITEE, University of Queensland, Australia
[2] Queensland Research Laboratory, National ICT Australia
{jiajun,huang,shenht,zxf}@itee.uq.edu.au

Abstract. Recent development in image content analysis has shown that the dimensionality of an image feature can reach thousands or more for satisfactory results in some applications such as face recognition. Although high-dimensional indexing has been extensively studied in database literature, most existing methods are tested for feature spaces with less than hundreds of dimensions and their performance degrades quickly as dimensionality increases. Given the huge popularity of histogram features in representing image content, in this papers we propose a novel indexing structure for efficient histogram based similarity search in ultra-high dimensional space which is also sparse. Observing that all possible histogram values in a domain form a finite set of discrete states, we leverage the time and space efficiency of inverted file. Our new structure, named two-tier inverted file, indexes the data space in two levels, where the first level represents the list of occurring states for each individual dimension, and the second level represents the list of occurring images for each state. In the query process, candidates can be quickly identified with a simple weighted state-voting scheme before their actual distances to the query are computed. To further enrich the discriminative power of inverted file, an effective state expansion method is also introduced by taking neighbor dimensions' information into consideration. Our extensive experimental results on real-life face datasets with 15,488 dimensional histogram features demonstrate the high accuracy and the great performance improvement of our proposal over existing methods.

1 Introduction

Image retrieval based on content similarity has been put in spotlight for the past few decades [8]. Histogram constructed by counting the number of pixels from an image in each of a fixed list of bins is one of the most popular features used in many applications [11], where each image is represented by a high-dimensional histogram feature vector. Among many distance functions proposed for histogram comparison, the histogram intersection and the Euclidean distance are widely used due to their high efficiency and effectiveness [16]. The dimensionality of an image histogram is typically about tens or hundreds. Recently, driven by the significant need of real-life applications such as identity verification, video surveillance, automated border control, crime scene footage analysis, and so on, more sophisticated image features are required to reduce false alarm rate under various conditions and noises in face recognition. For example, Local Binary Patterns (LBP) [1] and recently proposed Local Derivative Patterns (LDP) [19] are well known and proved to be very effective. According to the particular settings, an 88×88 face image can generate a 15,488-dimensional histogram feature or more. A major challenge that prevents face recognition from being widely applied on large-scale or real-time applications is the vast computational cost when faces are compared based on the above ultra-high dimensional histogram features. Obviously, without any database support, few applications can actually bear such high computational cost rooted from the ultra-high dimensionality. Although many high-dimensional indexing methods have been introduced in database literature [4], performance results on feature spaces over thousands dimensions are hardly found.

In this paper, we frame our work in the context of histogram-based similarity search. Our main idea comes from the following observations on histogram features. Firstly, given the known image resolution and the fixed number of bins, all the possible values in a histogram feature vector form a finite set of discrete values. Therefore, a value in an arbitrary dimension has a finite number of possible states. Secondly, many dimensional values could be zeros since features may not be evenly distributed, especially in the ultra-high dimensional space. Our LDP feature dataset extracted from standard face datasets show that more than 30% dimensional values are zeros. The particular characteristics of discrete state and high sparsity in the high-dimensional feature space have not been previously exploited to tackle the similarity search problem.

Motivated by the above observations and the high efficiency of inverted file in text retrieval where data are also discrete and sparse, we propose a novel two-tier inverted file structure to index the ultra-high dimensional histograms for efficient similarity search, where a dimension for a state (and a state for an image) is analogous to a word for a document. To be more specific, we make the following contributions.

– We model histogram feature values in a finite set of discrete states, based on which a two-tier inverted file structure is proposed to leverage the high efficiency of inverted file. In the new structure, the first tier represents the list of states for each individual dimension, and the second tier represents the list of images for each state. Meanwhile, techniques are also employed to remove those indiscriminate state lists for further performance improvement and space reduction.
– We propose a fast query processing algorithm based on a simple weighted state-voting scheme. Only those images with highest voting scores with respect to the query are remained for the actual similarity computations in the original space.
– We propose an effective state expansion method for each dimensional value of a histogram by taking its local information into consideration. Each dimension of an image is assigned with a larger number of possible states by

comparing itself with its left and right neighbor dimensions. The purpose of this is to further increase the discriminative power of inverted file.
- We conduct an extensive performance study on real-life face datasets with up to 15488-dimensional histogram features. The results demonstrate the high accuracy and the significant performance improvement of our proposal over existing methods.

The rest of the paper is organized as follows. We review some related work in Section 2. Section 3 provides some preliminary information on the ultra-high dimensional histogram feature and the related similarity measure. The proposed two-tier inverted file indexing structure is introduced in Section 4, which followed by the query processing in Section 5. Extensive experiments regarding effectiveness, efficiency and scalability has been conducted and analyzed in Section 6. Finally we conclude our work in Section 7.

2 Related Work

Towards effective database supports for high-dimensional similarity search, a lot of research efforts have been witnessed in database community. Various categories of high-dimensional indexing methods have been proposed to tackle the "curse of dimensionality".

Tree structures have achieved notable success in managing low-dimensional feature vectors, from early R-tree, kd-tree and their variants, to M-tree [6], A-tree [13] and many other trees [4]. The key idea is to prune tree branches as much as possible based on the established bounding distances so that the number of accessed feature vectors (or points) can be reduced significantly. However, their performance rapidly degrades as feature dimensionality increases, and eventually most of them are outperformed by sequence scan when dimensionality reaches high tens due to the massive overlap among different branches [18].

Apart from exact search, approximate search has recently drawn much attention. The aim is to gain performance improvement by sacrificing minor accuracy. One typical approach is Locality Sensitive Hashing (LSH) [9]. The basic idea is to use a family of locality sensitive hash functions composed of linear projection over random directions in the feature space. The intuition behind is that for at lease one of the hash functions, nearby objects have high probability of being hashed into the same state. Improvements to LSH have been made continuingly during the past decade, regarding its accuracy, time efficiency and space efficiency by improving the hashing distribution [7], by enforcing its projection method [3], and by combining efficient tree structures [17]. However, how to generate effective hash functions for thousands of dimensions or higher is unclear.

One-dimensional indexing using the efficient B^+-tree is another category, such as iDistance [10]. It partitions data points into clusters and indexes all the points by their distances to their respective reference points using a single B^+-tree. Its efficiency comes from the localized distances to corresponding reference points and B^+-tree. Its performance is further improved by finding the optimal reference points which can maximize the performance of B^+-tree [14]. Nonetheless, single dimensional distance values become very indistinguishable for ultra-high dimensional feature vectors.

Another direction is to reduce the number of dimensions of the high-dimensional data before indexing it. The data is first transformed into a much lower-dimensional space using dimensionality reduction methods and then an index is built on it to further facilitate the retrieval [15,5]. The key idea is to transform data from a high-dimensional space to a lower dimensional space without losing much information. However, it is mostly infeasible to reduce the dimensionality from thousands or higher to tens without losing critical information.

Instead of reducing dimensionality, some methods aim to approximate data, such as VA-file [18]. It approximates each dimension with a small number of bits, by dividing the data space into 2^b rectangular cells where b denotes a user specified number of bits. The VA-File allocates a unique bit-string of length b for each cell, and approximates data points that fall into a cell by that bit-string. The VA-File itself is simply an array of these compact, geometric approximations. Query process is performed by scanning the entire approximation file and excluding points from the actual distance computation based on the lower and upper bounds established from these approximations. This approach is insensitive to the dimensionality and thus able to outperform sequential scan if a small number of candidates are finally accessed. However, the improvement ratio is rather limited since every single dimension needs to be encoded. Some refined approaches based on VA-file have also been proposed to handle datasets of different distributions [2,12].

It is clear that most existing works are not deemed to index ultra-high dimensional feature vectors for efficient similarity search. VA-file is likely the most feasible one to have comparable performance with sequential scan in ultra-high dimensional spaces since its is dimension independent. Interestingly, inverted file has been a very effective solution for indexing large-scale text databases with extremely high dimensionality [20]. In this paper, by analyzing the histogram intrinsic properties, we introduce a novel and compact indexing structure called two-tier inverted file to index ultra-high dimensional histograms. The fact that dimensional values in histogram are discrete and finite motivates us to utilize the efficiency of inverted file for histogram-based similarity search.

3 Preliminaries

In this section, we provide the information on how ultra-high dimensional feature vectors can be generated from images and explain the observations which motivate our design. For easy illustration, we take the recently proposed Local Derivative Pattern (LDP) feature [19] in face recognition as the example.

3.1 LDP Histogram

Face recognition is a very important topic in pattern recognition. Given a query face image, it aims at finding the most similar face from a face database. Due to the strong requirement in high accuracy, face images are usually represented by very sophisticated features in order to capture the face in very detailed levels. Given a certain similarity measure, face recognition can be considered as the nearest neighbor search problem in ultra-high dimensional spaces.

An effective face feature or descriptor is one of the key issues for a well-designed face recognition system. The feature should be of high ability to discriminate between classes, has low intra-class variance, and can be easily computed. Local Binary Pattern (LBP) is a simple yet very efficient texture descriptor which labels the pixels of an image by thresholding the neighborhood of each pixel with the value of the center pixel and considers the result as a binary number [1]. Due to its discriminative power and computational simplicity, LBP has become a popular approach in face recognition. As an extension to LBP, the high-order Local Derivative Pattern (LDP) has been recently proposed as a more robust face descriptor, which significantly outperforms LBP for face identification and face verification under various conditions [19]. Next, we provide a brief review of these two descriptors.

Derived from a general definition of texture in a local neighborhood, LBP is defined as a grayscale invariant texture measure and is a useful tool to model texture images. The original LBP operator labels the pixels of an image by thresholding the 3×3 neighborhood of each pixel with the value of the central pixel and concatenating the results binomially to form a 8-bit binary sequence for each pixel. LBP encodes the binary result of the first-order derivative among local neighbors.

As an extension to LBP, LDP encodes the higher-order derivative information which contains more detailed discriminative features. The second order LDP descriptor labels the pixels of an image by encoding the first-order local derivative direction variations and concatenating the results as a 32-bit binary sequence for each pixel. A histogram can then be constructed based on the LDP descriptor to represent an image.

To get more precise image representation, an image is typically divided into small blocks, on which more accurate histogram is calculated. For example, given an image with resolution of 88×88, it can be divided into a number of 484 4×4 sized blocks. In [19], each block is represented by 4 local 8-dimensional histograms along four different directions, where each dimension represents the number of pixels in the bin. The final LDP histogram of the image is generated by concatenating all the local histograms of each block, i.e., 484 32-dimensional histogram. Its overall dimensionality is the number of blocks multiplied by the local histogram size, i.e., $484 \times 32 = 15,488$. Theoretically, the maximum dimensionality could reach $88 \times 88 \times 32$ when each pixel is regarded as a block. This LDP histogram is claimed as a robust face descriptor which is insensitive to rotation, translation and scaling of images.

For histogram features, the number of bins for an image (or block) is always predetermined. Since the number of pixels in the image (or block) is also known, the value along each dimension in the histogram is an integer within the range from 0 to the maximum number of pixels in the image (or block). For example, in LDP histogram, if the block size is 4×4, then the value in the histogram can only be an integer in the range of [0,16]. Clearly, the first observation is that the histogram values are discrete and from a finite set of numbers, where each number is regarded as a state. Note that values could also be float if some normalization is applied. However, normalization does not change the nature of being discrete and finite. At the same time, many dimensions may have zero value in ultra-high dimensional histograms. Motivated by the discrete and sparse characteristics, we utilize the efficiency of inverted file to achieve efficient similarity search in ultra-high dimensional histogram feature spaces, as to be presented in Section 4.

3.2 Histogram Similarity Measures

Many similarity measures have been proposed for histogram matching. The histogram intersection is a widely used similarity measure. Given a pair of LDP histograms H and S with D dimensions, the histogram intersection is defined as

$$Sim(H,S) = \sum_{i=1}^{D} min(H_i, S_i) \tag{1}$$

In the metric defined above, the intersection is incremented by the number of pixels which are common between the target image and the query image along each dimension. Its computational complexity is very low. It is used to calculate the similarity for nearest neighbor identification and has shown very good accuracy for face recognition [19]. Another popular measure is the classical Euclidean distance which has also been used in many other feature spaces. Although other similarity measures can be used, in this paper we will test both the histogram intersection and the Euclidean distance to see their effects on the performance.

4 Two-Tier Inverted File

As introduced in Section 3, the face image feature, LDP histogram, is usually in ultra-high dimensionality (i.e., more than ten thousands). Given its extremely high dimensionality, it is not practical to perform the full similarity computations for all database images. In this section, we present a novel two-tier inverted file for indexing ultra-high dimensional histograms, based on the discrete and sparse characteristics of histograms.

Inverted file has been used widely in text databases for its high efficiency [20] in both time and space, where the text dimensionality (i.e., the number of words) is usually very high and the word-document matrix is very sparse since a document only contains a small subset of the word dictionary. However, it has not been well investigated in the low-level visual feature databases. Here we exploit the discrete and finite nature of histograms and design a two-tier inverted file structure for efficient similarity search in ultra-high dimensional space.

In the traditional text-based inverted file, each word points to a list of documents which contain the word. Naive adoption of inverted file to histograms is to regard each dimension as a word pointing to a list of images whose values (or states) on the dimension is not zero. By doing this, all zero entries in histograms are removed. However, histograms also have some different features from text datasets. Firstly, word-document matrix is far more sparser than histograms, since the word dictionary size is typically much larger than the average number of words in documents. This leads to a rather long images list for each dimension. Secondly, all values in histograms are distributed in a predetermined state range from 0 to the maximum number of pixels allowed in a bin. This inspires us to create another level of inverted file for each dimension by regarding each state on the dimension as a word pointing to a list of images which have the same state. Therefore, a long image list can be further partitioned into multiple shorter lists for quicker identification. Thirdly, comparing with the number of images, the number of states is often much smaller. For example, LDP histograms generated from 4×4 sized blocks have 16 possible states only, without considering the zero state. To further improve the discriminative power of inverted file, we design an effective state expansion method, before we look at the overall structure of the two-tier inverted file.

4.1 State Expansion

Given that the number of states in histograms is relatively small, we aim to expand the number of states to balance the state list size and the image list size for better performance. The basic idea is to expand the original state on a dimension of an image into multiple states which are more specific and discriminative. The difficulty for state expansion lies in the preservation of the original state information. We propose to take the local neighbor information into account for expansion.

To illustrate the idea, we assume an image is divided into $4{\times}4$ sized blocks in LDP histogram. The number of pixels in each bin ranges from 0 to B, where B is the block size, i.e., B=16. Thus the number of possible states for a dimension is B+1. Since all zero entries in histograms are not indexed in inverted file, we have B states left to consider.

To expand the number of states, we consider the relationship between the states of i^{th} dimension with its neighbor dimensions, i.e., its left and right neighbors. Comparing the values of i^{th} dimension and $(i-1)^{th}$ dimension for an image, there exist three relationships, including " $<$ ", " $>$ " and " $=$ ". Similarly, the comparison between values of i^{th} dimension and $(i+1)^{th}$ dimension have three relationships as well. Therefore, by considering the relationship with its left and right neighbor dimensions, a single i^{th} dimension's state can be expanded into 3×3 possible states.

Given an image histogram $H = (h_1, h_2, ..., h_D)$, it can be transformed to the expanded feature $H' = (h'_1, h'_2, ..., h'_D)$, where h'_i is calculated by the following formula

$$h'_i = h_i \times 9 + t_1 \times 3 + t_2 \times 1, \qquad (2)$$

$$t_1 = \begin{cases} 0 & \text{if } h_i < h_{i-1} \text{ or } i = 1 \\ 1 & \text{if } h_i = h_{i-1} \\ 2 & \text{if } h_i > h_{i-1} \end{cases} \qquad t_2 = \begin{cases} 0 & \text{if } h_i < h_{i+1} \text{ or } i = D \\ 1 & \text{if } h_i = h_{i+1} \\ 2 & \text{if } h_i > h_{i+1} \end{cases}$$

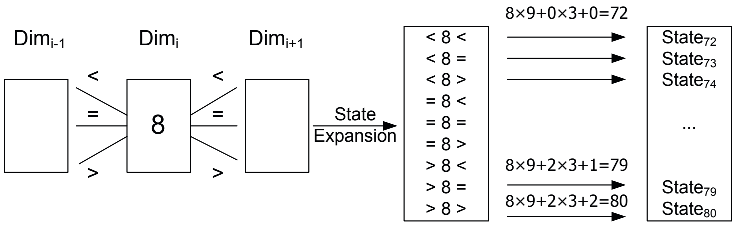


Fig. 1. An example for state expansion

Basically, each state is stretched into an interval which contains nine new states based on the local relationship with its left and right neighbors. The term $h_i \times 9$ is used to distinguish original states into different intervals, and the term $t_1 \times 3 + t_2 \times 1$ is used to differentiate nine local relationships within an interval. Figure 1 depicts an example where i^{th} dimension has an original state of 8 and is expanded into nine new states. Since a dimension of an image originally have B possible states without considering zero, the total number of states after expansion becomes $3 \times 3 \times B$. For example, when B is 16, the total number of possible states for a dimension is expanded to $3 \times 3 \times 16 = 144$.

State expansion is performed on the original feature for each dimension of every histogram. The i^{th} dimension of j^{th} image, H_{ij}, is assigned with the new value of $H'_{ij} = H_{ij} \times 9 + t_1 \times 3 + t_2 \times 1$. Note that more local relationships can be exploited if more neighbor dimensions are considered. If the number of histograms is overwhelming, more neighbors like $(i-2)^{th}$ dimension and $(i+2)^{th}$ dimension can be used. For our data scale used in experiments, expansion on two neighbor dimensions has shown very satisfactory performance.

State expansion achieves a more detailed description of histogram by considering neighbor information. It plays an important role in accelerating the search process, by distributing fixed number of images into a larger number of states. The average number of images on each state is hence reduced, making query process more efficient, as to be explained in Section 5.

4.2 Index Construction

Given an image dataset consisting of N histograms in D dimensionality, Figure 2 illustrates the general process of constructing the two-tier inverted file.

Given an image represented as a histogram, $H = (h_1, h_2, ..., h_D)$, it is firstly transformed to $H' = (h'_1, h'_2, ..., h'_D)$ by taking state expansion. In H', each dimension of an image is associated with a new state value, which is generated by considering the relationships with its neighbor dimensions.

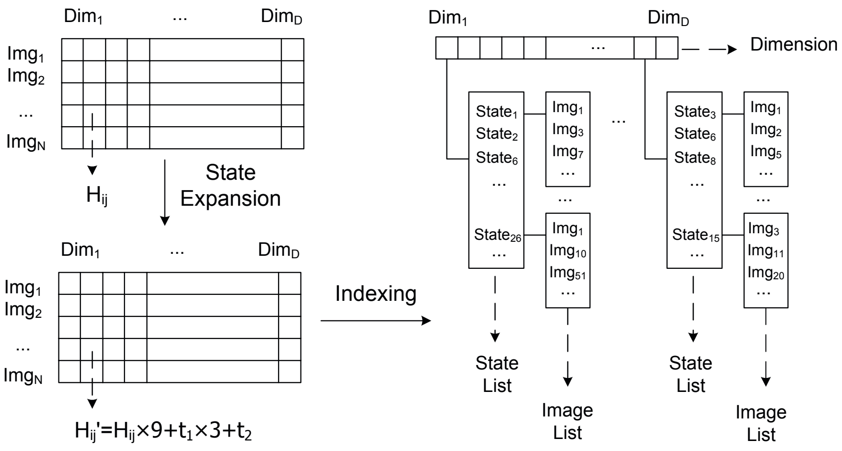


Fig. 2. Construction of the two-tier inverted file indexing structure

Motivated by the discrete nature of values (or states) in histogram, we propose a two-tier inverted file to effectively index H' and handle the sparsity issue. The right sub figure in Figure 2 shows an overview of the indexing structure. In the first tier, an inverted list of states is constructed for each individual dimension among all images. This tier indicates what states exist on a dimension. If the number of states is small while the number of images is large, all dimensions will basically have a complete list of states. By effective state expansion, each dimension is likely to have a different list of states. In the second tier, an inverted list of images is built for each state existing in a dimension. Denote the number of states as M. The maximum number of image lists is $M \times D$. Given the relatively small block size, M is usually much smaller than D and N. With state expansion, M can be enlarged so that a better balance between the state lists and the image lists can be obtained.

Like the traditional inverted file for documents, the new two-tier inverted file for histograms does not index the original zero states. Meanwhile, one question rises here. Is it necessary to keep those states with very long image lists? In text retrieval, we understand that frequent words are removed since they are not discriminative. Here we adopt the same assumption which is also verified by our experiments. A threshold on the length of image list, ε, is used to determine if an image list should be removed from the indexing structure. Only the states (and their image lists) who have less number of images than this threshold are kept in the two-tier inverted file. Note that rare states are also retained in our structure since some applications such as face recognition only search for the nearest neighbor. Rare information could be helpful in identifying the most similar result.

Thus, the original histograms in the ultra-high dimensional space are finally indexed by the compact two-tier inverted file. Given an image query, it can be efficiently processed in the structure via a simple weighted state-voting scheme, as to be explained next.

```
Input: $Q[]$, $D$, $B$, $L[][]$
Output: Nearest Neighbor
 1. for ($i = 1; i < D; i++$) do
 2.     $q'_i \leftarrow$ ComputeState($q_i$);
 3. end for
 4. $Candidates = \{\}$
 5. for ($i = 1; i <= D; i++$) do
 6.     $Candidates+ \leftarrow L[i].q'_i$;
 7. end for
 8. $Candidates[k] \leftarrow$ WeightedStateVoting($Candidates+$);
 9. $NearestNeighbor \leftarrow$ ComputeNearestNeighbor($Candidates[k]$);
10. return $NearestNeighbor$;
```

Algorithm 1. The Query Processing Algorithm

5 Query Processing

Based on the two-tier inverted file, query processing is efficient and straightforward. We use a simple weighted state-voting scheme to quickly rank all the images and only a small set of candidates will be selected for full similarity computations in the original space. Algorithm 1 outlines the query process.

Given a query image histogram, $Q = (q_1, ..., q_D)$, we firstly transform it to $Q' = (q'_1, ..., q'_D)$ by applying state expansion (lines 1-3). ComputeState() is the method to compute the new state value for q_i, based for Equation 2. Next, the two-tier inverted file, denoted as $L[][]$, is searched. For i^{th} dimension, its corresponding image list which has the same state value as q'_i is quickly retrieved via allocating i^{th} dimension in the first tier and then q'_i in the second tier in the structure. After all dimensions are searched, a set of candidates is generated (lines 5-7). Each image in the candidate set shares one or more common states with the query image in certain dimensions. Here a weighted state-voting method is employed to compute the amount of contribution to the final similarity between the query and a candidate. The frequency of an image in the candidate set reflects the number of common states it shares with the query image. Note that candidates are generated by matching states on each dimension. However, different matched states contribute differently to the final similarity when the histogram intersection is used. Matched states with larger values contribute more to the final similarity. Therefore, state values have to be considered when candidates are ranked. Since only expanded states are indexed in the data structure, the matched state q'_i has to be transformed back to the original state q_i, according to Equation 2. WeightedStateVoting() is the method to rank all the candidates (line 8). When the histogram intersection is applied, the ranking score for each candidate is computed based on $\sum \bar{q_i}$, where $\bar{q_i}$ is the value of matched state between Q and a candidate. Top-k candidates are returned for the actual histogram intersection computations to find the nearest neighbor to Q (line 9). For example, assume that $D = 3$, $Q = (1, 3, 2)$, $L[1].1 = \{img_1, img_2\}$, $L[2].3 = \{img_2, img_4\}$, and $L[3].2 = \{img_4\}$ without state expansion. The weighted state-voting result is for img_1, img_2 and img_3 is 1, 1+3, and 3+2 respectively. By setting $k = 2$, img_3 and img_2 are returned as the final candidates to compute their histogram intersection similarities with respect to the query to find the nearest neighbor. When the Euclidean distance is applied, two matched dimensions have distance of 0. In this case, by setting the same weight for all matched states, top-k most frequently occurring candidates in the candidate set are returned for the actual Euclidean distance computations. This is reasonable since more matched dimensions lead to a smaller overall distance with a higher probability. This algorithm also has the flexibility in returning more nearest neighbors which will affect the setting of k. The effect of k will be examined in the experiments.

It is noticeable that the above query processing algorithm only returns approximate results. There are three factors which affect the accuracy. Firstly, state expansion may cause information loss. In state expansion, one original state may be expanded into different new states if the neighbor relationships are different. Since the algorithm selects the candidates based on matching states and their voting scores, two different new states with the same original state cannot be matched. It is expected that this loss becomes relatively less significant as dimensionality increases and the encoded local information can compensate the loss to certain extent. Secondly, removal of frequent states in the two-tier inverted file may also affect the accuracy, as studied in text retrieval. Thirdly, since only top-k candidates are selected for final similarity computations, the correctness of the results cannot be guaranteed. In the next section, we extensively study the effects of the above three factors. Results on real-life ultra-high dimensional histograms show very promising performance with negligible sacrifice on quality, despite of the correctness guarantee problem.

6 Experiments

6.1 Set Up

We have collected 40,000 face images from different sources, including various standard face databases[1], such as FERET, PIE Database, CMU, the Yale Face Database, etc., and faces extracted from different digital albums. Both database images and query images are represented by 15,488 dimensional LDP histograms which have shown very good accuracy in face recognition [19]. All experiments are conducted on a desktop with 2.93GHz Intel CPU and 8GB RAM.

To measure the search effectiveness of our proposal, we use the standard precision, where the ground-truth for a query is the search results from sequential scan in the original space. In face recognition, typically only the top one result is needed. Thus, we only evaluate results on the nearest neighbor search, although more nearest neighbors can also be returned.

[1] `http://www.face-rec.org/databases/`

Before the performance comparison with existing methods, We first conduct experiments on FERET to test our method. FERET[2] is a standard face dataset consisting of 3,541 gray-level face images representing the faces of 1,196 people under various conditions (i.e., variant facial expression, illumination, and ageing). The dataset is divided into five categories, fa (i.e., frontal images), fb (i.e., facial expression variations), fc (i.e., under various illumination conditions), dup1 (i.e., face images taken later in time between one minute to 1,031 days) and dup2 (i.e., a subset of dup1; face images taken at least after 18 months). FERET is widely used as a standard dataset for evaluation of face recognition related algorithms and systems. For effectiveness and efficiency evaluation, categories fb, fc, dup1 and dup2 of FERET are considered as four query image sets.

Since we have 2 parameters in our scheme, ε, and k, representing the threshold on the image list for a state, and the number of candidates for actual similarity computations respectively, both of them need to be tested. By default, state expansion is applied, ε is 5% of the size of the image dataset, and $k = 20$. Due to the space limit, we only report the results by applying histogram intersection similarity measure. Euclidean distance actually shows very similar results.

6.2 Effect of ε

In our two-tier indexing structure, we assume that the length of an image list for a state reflects its discrimination power. If the number of images for a state is greater than ε, this image list is considered as non-discriminative and removed from the index structure. We test different values of ε including 5%, 7%, 15% and 20% of the total image size to observe its effect on effectiveness.

Observed from Figure 3(a), a larger ε leads to a better precision for all query sets since more image lists are maintained in the data structure. However, the overall precision under various settings is promising, i.e., all higher than 98%. The precision difference among different settings is not significant and nearly identical. The search time for different ε values is shown in Figure 3(b). As ε goes up, the indexing structure is larger and more images are likely to be accessed and compared. Therefore, for different sets of queries, they show the same trend. The search time drops quickly as ε goes up. Since ε has greater impact on efficiency, by default, we set $\varepsilon = 5\%$.

6.3 Effect of k

In query processing, voting scheme is applied to generate a set of candidates for further similarity calculation. Different settings on k lead to different precisions. Figure 3(c) shows the results of $k = 5, 10, 20$ and 50 for the nearest neighbor search. Precision reaches almost 100% when $k \geq 20$. The reason is that the more candidates we include, the higher probability that the correct results are finally accessed and returned. The search time increases as k increases since more candidates are compared, as shown in Figure 3(d). $k = 20$ is a reasonable default value for both precision and efficiency consideration.

[2] http://www.itl.nist.gov/iad/humanid/feret/feret_master.html

6.4 Effect of State Expansion

A key factor that contributes to the high effectiveness and efficiency of our method is that we expand the space of effective states and consequently encode more local distinctiveness into each of the states. In this subsection, we also test the effect of state expansion.

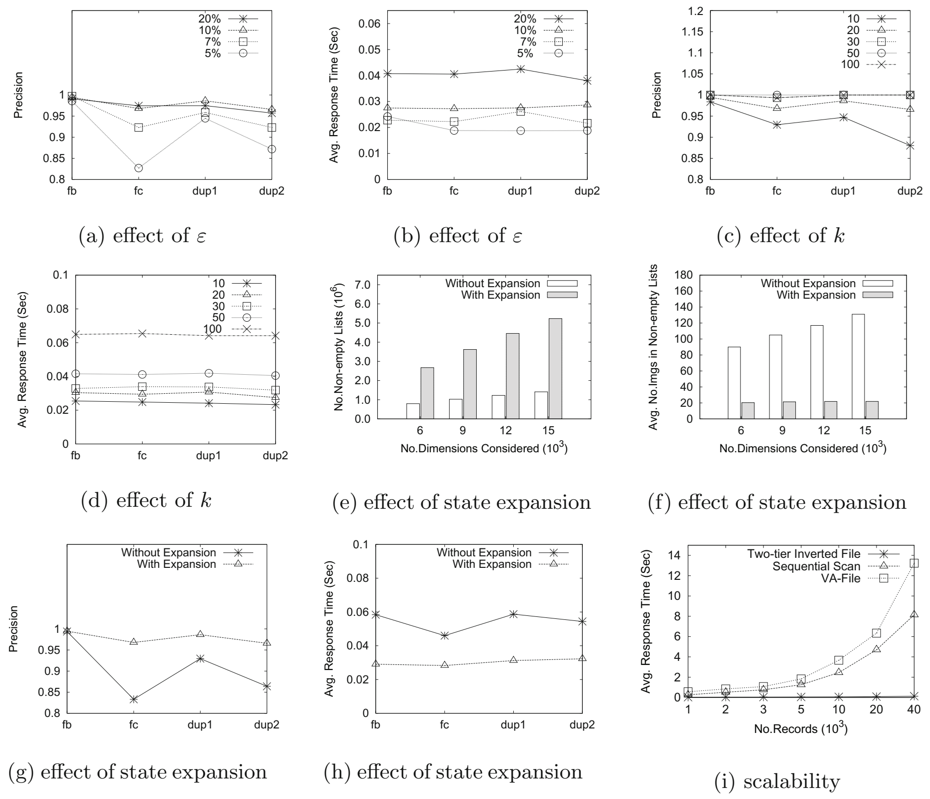
Figure 3(e) and 3(f) depict the selectivity improvement made by state expansion. The total number of image lists and the average number of images in each list are reported. Clearly by expanding the number of states, the average number of images for each state is greatly reduced. The average number of images per list is about 30 after state expansion.

The effect of state expansion on precision and efficiency is reflected in Figure 3(g) and 3(h) respectively. Very surprising, with our state expansion, the accuracy is even higher, especially for fc, dup1 and dup2 query sets. This is a bit hard to explain since state expansion may possibly miss some results if local neighbor relationships among their dimensions are different. Without state expansion, information loss mainly comes from the removal of long image lists. Because images lists without state expansion are expected to be much longer than those with state expansion (as depicted in 3(f)), there is a risk to remove more lists from the indexing structure. As a result, more information could be lost if the states are not expanded. Undoubtedly, state expansion improves the search efficiency (as shown in Figure 3(h)) since fewer and shorter lists are searched. In short, state expansion achieves improvement in both precision and efficiency.

6.5 Performance Comparison

In the last experiment, we conduct a comparison study on efficiency, with sequential scan, VA-file and iDistance. Sequential scan is included because that, in the ultra-high dimensional space, its performance is even better than most of existing indexing methods due to the “curse of dimensionality”. VA-file, on the other hand, is less sensitive to the dimensionality than most tree-based index structures. Two bits are used for each dimension in VA-file since only 17 original states exist in LDP histogram. Note that the above index structures return complete results, while two-tier inverted file is an approximate search scheme which offers superior efficiency with negligible precision loss. In order to compare the two-tier inverted file with other approximate searchs, we also adopt iDistance as an approximate search scheme. Ten clusters are used in iDistance and its search radius is increased until the scheme reaches the same precision as the two-tier inverted file. The whole dataset of 40,000 face images is used for this experiment.

Figure 3(i) shows the average search time for a single query with four different methods. We observe that our method outperforms all other three methods by more than two orders of magnitude. The search time for all methods increases as the data size increases. However, our method grows very slow as the data size increases from 1000 to 40,000 (up to 0.1 second), while the search time for sequential scan, VA-File and iDistance increase dramatically. Notice that VA-File is outperformed by sequential scan. There are two main reasons. Firstly, LDP



(a) effect of ε (b) effect of ε (c) effect of k

(d) effect of k (e) effect of state expansion (f) effect of state expansion

(g) effect of state expansion (h) effect of state expansion (i) scalability

Fig. 3. Effectiveness, efficiency and scalability

histograms are highly skew in different localities. Secondly, it is difficult for VA-File to have a tight bound for the histogram intersection similarity to achieve efficient pruning. IDistance shows slightly better performance than sequential scan. However, its search time still climbs quickly, because the distance between any point and the reference point tends to be very close when dimensionality is extremely high, making a minor increase on search radius include an excessive number of data points to process. This experiment proves that by utilizing the high efficiency of inverted file, our method is able to achieve real-time retrieval in ultra-high dimensional histogram spaces.

7 Conclusion

In this paper, we present a two-tier inverted file indexing method for efficient histogram-based similarity search in ultra-high dimensional spaces. It indexes the sparse and ultra-high dimensional histograms with a compact structure which utilizes the high efficiency of inverted file, by observing that histogram values are actually discrete and from a finite value set. An effective state expansion method is designed to further discriminate the data for an efficient and effective feature representation. An extensive study on a large-scale face image dataset confirms the novelty and practical significance of the proposal.

References

1. Ahonen, T., Hadid, A., Pietikäinen, M.: Face description with local binary patterns: Application to face recognition. IEEE TPAMI 28(12), 2037–2041 (2006)
2. An, J., Chen, H., Furuse, K., Ohbo, N.: Cva file: an index structure for high-dimensional datasets. Knowl. Inf. Syst. 7(3), 337–357 (2005)
3. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. CACM 51(1), 117–122 (2008)
4. 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)
5. Chakrabarti, K., Mehrotra, S.: Local dimensionality reduction: A new approach to indexing high dimensional spaces. In: VLDB, pp. 89–100 (2000)
6. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An efficient access method for similarity search in metric spaces. In: VLDB, pp. 426–435 (1997)
7. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Symposium on Computational Geometry, pp. 253–262 (2004)
8. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image retrieval: Ideas, influences, and trends of the new age. ACM Comput. Surv. 40(2) (2008)
9. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB, pp. 518–529 (1999)
10. 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 TODS 30(2), 364–397 (2005)
11. Lew, M.S., Sebe, N., Djeraba, C., Jain, R.: Content-based multimedia information retrieval: State of the art and challenges. ACM TOMCCAP 2(1), 1–19 (2006)
12. Lu, H., Ooi, B.C., Shen, H.T., Xue, X.: Hierarchical indexing structure for efficient similarity search in video retrieval. IEEE TKDE 18(11), 1544–1559 (2006)
13. Sakurai, Y., Yoshikawa, M., Uemura, S., Kojima, H.: The A-tree: An index structure for high-dimensional spaces using relative approximation. In: VLDB, pp. 516–526 (2000)
14. Shen, H.T., Ooi, B.C., Zhou, X., Huang, Z.: Towards effective indexing for very large video sequence database. In: SIGMOD, pp. 730–741 (2005)
15. Shen, H.T., Zhou, X., Zhou, A.: An adaptive and dynamic dimensionality reduction method for high-dimensional indexing. VLDB Journal 16(2), 219–234 (2007)
16. Swain, M.J., Ballard, D.H.: Color indexing. IJCV 7(1), 11–32 (1991)
17. Tao, Y., Yi, K., Sheng, C., Kalnis, P.: Quality and efficiency in high dimensional nearest neighbor search. In: SIGMOD, pp. 563–576 (2009)
18. 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)
19. Zhang, B., Gao, Y., Zhao, S., Liu, J.: Local derivative pattern versus local binary pattern: face recognition with high-order local pattern descriptor. IEEE TIP 19(2), 533–544 (2010)
20. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Comput. Surv. 38(2) (2006)

A Retrieval Strategy Using the Integrated Knowledge of Similarity and Associations

Yong-Bin Kang[1], Shonali Krishnaswamy[1], and Arkady Zaslavsky[2]

[1] Faculty of Information Technology,
Monash University, Australia
{yongbin.kang,shonali.krishnaswamy}@monash.edu
[2] Department of Computer Science and Electrical Engineering,
Luleå University of Technology, Sweden
arkady.zaslavsky@ltu.se

Abstract. Retrieval is often considered the most important task in Case-Based Reasoning (CBR), since it lays the foundation for overall performance of CBR systems. In CBR, a typical retrieval strategy is realized through similarity knowledge encoded in similarity measures. This strategy is often called similarity-based retrieval (SBR). This paper proposes and validates that association analysis techniques can be used to improve SBR. We propose a retrieval strategy USIMSCAR that performs the retrieval task by integrating similarity and association knowledge. We show its reliability, in comparison with several retrieval methods implementing SBR, using datasets from UCI ML Repository.

1 Introduction

Retrieval is a very important task in CBR, since it lays the foundation for overall performance of CBR systems [1]. The goal of this task is to retrieve *useful* cases that can be successfully reused to solve a new problem. If retrieved cases are not useful, CBR systems will not eventually produce any good solution for the new problem. For the retrieval task, CBR systems are typically reliant on a specific strategy that exploits *similarity knowledge*. This strategy is thus often called *similarity-based retrieval* (SBR) [2]. In SBR, similarity knowledge represents a heuristic for approximating the usefulness of stored cases [3]. This knowledge is usually encoded in *similarity measures*. By using similarity knowledge, SBR retrieves useful cases, ranked by their similarities to a new problem. Then, solutions, of the top ranked cases, are used to solve the new problem.

However, the main limitation of SBR is that it cannot *guarantee* that cases, retrieved through similarity knowledge, are sufficiently useful to solve a new problem. This limitation is rooted in the fact that SBR only considers the *problem space*−a set of problems−in a given case base, when formalizing similarity knowledge. This leads to trouble. The reason is that determining the usefulness of stored cases cannot be completed without considering how known problems are actually *associated* with specific known solutions. SBR thus cannot retrieve those useful cases, whose solutions have stronger associations with the problems of certain cases similar to the new problem.

The goal of this work is to improve SBR by incorporating *association analysis* techniques into retrieval in CBR. For this purpose, we propose a new retrieval strategy USIMSCAR that exploits both similarity knowledge and association knowledge. The aim of *association knowledge* is to represent implicit and potentially useful *associations* (dependencies), between problems and solutions, observed in stored cases. More precisely, this knowledge models a set of highly correlated attribute-value pairs, of problems and solutions, shared by a large number of stored cases. To formalize this knowledge we use *association rule mining* techniques. The key idea of USIMSCAR is the exploitation of association knowledge, ignored in SBR, in conjunction with similarity knowledge, to provide a more complete strategy for retrieval in CBR.

This paper is organized as follows. In Section 2, we provide an overview of SBR, and the main problem of SBR. In Section 3, we introduce a well-known principle that formalizes similarity knowledge, and describe association analysis techniques used for formalizing association knowledge. In Section 4, we present our association knowledge formalism, and the USIMSCAR algorithm. In Section 5, we evaluate USIMSCAR, using 6 datasets found from UCI ML Repository[1], in comparison with 5 retrieval methods implementing SBR. In Section 6, we review related work. In Section 7, we finally conclude this paper with future work.

2 Similarity-Based Retrieval and Its Main Problem

Similarity-based retrieval (SBR) is typically implemented through the technique using a derivative of the *nearest neighbor* algorithm [4,1]. This technique is called *k-nearest neighbor retrieval* or simply k-NN [1]. The idea of k-NN is that, to solve a new problem, useful cases are determined using its k most similar cases (i.e. nearest neighbors). Here, similarity is used to represent a heuristic for estimating such cases. Thus, similarity is the most important aspect in k-NN.

Let a case base $\mathcal{D}$ be a set of cases. These cases (including a new problem q) are described by m (numeric and discrete) attributes $A_1, ..., A_m$. Assume that any numeric attributes have been normalized to the range [0,1]; and each case is labeled with a solution label $y \in Y$. Our aim here is to assign an appropriate solution label to q. For this purpose, k-NN first determines the nearest neighbors (i.e. similar cases) of q by using a distance metric. The standard for this metric is the Euclidean distance [5]. For each case $c \in \mathcal{D}$, the Euclidean distance $DIST(q, c)$, between q and c, is represented as

$$DIST(q,c) = \sqrt{\sum_{i=1}^{m} dist(q_i, c_i)^2}, \quad dist(q_i, c_i) = \begin{cases} |q_i - c_i|, & \text{if } A_i \text{ is numeric,} \\ 0, & \text{if } A_i \text{ is discrete \& } q_i = c_i, \\ 1, & \text{otherwise,} \end{cases} \tag{1}$$

where q_i and c_i denote the values of the A_i of q and c, respectively, and $dist(q_i, c_i)$ represents their distance.

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html

The distance metric $DIST(q, c)$ has the merit that it allows knowledge to be brought to bear on the assessment of similarity—the nearer two objects are, the more similar they are. In the rest of this paper, we thus consider similarity only to find the nearest neighbors of q. Once the neighbors are selected, there are various ways for determining a solution of q. The simplest approach is to choose the majority solution among the neighbors, called *majority voting* [6].

Over the years, researchers have widely studied k-NN to improve its performance in terms of accuracy. For example, its sensitiveness to k is overcome by determining a best k through a learning technique such as *cross-validation* [5]. Feature selection [7] is a good technique that determines a subset of relevant features (attributes) among the original features of stored cases. Feature weighting [8] is also a useful technique, in which each feature (attribute) is multiplied by a weight. The weight is usually determined by considering the ability of the feature in distinguishing solution labels. In this work, we categorize k-NN and its extensions, integrated using the above enhancements, into representative techniques (or models) implementing SBR.

Problem Statement. We now present a main limitation of SBR. To illustrate, we choose an extension of k-NN as a representative model implementing SBR. Assume that this model is made by integrating k-NN and feature weighting, and denoted as $\mathcal{Z}$. Consider a medical diagnosis scenario[2], where 5 patient cases are stored in a case base $\mathcal{D}$ (See Table 1a). Each case consists of 5 symptoms (attributes) $A_1, ..., A_5$ and the corresponding diagnosed disease (solution).

Table 1. A patient case base and similarity results

(a) A patient case base

Patients	Local Pain	Other Pain	Fever	Appetite Loss	Age	Diagnosis
p_1	right flank	vomit	38.6	yes	10	appendicitis
p_2	right flank	vomit	38.7	yes	11	appendicitis
p_3	right flank	vomit	38.8	yes	13	appendicitis
p_4	right flank	sickness	37.5	yes	35	gastritis
p_5	epigastrium	nausea	36.8	no	20	stitch
q	right flank	nausea	37.8	yes	14	?
Weight	0.91	0.78	0.60	0.40	0.20	

(b) Similarity results

$SIM(q, p_1) = 0.631$
$SIM(q, p_2) = 0.623$
$SIM(q, p_3) = 0.618$
$SIM(q, p_4) = \mathbf{0.637}$
$SIM(q, p_5) = 0.420$

Our aim is to diagnose the correct disease of a new patient q. $\mathcal{Z}$ achieves this goal by identifying the k most similar cases to q. To find them, it selects cases whose symptoms are similar to q, using a similarity metric. Assume that we use the following metric[3] to measure the similarity between q and each case $p_k \in \mathcal{D}$:

$$SIM(q, p_k) = \frac{\sum_{i=1}^{m} w_i \cdot sim(q_i, p_{ki})}{\sum_{i=1}^{m} w_i}, \quad sim(q_i, p_{ki}) = \begin{cases} 1 - \frac{|q_i - p_{ki}|}{A_i^{\max} - A_i^{\min}}, & \text{if } A_i \text{ is numeric,} \\ 1, & \text{if } A_i \text{ is discrete } \& \ q_i = p_{ki}, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

[2] This scenario is a simple modification of the scenario found in the work [9].

[3] This similarity metric is the same one used in the work [9].

where w_i[4] is a weight of A_i; q_i and p_{ki} are the values of the A_i of, q and p_k, respectively; m is 5; $sim(q_i, p_{ki})$ is the similarity between q_i and p_{ki}; and $A_i^{\max}$ and $A_i^{\min}$ are the "max" and "min" values of A_i, respectively, in all the cases.

Using the metric $SIM(q, p_k)$, assume that $\mathcal{Z}$ chooses the single most similar case to q. As seen in Table 1b, we then choose the most similar case to q as p_4. This means that a diagnosis for q is chosen as 'gastritis'. However, this turns out to be wrong, since q is actually identified to suffer from 'appendicitis'[5]. This wrong diagnosis may lead to a serious error for q. If the disease 'appendicitis', that q really has, is not treated correctly, q's health may be endangered.

The scenario clearly shows that SBR has a significant limitation, rooted in the fact that SBR is strongly based on similarity knowledge. To overcome the limitation, a potential idea is to exploit the knowledge of how certain attribute-value pairs of known problems are *associated* with specific known solutions in $\mathcal{D}$. With the above scenario, we may obtain the following knowledge: the attribute-value pairs of p_1, p_2 and p_3 have a strong *association* with 'appendicitis', and those of p_4 with 'gastritis'. The strength of the former association, denoted as A_1, may be higher than that of the latter association, denoted as A_2. Because A_1 is supported by three cases, while A_2 is supported by only one case. If these associations were to be appropriately quantified and integrated with the similarity results in Table 1b, a diagnosis for q may be more accurately determined. This is the fundamental idea underlying USIMSCAR.

3 Similarity Knowledge and Association Analysis

We now present a similarity knowledge formalism, and the association analysis techniques used for building association knowledge. Before this, we first give the case model that is the basis for formalizing these two kinds of knowledge.

To represent cases, CBR systems often adopt well-known knowledge representation formalisms, such as attribute-value pairs or object-oriented representation [1]. We adopt the attribute-value pairs representation, due to its flexibility and popularity [3]. An *attribute-value pair* is represented as the form of (A_i, a_i), where A_i is an attribute (or feature[6]) and a_i is a value of A_i. Let a case be characterized by $m+1$ attributes $A_1, ..., A_m, A_{m+1}$ in a domain T. Let $\mathcal{P}$ be the problem space, a set of potential problems in T, where each problem $x \in \mathcal{P}$ is characterized by $A_1, ..., A_m$. Let $\mathcal{S}$ be the solutions space, a set of potential solutions in T, where each solution $s \in \mathcal{S}$ is characterized by A_{m+1}. We call A_{m+1} *solution-attribute*. A *case* is then defined as a pair (x, s_x), where x is a problem, $x = \{a_1, ..., a_m\} \in \mathcal{P}$, and s_x is a solution of x, $s_x = a_{m+1} \in \mathcal{S}$. For the sake of simplicity, we assume that x is associated with a unique solution s_x. In Section 7, we remark that USIMSCAR can be extended to the case, where a problem is described by more complex structures, and a problem is associated with more than one solution.

[4] The weight is borrowed from the work [9] and assigned by the domain expert.

[5] This fact is cited from the work [9].

[6] To simplify the presentation, we do not distinguish between terms attribute and feature.

3.1 Similarity Knowledge

Similarity knowledge is referred to as the knowledge encoded in similarity measures. SBR often uses a principle that models the similarity measures, suitable for the attribute-value pairs representation. It is *local-global principle* that decomposes an entire similarity computation by *local similarities* for individual attributes and a *global similarity* that aggregates these local similarities [3]. An accurate definition of local similarities relies on attribute types. An example of a similarity measure, based on this principle, is seen in Equation 2: "SIM" is a global similarity measure, and "sim" includes three local similarity measures.

3.2 Association Analysis

From the CBR viewpoint, *association rule mining* [10] is concerned about mining a set of highly correlated "attribute-value pairs of problems" and "solutions", shared by a large number of stored cases.

To formalize association knowledge, we build named *soft-matching class association rules* (scars) by using association rule mining techniques. A scar is a *class association rule* (car) [11] whose *antecedent* and *consequent* are generated by applying the *soft-matching criterion* [12]. A car is a special form of an *association rule.* Hence, we give an overview of association rules, cars, and the soft-matching criterion, involved in the formalization of scars.

• *association rules* [10]: Let $\mathcal{D}$ be a set of cases. Each case $T \in \mathcal{D}$ is characterized by attributes $A_1, ..., A_m, A_{m+1}$, where the problem is characterized by $A_1, ..., A_m$, and the solution by A_{m+1}. We call a pair $(A_i, a_i)_{1 \leq i \leq m+1}$ an *item.* Let I be a set of items. A set $X \subseteq I$, with $k = |X|$, is called a k-itemset or simply an itemset. We say that a case $T \in \mathcal{D}$ *supports* an itemset $X \subseteq I$, if $X \subseteq T$ holds. An *association rule* has two parts, *antecedent* and *consequent*, and denoted as $X \rightarrow Y$. Here, X is an itemset in the antecedent and Y is an itemset in the consequent, and $X \cap Y = \emptyset$ holds. The fraction of cases that support an itemset X in $\mathcal{D}$ is called the *support* of X, $supp(X) = |\{T \in \mathcal{D} | X \subseteq T\}|/|\mathcal{D}|$. The *support* of $X \rightarrow Y$ is defined as the probability that both X and Y occur together in a case T, $supp(X \rightarrow Y) = supp(X \cup Y)$. The *confidence* of $X \rightarrow Y$ is defined as $conf(X \rightarrow Y) = supp(X \cup Y)/supp(X)$. We say that an itemset X is *frequent*, if $supp(X) \geq$ minsupp (a user-specified minimum support). Apriori [10] is a representative algorithm widely used for association rule mining.

• *class association rules* (cars) [11]: A special subset of association rules, whose consequents are restricted to a single target, is called cars. From the CBR perspective, the solution-attribute (A_{m+1}) can become the target. We call a pair $(A_i, a_i)_{1 \leq i \leq m}$ an *item*, and call a pair (A_{m+1}, a_{m+1}) a *s-item.* Let I be a set of items, and SI be a set of s-items. A car is then an implication of the form $X \rightarrow s$, where X is an itemset $X \subseteq I$ and $s \in SI$ is a s-item.

• *soft-matching criterion* [12]: To discover frequent itemsets F, traditional association rule mining algorithms (e.g. Apriori) consider only itemsets that *exactly* match F. However, when treating attribute values, semantically related to each other, these algorithms may perform poorly. Because they ignore *semantic* *relevance* between those values. For example, they cannot find a rule like "80% of the customers, who buy milk-related (e.g. cheese) products and eggs-related (e.g. mayonnaise) products, also buy bread." To address this issue, the SoftApriori algorithm is proposed by [12]. SoftApriori uses *soft-matching* criterion, in which frequent itemsets are found by similarity assessment between itemsets.

4 Modeling Association Knowledge and USIMSCAR

We now propose an approach to formalizing association knowledge used in USIMSCAR. As mentioned above, this knowledge is encoded as scars, generated from stored cases. We then present the algorithm of USIMSCAR.

4.1 Soft-Matching Class Association Rules (SCARS)

A scar is a car whose antecedent and consequent are made by applying the soft-matching criterion. Our aim for building association knowledge is to encode the special knowledge of "how attribute-value pairs of known problems are actually *associated* with specific known solutions." Thus, it needs to be noted that we consider only cars representation, since it is suited to this objective.

A scar r, $X \rightarrow s$, reveals that a problem p is likely to be associated with a solution s, if p's attribute-value pairs are similar to an itemset X. The likelihood is quantified by r's *interestingness*. Interestingness measures are very useful to evaluate the quality of association rules [13]. For the measures, the support and confidence criteria are often used. On some occasions, a combination of them is used. Often, a rationale for doing so is to define a single optimal interestingness by leveraging the correlations between them. One example is the *Laplace* measure [13]. Below we describe it in detail. We first provide the definition of scars, and scars mining, following the definitions of terms in Section 3.2.

• *scars*: Let SM be an $m \times m$ similarity matrix, where m is the total number of items in I. Let $sim(x, y)$ be a similarity, between two items $x, y \in I$, driven from SM. We say that an item x is *similar* to an item y ($x \sim y$), iff $sim(x, y) \geq$ minsim (the user-specified minimum similarity). Let $SIM(X, Y)$ be a similarity between two itemsets $X, Y \in I$, $SIM(X, Y) = \sum_{x,y} sim(x, y)/|X|$, where $x, y \in X$ are two items characterized by the same attribute. We say that an itemset (or a case) Y *soft-support* of an itemset X ($X \subseteq_{soft} Y$), iff $SIM(X, Y) \geq$ minsim. The *soft-supporting-sum* of X regarding $\mathcal{D}$ is defined as $softSuppSum(X) = \sum_{t \in T} SIM(X, t)$, for each case $T \in \mathcal{D}$ satisfying $X \subseteq_{soft} T$. The *soft-support* of X regarding $\mathcal{D}$ is defined as $softSupp(X)$ $= softSuppSum(X)/|\mathcal{D}|$. The *soft-support* of a scar $X \rightarrow s$ is defined as the fraction of cases, in $\mathcal{D}$, that soft-support X and are described by the s-item s, $softSupp(X \rightarrow s) = softSupp(X \cup s)$. The *soft-confidence* of this rule is defined as $softConf(X \rightarrow s) = softSupp(X \rightarrow s)/softSupp(X)$. A *ruleitem* is of the form $\langle X, s \rangle$ and basically represents a rule $X \rightarrow s$. A scar is an implication of the form $X \rightarrow s$, whose soft-support is greater than minsupp. The definition of our soft-support differs from the one used in SoftApriori. In SoftApriori, the soft-support of each itemset is calculated by summing the number of *occurrences* of all similar itemsets. For example, the soft-support, of an 1-itemset $x \in I$, is computed as "$softSupp(x) = \sum_{y \in I} sim_{01}(x, y) \cdot supp(y)$", where $sim_{01}(x, y)$ is a *binary* function which is 1, if $x \sim y$, and 0, otherwise; and $supp(y)$ is the support of 1-itemset $y \in I$. Unfortunately, $softSupp(x)$ cannot reflect the different degrees of the similarities between x and all $y \in I$. In contrast, our definition replaces $sim_{01}(x, y)$ by $SIM(x, y)$ so that it can reflect such degrees.

• scars *mining*: The key operation of scars mining is to find all "ruleitems" that have soft-supports greater than minsupp. We call such ruleitems *frequent* ruleitems. For all the ruleitems that have the same set of items in the antecedent, one with the highest interestingness is chosen as *possible rule* (PR). To measure the interestingness, we choose a measure that combines soft-support and soft-confidence such that they are monotonically related (i.e. positively correlated). Thus, we choose the *Laplace* measure [13]. Given a scar r, this measure is defined as $Laplace(r) = \frac{N \cdot softSupp(r)+1}{N \cdot softSupp(r)/softConf(r)+2}$, where N is the total number of cases in $\mathcal{D}$. Since N is a constant, it is easy to see that this measure is monotonically related to soft-support and soft-confidence. If $Laplace(r)$ is greater than a user-specified minimum interestingness, called min-interesting, we say that r is *accurate*. A candidate set of scars consists of all PRs that are both "frequent" and "accurate". To select optimal scars, we finally ignore a scar $X \rightarrow s$ in the set, where $|X|$ is less than a user-specified minimum itemset size, named minitemsize.

4.2 The USIMSCAR Algorithm

USIMSCAR is designed to find potentially useful *objects* that can be used to solve a new problem by exploiting similarity and association knowledge. Each of these objects can be either a case or a scar. Given a new problem q, the goal of USIMSCAR is to generate a *retrieval result set* (RR) that holds those objects.

Let $\mathcal{D}$ be a set of cases; $prSCARS$ be the set of scars to be generated; and SM be the same similarity matrix used in scars mining. We now present the USIMSCAR algorithm $\mathcal{M}$ in the following:

(1) $\mathcal{M}$ retrieves the k most similar cases to q in $\mathcal{D}$, and stores them into a set RC. Assume that $SIM(q, c)$ is the function used for measuring the similarity between q and a case c in $\mathcal{D}$. This function can be defined using the global-local principle. The local similarities, for individual attributes of q and c, are computed using SM.

(2) $\mathcal{M}$ retrieves the k most similar scars to q in $prSCARS$. An important question raised here is how to determine the similarity $SIM(q, r)$ between q and a scar r. Its answer lies in our choice of cars representation for scars mining. This implies that scars have the *identical* structure to all cases in $\mathcal{D}$. To illustrate, assume a case c is simply characterized by attributes A_1 and A_2. So, c is formed as $c = (a_1, a_2)$, where $a_1 \in A_1$ is a problem and $a_2 \in A_2$ is a solution of a_1. Using c, we can generate a scar r, $\{(A_1, a_1)\} \rightarrow (A_2, a_2)$. Note that the values a_1 and a_2, in the antecedent and consequent of r, correspond to the problem a_1 and the solution a_2 of c, respectively. This choice allows $\mathcal{M}$ to compute $SIM(q, r)$ using

the same similarity measure used in the step (1). The retrieved rules are stored into a set RS.

(3) For each case $c \in RC$, $\mathcal{M}$ selects a specific scar $r_c \in prSCARS$ that meets the following condition: a scar $r \in pcSCARS$ is chosen as r_c, if it has the highest interestingness, $Laplace(r)$, among specific scars in $pcSCARS$. These scars must *cover*[7] c and also their solutions are equal to the solution of c. Then, $\mathcal{M}$ computes a *combined score*, for q and c, along with r_c by integrating two factors: $SIM(q, c)$, computed by using similarity knowledge, and $Laplace(r_c)$, calculated by using association knowledge. We denote this score as $cs(q, c)$, defined by $cs(q, c) = SIM(q, c) \cdot Laplace(r_c)$. If r_c is more than one, for example $r_c = \{r_{c1}, ..., r_{cm}\}$, $\mathcal{M}$ uses the average of $Laplace(r_{c1})$, ..., $Laplace(r_{cm})$. If there is no r_c for c, we use the given min-interesting[8], instead of $Laplace(r_c)$. Our combination scheme aims to enhance the significance of $SIM(q, c)$ by weighting the interestingness of r_c. Then, a *universe object*[9] is created. It has two fields. The *instance* field stores c, and the *cs* field holds $cs(q, c)$. This object is added to a set UR.

(4) For each scar $r \in RS$, $\mathcal{M}$ computes a combined score of r regarding q. That is, $cs(q, r) = SIM(q, r) \cdot Laplace(r)$. A universe object is then created, whose *instance* field stores r and *cs* field holds $cs(q, r)$. It is also added to the UR. This combination aims to enhance the significance of r's interestingness by weighting it with r's similarity to q.

(5) $\mathcal{M}$ finally produces the set RR that is a subset of UR through the function $getRR(UR)$. Before we explain this function, we first illustrate how the above four steps (1)-(4) perform using an example, to ease of the readability of $\mathcal{M}$. Then, we give the description of $getRR(UR)$.

An Example. Consider again the patient cases in Table 1a. Using these cases, we can obtain four scars shown in Table 2. To generate the scars in the above table, we used the similarity knowledge encoded in the similarity measure "SIM" in Equation 2. Recall that $\mathcal{Z}$[10] retrieved p_4 as the most useful case to q, and its diagnosis 'gastritis' was determined to be the diagnosis of q. However, this led to trouble, since q suffered from 'appendicitis', not 'gastritis'.

Table 2. The scars generated: A_1 = 'Local Pain', A_2 = 'Other Pain', A_3 = 'Fever', A_4 = 'Appetite Loss', A_5 = 'Age' and A_6 = 'Diagnosis'

ID	Antecedent	Consequent	*Laplace*	Covered by
r_1	$\{(A_1$,right flank),$(A_2$,vomit),$(A_3$,38.6),$(A_4$,yes),$(A_5$,13)$\} \rightarrow$	$(A_6$,appendicitis)	0.922	p_1, p_2, p_3
r_2	$\{(A_1$,right flank),$(A_2$,vomit),$(A_3$,38.7),$(A_4$,yes),$(A_5$,10)$\} \rightarrow$	$(A_6$,appendicitis)	0.922	p_1, p_2, p_3
r_3	$\{(A_1$,right flank),A_2,vomit),$(A_3$,38.8),$(A_4$,yes),$(A_5$,10)$\} \rightarrow$	$(A_6$,appendicitis)	0.922	p_1, p_2, p_3
r_4	$\{(A_1$,right flank),$(A_2$,sickness),$(A_3$,37.5),$(A_4$,yes),$(A_5$,35)$\} \rightarrow$	$(A_6$,gastritis)	0.775	p_4

USIMSCAR can overcome this trouble. It takes the following steps (assume k=2): (1) It generates the 2 most similar cases to q by using SIM. Thus, RC =

[7] We say that a scar r *covers* a case c, iff r is generated being soft-supported by c.

[8] This is mentioned in "scars mining" in Section 4.1.

[9] We refer to a universe object as a generic object that can encapsulate any case and scar and also have any attributes.

[10] $\mathcal{Z}$ was used as a representative model of SBR in Section 2.

$\{p_4, p_1\}$ (See Table 1b). (2) It generates the 2 most similar scars to q by also using SIM. Thus, $RS = \{r_1, r_4\}$ with $SIM(q, r_1) = 0.640$ and $SIM(q, r_4) = 0.637$. (3) For each case $c \in RC$, the r_c is determined. With this example, for p_4, r_{p_4} is selected as r_4, and, for p_1, r_{p_1} as r_1, r_2 and r_3. Then, the combined scores $cs(q, p_4)$ and $cs(q, p_1)$ are computed: $cs(q, p_4) = 0.494$ and $cs(q, p_1) = 0.582$. Thereafter, p_4 and p_1, with their combined scores, are copied to new universe objects. These object are then added to a set UR. (4) For each scar $r \in RS$, its combined score, regarding q, is computed: $cs(q, r_1) = 0.590$, and $cs(q, r_4) = 0.494$. Then, r_1 and r_4, with their combined scores, are copied to new universe objects. These objects are also added to the UR. The further exploitation of the UR is explained below.

Function *getRR(UR)*. This function aims to retrieve a subset of "universe objects" (simply objects) from the UR. These objects are selected to be potentially useful to solve the query q. To realize this function, we use both the "combined scores" of objects in the UR, and the "number" of objects in the UR that are associated with the same solution. The solution of each object $o \in UR$ is differently interpreted, according to whether o was created from a case c or a scar r. If created from c, its solution corresponds to c's solution, Otherwise, its solution corresponds to the solution in the r's consequent.

Let S_e be a set of solutions of objects in the UR. For each object in the UR, we find a subset $S_{e_k} \in S_e$, where all objects in $(S_{e_k})_{k \leq |S_e|}$ are associated with the same solution. For any $i, j \leq |S_e|$, thus $S_{e_i} \cap S_{e_j} = \emptyset$ holds. Then, for each $S_{e_k} \in S_e$, we compute the average of the combined scores of objects in S_{e_k}, denoted as $avg(S_{e_k})$. This $avg(S_{e_k})$ is further enhanced by multiplying a ratio, denoted as $strength(S_{e_k}) = |S_{e_k}|/|UR|$. The enhanced score is called the *final score* of S_{e_k}, and denoted as $fs(S_{e_k}) = avg(S_{e_k}) \cdot strength(S_{e_k})$. We then retrieve the objects, in the UR, grouped by the n top ranked solutions, by means of their final scores. If $n = 1$, we retrieve the objects grouped by the solution satisfying $\max(fs(S_{e_i}))_{i \leq |S_e|}$. These objects are finally stored into the RR.

To illustrate, consider the UR formed in the above example. Assuming that s_1 = 'gastritis' and s_2 = 'appendicitis', the UR has four objects: $UR = \{o_1, o_2, o_3, o_4\}$[11], where $\{o_1.cs = 0.494,\ o_1.s = s_1\}$, $\{o_2.cs = 0.582,\ o_2.s = s_2\}$, $\{o_3.cs = 0.590,\ o_3.s = s_2\}$ and $\{o_4.cs = 0.494, o_4.s = s_1\}$. With the UR, $fs(s_1) = 0.247$ and $fs(s_2) = 0.293$. If we choose the objects, in the UR, grouped by the solution satisfying $\max(fs(s_i))_{i=1,2}$, USIMSCAR returns the result set $RR = \{o_2, o_3\}$. Then, objects in the RR can be used to determine a solution of q using voting. With this example, since o_2 and o_3 have the solution 'appendicitis'. USIMSCAR thus give a diagnosis for q as 'appendicitis' that q really had.

5 Evaluation

Our evaluation goal is to empirically validates that USIMSCAR can improve similarity-based retrieval (SBR). To achieve it, we need to determine two

[11] Assume that each object has another field s representing "solution".

essential ingredients. The first is a set of representative models, implementing SBR, to be compared with USIMSCAR. The second is an application field, where the models and USIMSCAR can be properly tested. As the representative models, we choose k-NN and its several extensions, since SBR is typically implemented through the technique using a derivative of k-NN, as mentioned in Section 2. Regarding the application field, we choose *classification*, since the case-based approach, using the chosen models, to classification usually requires no sophisticated adaptation methods [3]. For the classification task, the performance of the models relies almost completely on the retrieval task identifying the similar cases to a new problem q [14]. So, we choose several k-NN based classifiers, for our comparison purpose, that will be described in the following section.

From the viewpoint of k-NN based classifiers, classification has two stages. The first is the determination of the nearest neighbors of a new problem q, driven by similarity knowledge. The second is the determination of the class of q using these neighbors. From the viewpoint of USIMSCAR, the first is the determination of the "useful cases and rules" $\mathcal{CR}$ for q, driven by "similarity knowledge and association knowledge". The second is the determination of the class of q using $\mathcal{CR}$. Our work is only focused on the first stage. Hence, to achieve the classification task, we use the simplest approach, majority voting (See Section 2), for the second stage. For the fair comparison, we configure majority voting to be used in the k-NN based classifiers compared. By applying this evaluation scheme, we justify the performance comparison between USIMSCAR and SBR.

5.1 Evaluation Methodology

We compare 5 k-NN based classifiers with USIMSCAR in our evaluation. These are all implemented in Weka [15]: (1) IB1 [16] is the simplest form of k-NN classifiers. To classify a new problem q, its nearest neighbor n_1 is selected by using the Euclidean distance. Then q is classified to be the class of n_1. As the baseline for our comparison purpose, we choose IB1 due to its simplicity. (2) IBkBN extends IB1 by using the best $k-$the number of nearest neighbors. The best k is determined by cross-validation. (3) IBkFS extends IBkBN by using feature selection. We choose the correlation-based feature selector [17] (known as CfsSubsetEval in Weka), to determine the goodness of feature subsets. (4) IBkFW extends IBkBN by using feature weighting. We choose InfoGainAttributeEval evaluator in Weka, due to its popularity. It assigns weights to features individually, based on the information gain with respect to the class. (5) KStar is an implementation of K* [18], where the similarity for finding nearest neighbors is defined by their *entropy*. The entropy is measured as the complexity of transforming one instance into the other. To classify q, KStar uses the probability of q being in class c by summing the probabilities from q to each member of c. It then chooses the class with the highest probability as the classification of q.

As our test datasets, we used 6 datasets found from UCI ML Repository (See Table 3). These were chosen by the criteria of being different in terms of number of instances (cases), number and types of attributes, and number of classes.

Table 3. The test datasets used in the experiments

Dataset	Instance #	Attribute #	Attribute Type			Class #
			Numeric	Binary	Nominal	
Breast Cancer	683	9	9			2
Car	1728	6			6	4
Heart Disease	270	13	6	3	4	2
Clever	297	13	6	3	4	2
Crx	653	15	6	4	5	2
Tae	151	5	1	2	2	3

For the evaluation metric, we used *classification accuracy*, since it is often assumed to be the best performance indicator in classification [19]. It measures the ratio of correctly classified instances over all the instances tested. To test a model on the datasets, we used *10-fold cross-validation*, in which each dataset is divided into 10 subsets. Of the 10 subsets, a subset is retained as *testing data*, and the remaining 9 subsets are used as *training data*. The validation process is then repeated 10 folds (times). Then, the 10 results from the folds were used to measure the classification accuracy of the model.

For USIMSCAR, the similarity knowledge was defined on two attribute types, numeric and categorical (including boolean). It is defined using the global-local principle, actually encoded in the similarity measure in Equation 2. The feature weights, in Equation 2, were equally assigned. Note that this measure is another form of the Euclidean distance that is used in the classifiers compared.

To generate scars, the following parameters were used: $\mathsf{minsupp} = 0.02$ (2%), $\mathsf{minsim} = 0.98$ (98%), $\mathsf{min\text{-}interesting} = 0.7$ (70%), $\mathsf{minitemsize} = 0.8$ (80%). To run USIMSCAR, we set k to 6.

5.2 Results and Analysis

We now evaluate the results of USIMSCAR, in comparison with the compared classifiers, in classification accuracy. We first compare the results of USIMSCAR and the baseline IB1. The results are shown in Table 4.

Referring to the above table, for each dataset, the better one in the accuracy is denoted in boldface. Also, the mark "•" indicates that USIMSCAR is determined to attain a statistically significant improvement over the target classifier, while "◦" shows there is no significant improvement found. As observed in the table, USIMSCAR outperforms IB1 on all the datasets in the accuracy. Outstandingly, USIMSCAR achieves 16.425% (maximum difference) higher than IB1 on the Car. Using the Z-test [20] at 95% confidence, for differences in the accuracy, USIMSCAR shows a significant improvement over IB1 on five datasets.

We now compare the results, of USIMSCAR with IBkDN, IBkFS and IBkFW, in classification accuracy. The results are seen in Table 5, where the best one in the accuracy for each dataset is also denoted in boldface. Also, the k, selected from 1 to 30, that gives the best classification accuracy on each dataset for each classifier, is denoted in the parentheses. As observed in the table, USIMSCAR occupies the 2^{th} place on the Breast Cancer, with a small difference (0.44%), compared to the

Table 4. USIMSCAR vs. IB1 in classification accuracy (%)

Dataset	IB1: Baseline	USIMSCAR
Breast Cancer	95.461 ○	**96.193**
Car	80.334 ●	**96.759**
Heart Disease	75.556 ●	**85.185**
Clever	76.014 ●	**84.459**
Crx	81.317 ●	**87.136**
Tae	64.238 ●	**67.550**

Table 5. USIMSCAR vs. IBkBN, IBkFS and IBkFW in classification accuracy (%).

Dataset	IBkBN	IBkFS	IBkFW	USIMSCAR
Breast Cancer	**96.633** (k=5)	**96.633** (k=5)	95.900 (k=5) ○	96.193
Car	80.334 (k=1) ●	80.334 (k=1) ●	84.833 (k=1) ●	**96.759**
Heart Disease	81.852 (k=8) ○	77.407 (k=6) ●	82.222 (k=10) ○	**85.185**
Clever	82.432 (k=7) ○	79.730 (k=10) ○	81.419 (k=7) ○	**84.459**
Crx	86.524 (k=10) ○	85.605 (k=3) ○	86.630 (k=7) ○	**87.136**
Tae	64.238 (k=1) ●	46.358 (k=2) ●	44.371 (k=1) ●	**67.550**

accuracy of IBkBN and IBkFS. However, USIMSCAR outperforms all the compared classifiers on all the remaining datasets. Outstandingly, it achieves 16.425% better than IBkBN on the Car, 21.192% better than IBkFS on the Tae, and 23,179% better than IBkFW on the Tae. Each mark "●" shows that there is a statistically significant difference between USIMSCAR and the target classifier on the considered dataset, using the Z-test at 95% confidence.

We now compare the results of USIMSCAR and KStar in classification accuracy. As observed in Table 6, USIMSCAR outperforms KStar on all the datasets. Outstandingly, USIMSCAR performs 10.370% and 17.463% better than KStar on the Car and Heart Disease, respectively, in the accuracy. Using the Z-test at 95% confidence in these results, we observe that the differences between them are statistically significant on five datasets, as the mark "●" indicates.

Table 6. USIMSCAR vs. KStar in classification accuracy (%)

Dataset	KStar	USIMSCAR
Breast Cancer	94.876 ○	**96.193**
Car	79.296 ●	**96.759**
Heart Disease	74.815 ●	**85.185**
Clever	75.675 ●	**84.459**
Credit Approval	78.560 ●	**87.136**
Tae	59.603 ●	**67.550**

We finally compare USIMSCAR with the classifiers, in the averages of the results in Tables 4 - 6. The comparison is presented in Fig. 1. As observed, USIMSCAR performs 7.393%, 4.211%, 8.536%, 6.985% and 9.076% better than IB1, IBkBN, IBkFS, IBKFW and KStar, respectively. These differences show that USIMSCAR achieves statistically significant improvements over the classifiers using the Z-test at 95% confidence. Through the experimental evaluation, we empirically verify that USIMSCAR is an effective retrieval strategy for CBR.

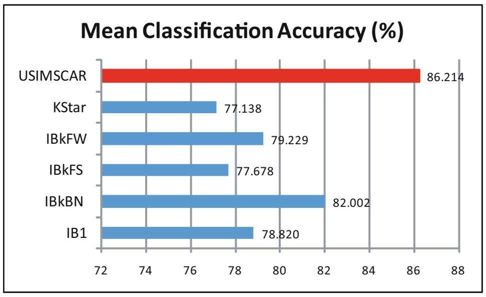


Fig. 1. The mean classification accuracy results

6 Related Work

SBR is typically implemented through the technique using a derivative of k-NN. To improve its accuracy, various extensions have been developed, including the integration of k-NN and the best k, feature selection, or feature weighting. USIMSCAR differs from these techniques. The most distinctive difference is the use of association knowledge for the retrieval task. The last two techniques often focus on finding the features that are highly correlated to specific solutions. But they only consider relationships between individual features and each solution. This leads to trouble, since they ignore complex relationships between multiple features and each solution. For example, two features may be individually correlated with a certain solution, but together they may not, or vice versa. In contrast, USIMSCAR exploits not only the individual relationships, but complex relationships between multiple features (itemsets in scars) and solutions.

Several researchers have attempted to augment SBR with certain factors obtained through statistical learning and adaptation knowledge. For example, Park et al. [21] suggest a new case retrieval technique, called *statistical* CBR (SCBR). The idea of SCBR is that an optimal number of neighbors can be dynamically obtained by considering the distribution of distances between potential similar neighbors for a new problem. SCBR finds the optimal distance threshold θ, and selects similar neighbors satisfying θ. Smyth and Keane [2] propose the adaptation-guided retrieval approach that provides direct link between the retrieval and the adaptation task in CBR. This approach utilizes formulated adaptation knowledge about whether a case can be easily modified to fit a new problem to influence similarity assessment during the retrieval phase. Hoffmann [22] also applies this approach for a dietary consultation evaluation for patients. USIMSCAR also significantly differs from the above approaches. First, it exploits association knowledge derived using data mining techniques and incorporates it into the retrieval task. Second, it does not assume that any kind of adaptation knowledge must be formalized in advance.

7 Extension Schemes and Conclusion

In CBR, cases can also be represented by more complex structures, like object-oriented (OO) or hierarchical (HR) representation [1]. We briefly give possible extension schemes, in which USIMSCAR could support the cases modeled using such structures. The OO representation utilizes the data modeling approach of the OO paradigm, such as “is-a” and inheritance [1]. In the HR representation, a case is characterized through multiple levels of abstraction, and its attribute values can reference nonatomic cases [1]. For USIMSCAR to treat the cases, characterized by those two representations, two issues must be addressed−how to formalize similarity knowledge, and how to generate association knowledge. To address the former, one may use similarity measures, for OO data [23] or HR data [24], widely studied in IR. To address the latter, one may integrate the soft-matching criterion and specific algorithms extending Apriori [10]. A possible choice for such algorithms is OR-FP [25] for OO data and DFMLA [26] for HR data.

USIMSCAR may also be extended to cases, where each case problem is associated with more than one solution. This occasion can be simply generalized into the occasion−each case problem is associated with one solution. The generalization is possibly done by splitting a case $\mathcal{C}$ into k number of sub cases (k: the number of solutions). We then restrict all the sub cases to have the same case identification with $\mathcal{C}$. Then, USIMSCAR may run for the cases, whose solutions are more than one, without any modification. This scheme even may be extended for solutions described in free-text. As long as such solutions are converted to the “bag-of-words” representation [6], the above scheme can be also applied.

In this paper, we proposed a new retrieval strategy USIMSCAR, aimed to improve similarity-based retrieval (SBR), used in many CBR systems. The uniqueness of USIMSCAR is to exploit the specific knowledge, integrating similarity knowledge and association knowledge, into retrieval in CBR. Similarity knowledge is encoded in similarity measures, while association knowledge is derived using association rule mining techniques. The goal of association knowledge is to represent implicit, previously unknown and potentially useful associations between problem features and solutions among stored cases. This knowledge is combined with similarity knowledge to make a more complete retrieval strategy. We empirically evaluated the performance of USIMSCAR, in comparison with several retrieval methods adopting SBR. The evaluation results showed that USIMSCAR is an effective retrieval strategy for CBR.

References

1. Lopez De Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, reuse, revision and retention in case-based reasoning. Knowl. Eng. Rev. 20, 215–240 (2005)
2. Smyth, B., Keane, M.T.: Adaptation-guided retrieval: questioning the similarity assumption in reasoning. Artif. Intell. 102, 249–293 (1998)

3. Stahl, A.: Learning of knowledge-intensive similarity measures in case-based reasoning. PhD thesis, Technical University of Kaiserslautern (2003)
4. Dudani, S.A.: The Distance-Weighted k-Nearest-Neighbor Rule. IEEE Transactions on Systems, Man and Cybernetics SMC-6, 325–327 (1976)
5. Jiang, L., Cai, Z., Wang, D., Jiang, S.: Survey of Improving K-Nearest-Neighbor for Classification. In: FSKD 2007: Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery, pp. 679–683 (2007)
6. Cunningham, P.: A Taxonomy of Similarity Mechanisms for Case-Based Reasoning. IEEE Trans. on Knowl. and Data Eng. 21, 1532–1543 (2009)
7. Yusta, S.C.: Different metaheuristic strategies to solve the feature selection problem. Pattern Recogn. Lett. 30, 525–534 (2009)
8. Wettschereck, D., Aha, D.W.: Weighting features. In: Proceedings of the First International Conference on CBR Research and Development, pp. 347–358 (1995)
9. Castro, J.L., Navarro, M., Sánchez, J.M., Zurita, J.M.: Loss and gain functions for CBR retrieval. Inf. Sci. 179, 1738–1750 (2009)
10. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB 1994, pp. 487–499 (1994)
11. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Proceedings of the 4th KDD, pp. 443–447 (1998)
12. Nahm, U.Y., Mooney, R.J.: Mining soft-matching association rules. In: Proceedings of CIKM 2002, pp. 681–683 (2002)
13. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: A survey. ACM Comput. Surv. 38, 9 (2006)
14. Jurisica, I., Glasgow, J.: Case-Based Classification Using Similarity-Based Retrieval. In: Proceedings of ICTAI (1996)
15. Witten, I.H., Frank, E.: Data mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco (2000)
16. Aha, D.W., Kibler, D., Albert, M.K.: Instance-Based Learning Algorithms. Mach. Learn. 6, 37–66 (1991)
17. Hall, M.A.: Correlation-based Feature Subset Selection for Machine Learning. PhD thesis, University of Waikato, Hamilton, New Zealand (1998)
18. Cleary, J.G., Trigg, L.E.: K*: An Instance-based Learner Using an Entropic Distance Measure. In: Proceedings of the 12th ICML, pp. 108–114 (1995)
19. Lim, T.S., Loh, W.Y., Shih, Y.S.: A comparison of prediction Accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learning, 203–229 (2000)
20. Richard, C.S.: Basic Statistical Analysis. Allyn & Bacon, Boston (2003)
21. Park, Y.J., Kim, B.C., Chun, S.H.: New knowledge extraction technique using probability for case-based reasoning: application to medical diagnosis. Expert Systems 23, 2–20 (2006)
22. Hoffmann, A., Khan, A.S.: A new approach for the incremental development of retrieval functions for CBR. Applied Artificial Intelligence 20, 507–542 (2006)
23. Bergmann, R., Stahl, A.: Similarity measures for object-oriented case representations. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 25–36. Springer, Heidelberg (1998)
24. Ganesan, P., Garcia-Molina, H., Widom, J.: Exploiting hierarchical domain structure to compute similarity. ACM Trans. on Infor. Sys. 21, 64–93 (2003)
25. Kuba, P., Popelinsky, L.: Mining frequent patterns in object-oriented data (2005)
26. Pater, S.M., Popescu, D.E.: Market-Basket Problem Solved With Depth First Multi-Level Apriori Mining Algorithm. In: SOFA 2009, 3rd International Workshop on Soft Computing Applications, pp. 133–138 (2009)

PG-Skip: Proximity Graph Based Clustering of Long Strings

Michail Kazimianec and Nikolaus Augsten

Faculty of Computer Science, Free University of Bozen-Bolzano,
Dominikanerplatz 3, 39100 Bozen, Italy
{kazimianec,augsten}@inf.unibz.it

Abstract. String data is omnipresent and appears in a wide range of applications. Often string data must be partitioned into clusters of similar strings, for example, for cleansing noisy data. A promising string clustering approach is the recently proposed Graph Proximity Cleansing (GPC). A distinguishing feature of GPC is that it automatically detects the cluster borders without knowledge about the underlying data, using the so-called proximity graph. Unfortunately, the computation of the proximity graph is expensive. In particular, the runtime is high for long strings, thus limiting the application of the state-of-the-art GPC algorithm to short strings.

In this work we present two algorithms, PG-Skip and PG-Binary, that efficiently compute the GPC cluster borders and scale to long strings. PG-Skip follows a prefix pruning strategy and does not need to compute the full proximity graph to detect the cluster border. PG-Skip is much faster than the state-of-the-art algorithm, especially for long strings, and computes the exact GPC borders. We show the optimality of PG-Skip among all prefix pruning algorithms. PG-Binary is an efficient approximation algorithm, which uses a binary search strategy to detect the cluster border. Our extensive experiments on synthetic and real-world data confirm the scalability of PG-Skip and show that PG-Binary approximates the GPC clusters very effectively.

1 Introduction

String clustering is required by many applications, such as Web analysis, document retrieval, and text cleansing. Different well-known clustering techniques were used to cluster syntactically similar strings, e.g., hierarchical, density-based, or partitional clustering. Typically, these methods require knowledge about the underlying data, for example, the number of expected clusters or a distance threshold. Such knowledge is often not available and hard to estimate.

Recently, Mazeika and Böhlen proposed the Graph Proximity Cleansing (GPC) method [1] for clustering strings. A distinctive feature of GPC is the automatic detection of the cluster borders. GPC randomly chooses a cluster center among the strings in the dataset and includes all strings within some similarity neighborhood into the cluster. Intuitively, the neighborhood is increased until further increasing it does not increase the cluster size. After each step that adds new strings to the cluster the cluster center is adjusted.

Technically, the problem of computing the cluster border for a given center is solved by computing its proximity graph. The proximity graph shows the similarity threshold τ on the x-axis and the number of strings within the respective similarity neighborhood on the y-axis. The cluster border is the rightmost endpoint of the longest horizontal line in the proximity graph (or the rightmost horizontal line if there are multiple horizontal lines of the same length). The proximity graph for the string `pauline` in the dataset {`adriana`, `irvin`, `piper`, `linda`, `paulina`, `pauline`} is shown in Figure 1. The longest horizontal line is between the similarity thresholds 4 and 6, thus $\tau = 6$ defines the cluster border, i.e., all strings with similarity 6 or more form a cluster around the center `pauline`.

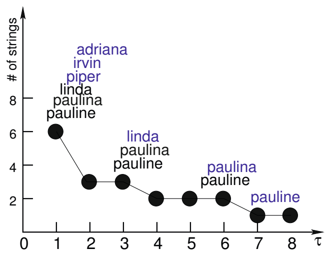


Fig. 1. Proximity Graph for the String 'pauline'

Unfortunately, the computation of the proximity graph is expensive, in particular for long strings. The computation time of the proximity graph critically depends on the number of neighborhoods that must be computed. For long strings, many neighborhoods must be computed, thus leading to high runtimes and limiting the state-of-the-art GPC algorithm to short strings.

In this paper we address the problem of computing GPC for large datasets with long strings efficiently. We present two new algorithms, PG-Skip and PG-Binary, that substantially reduce the runtime of the cluster border detection. Our algorithms leverage the fact that not all the neighborhoods of a proximity graph need to be computed to find the cluster border. In particular, only the neighborhoods required to determine the longest horizontal line are relevant.

PG-Skip implements prefix pruning, a strategy that skips the neighborhood computation for all points in the proximity graph below a certain similarity. The intuition is that the cluster is typically defined by a high similarity threshold, thus low similarities are unlikely to be relevant. Prefix pruning is effective since the neighborhoods for low similarities contain many strings and thus are more expensive to compute. The problem in prefix pruning is to skip only points that are not relevant for computing the cluster border. We prove that PG-Skip is a *correct and optimal* prefix pruning, i.e., it skips only irrelevant points and it skips all irrelevant points.

Summarizing, our contributions are the following:

- We propose the PG-Skip algorithm for computing the GPC cluster borders efficiently. We formally show that PG-Skip is the optimal prefix pruning.
- We present PG-Binary, an efficient and effective approximation algorithm for GPC cluster borders.
- Our extensive experiments on three real-world datasets show that our algorithms are substantially faster then the state-of-the-art algorithm, especially for long strings. The approximation algorithm, PG-Binary, is faster than PG-Skip and reaches an effectiveness of up to 99%.

The remaining paper is organized as follows. We briefly revisit the state-of-the-art in computing GPC clusters in Section 2. In Section 3 we introduce our new algorithms, PG-Skip and PG-Binary, which are empirically evaluated in Section 4. Related work is discussed in Section 5. We conclude and identify future work in Section 6.

2 Background

In this section we give a short introduction to GPC and the state-of-the-art in computing the cluster border using the proximity graph.

2.1 Proximity Graph

Let s be a string, and $\bar{s}$ be s prefixed and suffixed with $q-1$ characters '#'. The **profile** $P(s,q)$ of the string s is the multiset of all substrings of $\bar{s}$ of length q, called q-grams. The **overlap** of two profiles $P(s',q)$ and $P(s'',q)$ is the cardinality of their intersection, i.e., $o(P(s',q),P(s'',q)) = |P(s',q) \uplus P(s'',q)|$. The overlap measures the similarity between two strings; the higher the overlap, the more similar are the strings.

Given a set of strings, D, and a profile, P, the **neighborhood** of P in D for similarity threshold τ (τ-neighborhood) is the subset of all strings of D that have an overlap of at least τ with P, $N(D,P,\tau) = \{s \in D : o(P,P(s,q)) \geq \tau\}$; we simply write N_τ if P and D are clear from the context. The **center** $P_c(N_\tau,q)$ of the neighborhood N_τ is the profile that consists of the K most frequent q-grams in the neighborhood, i.e., in $\biguplus_{s \in N_\tau} P(s,q)$, where $K = \sum_{s \in N_\tau} |P(s,q)|/|N_\tau|$ is the average profile size in the neighborhood N_τ.

Definition 1. *Let D be a set of strings, $s \in D$, q the size of the q-grams. The* **proximity graph** *of string s is defined as $PG(s,D,q) = ((1,|N_1|), (2,|N_2|), \ldots, (m,|N_m|))$, $m = |P(s,q)|$, where N_τ is recursively defined as follows:*

$$N_\tau = \begin{cases} \{s\} & \text{if } \tau = |P(s,q)|, \\ N(D,P_c(N_{\tau+1},q),\tau) \cup N_{\tau+1} & \text{otherwise.} \end{cases} \quad (1)$$

Table 1. Computation of the Proximity Graph $PG(pauline, D, q)$ in Example 1

τ	Neighborhood, N_τ	Center of N_τ, $P_c(N_\tau, q)$
8	$\{s_6\}$	$\{\#p, pa, au, ul, li, in, ne, e\#\}$
7	$\{s_6\}$	$\{\#p, pa, au, ul, li, in, ne, e\#\}$
6	$\{s_5, s_6\}$	$\{\#p, pa, au, ul, li, in, na, a\#\}$
5	$\{s_5, s_6\}$	$\{\#p, pa, au, ul, li, in, na, a\#\}$
4	$\{s_5, s_6\}$	$\{\#p, pa, au, ul, li, in, na, a\#\}$
3	$\{s_4, s_5, s_6\}$	$\{\#p, pa, au, ul, li, in, a\#\}$
2	$\{s_4, s_5, s_6\}$	$\{\#p, pa, au, ul, li, in, a\#\}$
1	$\{s_1, s_2, s_3, s_4, s_5, s_6\}$	$\{\#p, au, ul, li, in, na, a\#\}$

The proximity graph maps similarity thresholds τ, $1 \leq \tau \leq |P(s, q)|$, to the size of the respective neighborhood N_τ and is computed from right to left, i.e., from the largest to the smallest overlap. The neighborhood of the largest overlap is defined to be $\{s\}$. For the remaining points the neighborhood is computed around the center of the previous neighborhood.

Example 1. Let $D = \{s_1, s_2, \ldots, s_6\} = \{$*adriana*, *irvin*, *piper*, *linda*, *paulina* *pauline*$\}$, $q = 2$. We compute the proximity graph for the string $s_6 =$ *pauline*. The similarity threshold ranges between $\tau = 1$ and $\tau = |P(s_6, q)| = 8$. $N_8 = \{s_6\}$ by definition. The neighborhood N_7 is computed around the center $P_c(N_8, q) = \{\#p, pa, au, ul, li, in, ne, e\#\}$ of the neighborhood N_8, N_6 is computed around $P_c(N_7, q)$, and so on. Table 1 shows all neighborhoods and their centers, Figure 1 illustrates the resulting proximity graph.

2.2 Cluster Border Detection and State-of-the-Art Algorithm

Let $PG = \{(1, |N_1|), (2, |N_2|), \ldots, (m, |N_m|)\}$ be a proximity graph. We define the horizontal lines in the proximity graph by their endpoints. The set of all **horizontal lines** in the proximity graph PG is defined as $H(PG) = \{(i, j) |\ (i, |N_i|), (j, |N_j|) \in PG, |N_i| = |N_j|, i \leq j\}$. The length of a horizontal line (i, j) is $j - i$. The **cluster border**, $border(PG) = \{j |\ (i, j) \in H(PG), \forall(x, y) \in H(PG) :\ y - x \leq j - i\}$, is the right endpoint of the rightmost horizontal line of maximal length (border-defining horizontal line). The **GPC cluster** is the neighborhood N_b for the similarity threshold $b = border(PG)$.

Example 2. In the proximity graph in Figure 1 the cluster border is $b = 6$, and the GPC cluster is $N_b = \{pauline, paulina\}$.

PG-GPC (Algorithm 1) is the state-of-the-art in computing GPC clusters. Given a set D of strings, a string $s \in D$, and the q-gram size q, PG-GPC first computes the full proximity graph $PG(s, D, q)$, i.e., all neighborhoods N_τ in the range $\tau = 1, 2, \ldots, |P(s, q)|$, and then detects the cluster border b in the computed proximity graph.

Algorithm 1. PG-GPC(s, D, q)

```
Data: $s$: center; $D$: set of strings; $q$: size of $q$-grams
Result: cluster of strings
begin
    $\tau \leftarrow |P(s,q)|$;
    $PG[1..\tau]$ : empty array of neighborhoods; // proximity graph
    $PG[\tau] \leftarrow \{s\}$; // neighborhood for $\tau = |P(s,q)|$ is $\{s\}$
    $\tau \leftarrow \tau - 1$;
    while $\tau \geq 1$ do
        $P \leftarrow P_c(PG[\tau+1], q)$; // $P$ is the center of the previous neighborhood
        $PG[\tau] \leftarrow N(D, P, \tau) \cup PG[\tau+1]$; // $\tau$-neighborhood of $P$ in $D$
        $\tau \leftarrow \tau - 1$;
    $b \leftarrow border(PG)$; // compute the cluster border
    return $PG[b]$; // return the cluster
end
```

Fig. 2. Computation of the Cluster

Algorithm 2. GPC(D, q)

```
Data: $D$: set of strings; $q$: size of $q$-grams
Result: set of mutually disjoint clusters covering $D$
begin
    $\zeta \leftarrow \emptyset$; // initialize the clustering
    $\phi \leftarrow D$; // initialize the set of eligible centers
    while $\phi \neq \emptyset$ do // while non-clustered strings are left
        $s \leftarrow$ random string from the set $\phi$ of eligible centers;
        $C = PG\text{-}GPC(s, D, q)$; // compute the GPC cluster around $s$
        // update the clustering $\zeta$ and the eligible centers $\phi$
        $\zeta \leftarrow \zeta \cup \{C\}$; // add new cluster
        $\phi \leftarrow \phi \setminus C$; // clustered strings not eligible as centers
    // merge overlapping clusters
    while $\exists C_i, C_j \in \zeta : C_i \neq C_j, C_i \cap C_j \neq \emptyset$ do
        $\zeta \leftarrow Clusters \setminus \{C_i, C_j\} \cup \{C_i \cup C_j\}$;
    return $\zeta$;
end
```

Fig. 3. GPC Clustering Algorithm

2.3 The GPC Method

GPC (Algorithm 2) takes a string set D and the q-gram size q at the input and returns a set of disjoint clusters $\zeta = \{C_1, C_2, \ldots, C_k\}$ that cover D. It randomly selects a non-clustered string $s \in D$ as the cluster center and computes the cluster border b. The new cluster C is the b-neighborhood in the proximity graph of s. GPC marks all the strings of C as clustered and proceeds with the next center until all strings are clustered. Overlapping clusters are merged to produce a hard clustering.

3 Efficient Border Detection

The computation of the proximity graph is at the core of the GPC method. The state-of-the-art GPC algorithm [2] computes the full proximity graph to detect the cluster borders. This is expensive, especially for long strings for which many neighborhoods must be computed. Among the neighborhoods, those of low similarity thresholds are harder to compute since they include more strings.

In this section we introduce the concept of prefix pruning and two new algorithms, PG-Skip and PG-Binary, to detect the cluster border. Both algorithms leverage the fact that not all points of the proximity graph must be computed to detect the cluster border. PG-Skip is shown to be a correct and optimal prefix pruning algorithm, PG-Binary is an efficient approximation. The input of the algorithms is a set of strings D, the center string $s \in D$, and the q-gram size q. The algorithms return the (approximated) GPC cluster around s.

3.1 Prefix Pruning of the Proximity Graph

We observe two properties of the proximity graph. (1) The neighborhoods of the low similarity thresholds are more expensive to compute since they contain more strings. (2) The cluster border is often a high similarity threshold and the low similarity thresholds are not relevant to detect it.

These observations lead to a pruning strategy that avoids computing the neighborhoods of low similarity thresholds, the prefix pruning. Given a proximity graph $PG = ((1, n_1), (2, n_2), \ldots, (m, n_m))$, a k-prefix of PG is defined as $\text{pref}(PG, k) = \{(i, n_i) | (i, n_i) \in PG, i \leq k\}$. A prefix pruning algorithm does not compute the full proximity graph, but only a suffix $\text{suff}(PG, k+1) = PG \setminus \text{pref}(PG, k)$ for some cutoff point k that depends on the proximity graph.

Intuitively, a prefix pruning algorithm is *correct* if the neighborhoods in the pruned prefix are not relevant for the border detection. A prefix pruning algorithm is *optimal* if it always prunes the largest possible prefix that leads to a correct pruning and thus minimizes the computation cost. We formally define the correctness and optimality of a prefix pruning algorithm.

Definition 2. *A prefix pruning algorithm is* ***correct*** *iff for all proximity graphs PG it prunes a prefix* $\text{pref}(PG, k)$ *such that* $\forall PG' : \text{suff}(PG', k+1) = \text{suff}(PG, k+1) \Rightarrow \mathit{border}(PG') = \mathit{border}(\text{suff}(PG, k+1))$. *A prefix pruning algorithm is* ***optimal*** *iff it is correct and for all proximity graphs PG it prunes the maximal prefix* $\text{pref}(PG, k)$*, i.e., for any prefix* $\text{pref}(PG, x)$*,* $x > k$*,* $\exists PG' : \text{suff}(PG', x+1) = \text{suff}(PG, x+1) \wedge \mathit{border}(PG') \neq \mathit{border}(\text{suff}(PG, x+1))$.

If we prune a prefix $\text{pref}(PG, k)$, we do not know how the proximity graph PG continues in the range $[1, k]$. PG' in the definition stands for all possible variants of PG, given its suffix $\text{suff}(PG, k+1)$.

3.2 The PG-Skip Algorithm

In this section we present the PG-Skip algorithm that implements the prefix pruning strategy, i.e., it skips the computation of irrelevant neighborhoods at the left end of the proximity graph. We prove that PG-Skip is correct and optimal.

PG-Skip (Algorithm 3) computes the proximity graph from right to left in decreasing order of the similarity threshold. After the neighborhood for the current

Algorithm 3. PG-Skip(s, D, q)

```
Data: $D$: set of strings; $s$: center string, $q$: gram size
Result: GPC cluster around the center string $s$
1  begin
2      $m \leftarrow |P(s,q)|$;
3      $PG[1..m]$ : empty array of neighborhoods; // proximity graph
4      $PG[m] \leftarrow \{s\}$; // neighborhood for $\tau = |P(s,q)|$ is $\{s\}$
5      $l_{max} \leftarrow 0$; // length of the border-defining horizontal line
6      $l \leftarrow 0$; // length of the current horizontal line
7      $b \leftarrow m$; // cluster border
8      for $\tau = m - 1$ downto 1 do
9          $P \leftarrow P_c(PG[\tau+1], q)$; // $P$ is the center of the previous neighborhood
10         $PG[\tau] \leftarrow N(D, P, \tau) \cup PG[\tau+1]$; // $\tau$-neighborhood of $P$ in $D$
11         if $|PG[\tau]| = |PG[\tau+1]|$ then $l \leftarrow l + 1$; // increase current horizontal line
12         else $l \leftarrow 0$; // start new horizontal line
13         if $l > l_{max}$ then // current horizontal line is new border-defining line
14             $l_{max} \leftarrow l$; $b \leftarrow \tau + l$;
           // prefix pruning
15         if $(b = \tau + l \wedge \tau - 2 \leq l_{max}) \vee (\tau - 1 + l \leq l_{max})$ then
16             return $PG[b]$; // prune prefix of length $\tau - 1$
17     return $PG[b]$;
18 end
```

Fig. 4. PG-Skip: Border Detection Algorithm

similarity threshold τ is computed (Line 10), the length l of the current horizontal line (i.e., the longest horizontal line with the left endpoint in τ) is updated (Lines 11–12). If $l > l_{max}$, the current horizontal line is longer than all horizontal lines found so far, l_{max} is updated, and the cluster border b is set to the right end of the current horizontal line (Lines 13–14).

The pruning is done in Lines 15–16. In the pruning condition in Line 15, the left side of the disjunction covers the case when τ is the left endpoint of the border-defining horizontal line ($b = \tau + l$): if $\tau - 2 \leq l_{max}$, the prefix pref$(PG, \tau - 1)$ is pruned and the cluster defined by the current border is returned. The intuition is that the pruned prefix can not contain a new horizontal line that is longer than l_{max}. The right side of the disjunction covers the case $b \neq \tau + l$: the prefix pref$(PG, \tau - 1)$ is pruned if the horizontal line that starts or continues in τ can not grow longer than l_{max} in the pruned prefix.

Example 3. Figure 6(b) illustrates PG-Skip. The algorithm computes the neighborhoods from right to left starting with the similarity threshold $\tau = 16$. The algorithm stops at $\tau = 4$ and prunes the prefix of length 3 since $\tau - 1 - l \leq l_{max}$ ($l = 0, l_{max} = 3$). $l_{max} = 0$ and $b = 16$ in the similarity range $[15, 16]$; $l_{max} = 1$, $b = 15$ in $[10, 14]$, $l_{max} = 2$, $b = 11$ for $\tau = 9$, $l_{max} = 3$ and $b = 11$ in $[4, 8]$.

Theorem 1. *The prefix pruning algorithm PG-Skip is correct and optimal.*

Proof. In the pruning condition (Line 15), τ is the smallest similarity threshold computed so far, l is the length (possibly zero) of the current horizontal line, l_{max} is the length of the border-defining horizontal line in suff(PG, τ). We distinguish two complementary cases and show correctness and optimality by contradiction.

(1) τ is the left endpoint of the border-defining horizontal line in suff(PG, τ): Since $b = \tau + l$, $l = l_{max}$, and $\tau - 1 \leq 0 \wedge l_{max} > 0 \Rightarrow \tau - 2 \leq l_{max}$, the pruning

condition is equivalent to $\tau - 2 \leq l_{max}$. *Correctness:* Assume a proximity graph PG', $\text{suff}(PG', \tau) = \text{suff}(PG, \tau)$ with $border(PG') \neq border(PG)$. The border-defining horizontal line of PG' must be in the interval $[1, \tau - 1]$ and can be at most of length $\tau - 2$. This is not possible since $\tau - 2 \leq l_{max}$. *Optimality:* Assume we prune a prefix $\text{pref}(PG, x)$, $x > \tau - 1$. Case $\tau - 2 = l_{max}$: The prefix $\text{pref}(PG, x)$ can contain a horizontal line $(1, \tau)$ of length $\tau - 1 > l_{max}$ and thus the pruning is not correct; case $\tau - 2 < l_{max}$: $border(\text{suff}(PG, \tau)) \neq border(\text{suff}(PG, x))$ since the condition $\tau - 2 \leq l_{max}$ also holds for the similarity threshold $\tau + 1$, but the algorithm did not exit in the previous loop. The prefix $\text{pref}(PG, x)$ may not contain a horizontal line and thus the pruning is not correct.

(2) τ is *not* the left end of the border-defining horizontal line in $\text{suff}(PG, \tau)$: Since $b \neq \tau + 1$, the pruning condition is equivalent to $\tau - 1 + l \leq l_{max}$. *Correctness:* Assume a proximity graph PG', $\text{suff}(PG', \tau) = \text{suff}(PG, \tau)$ with $border(PG') \neq border(PG)$. The border-defining horizontal line of PG' must be in the interval $[1, \tau + l]$ and can be at most of length $\tau + l - 1$. This is not possible since $\tau + l - 1 \leq l_{max}$. *Optimality:* The condition $\tau - 1 + l \leq l_{max}$ only holds if $l = 0$ since on a horizontal line $\tau - 1 + l$ is constant and is first evaluated for the right endpoint; thus $\tau - 1 \leq l_{max}$. Assume we prune a prefix $\text{pref}(PG, x)$, $x > \tau - 1$. (a) $\tau - 1 = l_{max}$: The prefix $\text{pref}(PG, x)$ may be such that $(1, \tau + 1)$ is a horizontal line and thus the pruning is not correct; (b) $\tau - 1 < l_{max}$: Since the pruning condition did not hold in the previous loop, $\tau + 1$ is the left end of a horizontal line; since the pruning condition did not hold for any point on the horizontal line, the line must start at a similarity threshold $y > l_{max} + 1$. For any $x > \tau - 1$ the pruning is incorrect since there is a proximity graph PG' with the horizontal line $(1, y)$ (length $y - 1 > l_{max}$) for which $\text{suff}(PG', x + 1) = \text{suff}(PG, x + 1)$. □

3.3 The PG-Binary Algorithm

In this section we present PG-Binary, our approximation algorithm for computing GPC clusters. In addition to pruning a prefix of the proximity graph (as PG-Skip does), PG-Binary also prunes other neighborhood computations.

PG-Binary (Algorithm 4) uses the fact that the proximity graph is monotonically decreasing. If the neighborhoods for two similarity thresholds $a < b$ are of the same cardinality, $|N_a| = |N_b|$, then there is a horizontal line (a, b) in the proximity graph. The neighborhoods between a and b need not to be computed to verify the horizontal line (a, b).

PG-Binary maintains the length l_{max} of the border-defining horizontal line and computes the proximity graph from right to left for decreasing starting points st of the current horizontal line. In each iteration, PG-Binary does the following steps. If st starts a new border-defining horizontal line (that ends in $en = st - l_{max} - 1$), update l_{max} and the current border b. Otherwise, move st to the starting point of the horizontal line that goes through en.

When st is moved, two cases are distinguished. (1) st starts the border-defining horizontal line of length $st = l_{max}$; in this case, en starts a new horizontal line

Algorithm 4. PG-Binary(s, D, q)

```
Data: $D$: set of strings; $s$: center string, $q$: gram size
Result: cluster of strings for the center $s$
1  begin
2      $m \leftarrow |P(s, q)|$;
3      $PG[1..m]$ : empty array of neighborhoods; // proximity graph
4      $PG[m] \leftarrow \{s\}$; // neighborhood for similarity threshold $m$ is $\{s\}$
5      $l_{max} \leftarrow 0$; // length of the border-defining horizontal line
6      $b \leftarrow m$; // cluster border
7      $st \leftarrow m$; // start (right endpoint) of the current horizontal line
8      while $st > 1$ do
9          $en \leftarrow st - l_{max} - 1$;
10         $P \leftarrow P_c(PG[st], q)$; // center of neighborhood $N_{st}$
11         $PG[en] \leftarrow N(D, P, en) \cup PG[st]$; // neighborhood $N_{en}$ of $P$ in $D$
12         if $|PG[en]| = |PG[st]|$ then // border-defining horizontal line starts in $st$
13             $l_{max} \leftarrow l_{max} + 1$; $b \leftarrow st$;
14         else if $b = st$ then // new horizontal line starts in $en$
15             $st \leftarrow en$;
16         else
               // binary search for start $st$ of horizontal line through $en$
17             while $st \neq en \wedge st - 2 \geq l_{max} + 1$ do
18                 $mid = \lceil (en + st)/2 \rceil$;
19                 if $PG[mid] \neq \emptyset$ then $PG[mid] \leftarrow N(D, P, mid) \cup PG[st]$;
20                 if $|PG[en]| = |PG[mid]|$ then $en \leftarrow mid$; else $st \leftarrow \min(mid, st - 1)$;
21             if $en \neq st$ then return $PG[b]$; // pruning in binary search
22         if $(b = st \wedge st - l_{max} - 2 < l_{max} + 1) \vee (st - 1 < l_{max} + 1)$ then
23             return $PG[b]$; // prefix pruning
24     return $PG[b]$;
25 end
```

Fig. 5. PG-Binary: Border Detection Algorithm

and st is set to en. (2) Otherwise, the starting point of the horizontal line through en is between en and st, and a binary search is performed in the interval $[en, st)$.

The binary search computes the middle point mid between en and st. If there is a horizontal line between en and mid ($|N_{en}| = |N_{mid}|$), a binary search is done in $[mid, st)$, otherwise in $[en, mid)$. The binary search terminates when the starting point of the horizontal line is found or no horizontal line of length $l_{max} + 1$ can start in the search interval $[en, st)$. In the latter case, the algorithm prunes all remaining neighborhood computations and stops.

In Line 22, PG-Binary prunes the prefix $\text{pref}(PG, st - 1)$ (a) if st starts the border-defining horizontal line ($b = st$) and no new border-defining horizontal line can start to the left of st ($st - l_{max} - 2 < l_{max} + 1$), or (b) if st can not be the starting point of a new border-defining horizontal line ($st - 1 < l_{max} + 1$).

Example 4. Figure 6(c) illustrates PG-Binary. The algorithm moves st from left to right and checks for new border-defining horizontal lines (en, st). For $(14, 15)$, $(9, 11)$, $(8, 11)$, the check is a success and the border b and l_{max} are updated in Line 13; st is the same in the next iteration, but en changes (due to l_{max}). For $(15, 16)$, $(13, 15)$, $(7, 11)$ en is the next point after the border-defining horizontal line and st is set to en (no binary search required). For $(11, 13)$ and $(3, 7)$ a binary search identifies the start of the horizontal line through en

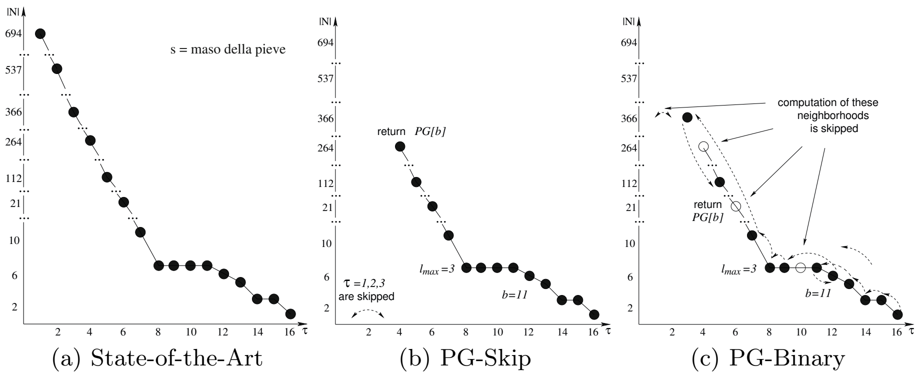


Fig. 6. PGs for the String 'maso della pieve' in the Street Name Dataset

(Lines 17–20). The algorithm stops the binary search for $st = 5$, since there cannot be a horizontal line longer than $l_{max} = 3$ to the left of st (Line 17) and terminates (Line 21). The arrows show the order in which the neighborhoods are computed.

Discussion. In some cases, PG-Binary only approximates the correct GPC cluster. The reason is that in the original GPC algorithm the center for computing a neighborhood N_τ depends on the center of the previous neighborhood $N_{\tau+1}$. Since PG-Binary skips intermediate points, $N_{\tau+1}$ may not be available and an other neighborhood must be used to compute the center.

In practice, the GPC cluster computed by PG-Binary is often correct. In particular, errors can only be introduced

- to the left of the rightmost horizontal line of length > 0,
- by points that start a horizontal line of length ≥ 0,
- if the center of the skipped neighborhoods is different from the center of the next neighborhood to the right.

The prefix pruning does not introduce an additional error. In Section 4 we empirically evaluate the quality of the approximation on three real world datasets.

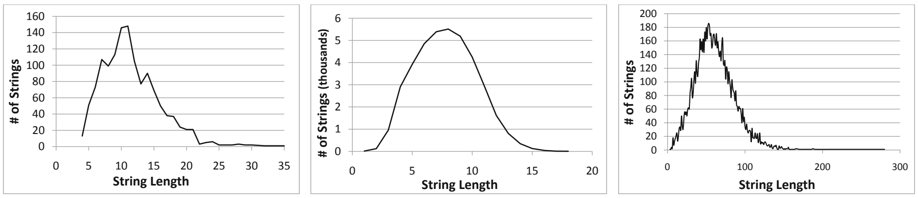
4 Experiments

4.1 Datasets and Experimental Setup

We compare our algorithms PG-Skip (Algorithm 3) and PG-Binary (Algorithm 4) with the state-of-the-art algorithm [2] for computing GPC clusters (PG-GPC, Algorithm 1). We evaluate our algorithms on synthetic data and three real-world datasets: Bozen Street Names is a dataset with 1313 syntactic representations of 403 street names in the city of Bozen-Bolzano[1]; the Oxford misspellings[2] are 39030

[1] `http://www.inf.unibz.it/~augsten/publ/tods10`
[2] `http://www.dcs.bbk.ac.uk/~roger/corpora.html`

strings in 6003 non-overlapping clusters (each cluster consists of a correct string and its misspellings); the DBLP dataset consists of 10000 paper titles chosen from the online bibliography DBLP[3]. The distribution of the string lengths is very different between the datasets and is shown in Figure 7.



(a) Bozen Sreet Names (b) Oxford Misspellings (c) DBLP Paper Titles

Fig. 7. Distribution of String Length

4.2 Proximity Graph Computation

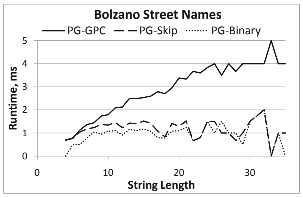
Scalability and String Length. Figure 8 shows the average time for computing the proximity graph for a string of a specific length. Our algorithms are consistently faster than the state-of-the-art algorithm. The performance advantage clearly increases with the string length. For the DBLP dataset with its long strings, PG-Binary is faster than PG-Skip.

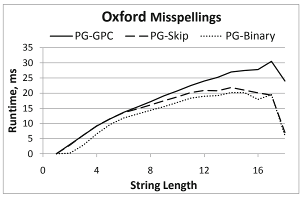
Effectiveness of the Pruning Strategy. In this experiment, we count the number of neighborhood computations that are skipped by the algorithms. Let n be the total number of neighborhoods that can be computed for some similarity threshold τ in the dataset, let k be the number τ-neighborhoods that where actually computed. Then $PSN = \frac{n-k}{n}$ measures the percentage of skipped neighborhoods. $PSN = 1$ for τ if no τ-neighborhood was computed by an algorithm, $PSN = 0$ if all τ-neighborhoods where computed; thus high PSN values are good.

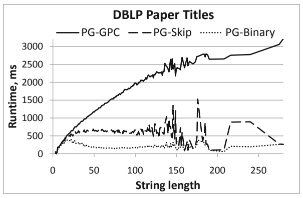
The results for the three real-world dataset are shown in Figure 9. The baseline algorithm always computes the full proximity graph and no neighborhoods are pruned ($PSN = 0$). The pruning power of both PG-Skip and PG-Binary increases for the datasets with long strings and is best for DBLP. For low similarity thresholds much more neighborhoods are pruned than for high thresholds. PG-Binary skips more neighborhoods since in addition to pruning a prefix, it also skips intermediate neighborhood computations.

Overall, the results are very encouraging. Our pruning strategy works best on the datasets with long strings. The long strings are more expensive to compute, because of the greater number of neighborhoods in the proximity graphs. Further, most of the pruned points are small similarity thresholds, for which the neighborhoods are more expensive to compute, because they contain more strings.

[3] `http://www.informatik.uni-trier.de/~ley/db`

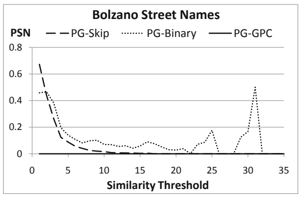


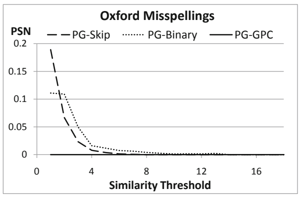


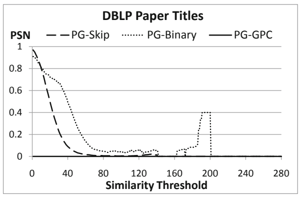


(a) Bozen Sreet Names (b) Oxford Misspellings (c) DBLP Paper Titles

Fig. 8. Proximity Graph Computation







(a) Bozen Sreet Names (b) Oxford Misspellings (c) DBLP Paper Titles

Fig. 9. Percentage of the Skipped Neighborhoods in the Proximity Graphs

4.3 Scalability Results on Synthetic Data

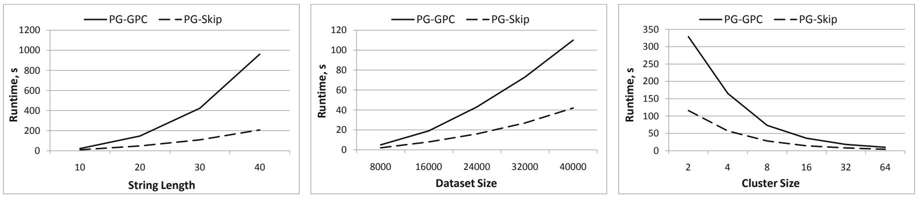
In this section we do a controlled experiment on synthetic data and compare our optimal prefix pruning algorithm PG-Skip with the state-of-the-art algorithm PG-GPC. We produce datasets with random strings. For each random string we produce a number of noisy strings by deleting, inserting, or renaming some characters.

In Figure 10(a) we produce datasets with a fixed number of strings ($|D| = 32000$) and clusters of size $|C| = 8$ and vary the string length. PG-Skip clearly outperforms PG-GPC and the performance advantage increases with the string length; for strings of length 40, PG-Skip is almost five times faster. Figure 10(b) shows that PG-Skip scales better to larger datasets than the state-of-the-art algorithm.

The runtime of GPC decreases with the cluster size, because less proximity graphs must be computed (one for each cluster) if the clusters become larger (Figure 10(c)). For all cluster sizes, our algorithm is faster than PG-GPC.

4.4 Clustering Runtime and Quality

Runtime. Table 2 shows the runtime results for clustering the real-world datasets. Our algorithms are always faster than PG-GPC. We get the best performance gain for DBLP, for which PG-Skip is more than 2 times faster, PG-Binary even more than 6 times.



(a) $|D| = 32000, |C| = 8$ (b) $|s| = 16, |C| = 8$ (c) $|s| = 16, |D| = 32000$

Fig. 10. Clustering of Synthetic Data

Table 2. Quality of the Pruning Methods

	PG-GPC	**PG-Skip**		**PG-Binary**	
Dataset	*time*	*time*	*F-measure*	*time*	*F-measure*
Street Names	2120 ms	1597 ms	1	1139 ms	0.79
Oxford	733 s	637 s	1	570 s	0.98
DBLP Sample	11443 s	5245 s	1	1735 s	0.99

Quality. We compare the clustering obtained by our GPC approximation PG-Binary with the clustering that results from the original GPC algorithm. To measure how close the approximated clusters are to the GPC clusters we apply the F-measure, which is defined as the harmonic mean of precision p and recall r, $F = 2 \cdot \frac{p \cdot r}{p+r}$.

We consider the pairwise statistics of the dataset elements in order to compute precision and recall. Let D be the string dataset, ξ be the GPC clustering, and ζ be the clustering obtained by the pruning method. Each pair $(s_1, s_2) \in D \times D$ is classified into one of the following sets:

- TP (true positive) if $\exists K \subset \zeta : s_1, s_2 \in K$ and $\exists H \in \xi : s_1, s_2 \in H$,
- TN (true negative) if $\exists K, L \in \zeta : s_1 \in K, s_2 \in L$ and $\exists H, P \in \xi : s_1 \in H, s_2 \in P$,
- FP (false positive) if $\exists K \in \zeta : s_1, s_2 \in K$ and $\exists H, P \in \xi, s_1 \in H, s_2 \in P$,
- FN (false negative) if $\exists K, L \in \zeta : s_1 \in K, s_2 \in L$ and $\exists H \in \xi : s_1, s_2 \in H$.

Then precision and recall are computed as usual, $p = |TP|/(|TP| + |FP|)$ and $r = |TP|/(|TP| + |FN|)$.

The results are shown in Table 2. GPC-Binary performs best for the Oxford and the DBLP dataset (F-measure at least 0.98). In the Oxford dataset, the strings are short and have small horizontal lines and PG-Binary skips only few points, increasing the clustering quality but also increasing the runtime; in fact, the gain over PG-Skip is small. The best setting for PG-Binary is the DBLP dataset. The strings are long and have long horizontal lines such that PG-Binary can take full advantage of the binary search. Since the PG-Skip pruning algorithm computes the exact GPC clusters, the clusterings are identical, and the F-measure is always 1.

Our experimental evaluation confirms the analytic result and suggests that PG-Skip should always be preferred over the state-of-the-art algorithm, since it is always faster and results in the same clustering. For datasets with long strings, PG-Binary should be considered, because it is faster than PG-Skip and the approximation error in the clustering is small.

5 Related Work

Proximity Graph Cleansing (GPC) was originally developed by Mazeika and Böhlen [1] to cleanse string data that cannot be found in a dictionary, like proper names in biology, medicine, or geography. The strings are clustered and all strings in a cluster are substituted with the most frequent spelling in the cluster. For this task traditional clustering techniques are hard to use, since they often require input parameters, for example, the number of clusters that are not known in a data cleansing setting. Mazeika and Böhlen only approximate the GPC clusters using sampling. In our work we develop new algorithms for computing the GPC clusters of Mazeika and Böhlen efficiently.

Kazimianec and Augsten [2] recently presented the first algorithms, PG-DS and PG-SM, for computing the exact GPC clusters. PG-DS is based on the τ-occurrence problem [3], PG-SM on sort-merge joins of q-grams [4]. In combination with the DivideSkip technique on the inverted list index that was developed by Chen Li *et al.* [5], the exact PG-DS algorithm outperforms the approximate algorithm of Mazeika and Böhlen for reasonable sample sizes. In the present paper, we develop PG-Skip and show that it is correct, i.e., it always computes the exact GPC clusters, and it is always faster than PG-DS.

The application of the GPC method to short strings was investigated by Kazimianec and Mazeika [6]. In datasets with short strings the proximity graph may not have a horizontal line. The authors propose a new border criterion and get better results for short strings. Further the cluster quality increases when the longest non-clustered string in a dataset is chosen as the center string, rather than choosing strings randomly.

GPC uses the overlap of q-gram profiles to measure similarity between strings. In order to improve the effectiveness of GPC, other types of q-grams can be used, e.g. positional q-grams or q-grams of variable length [7]. In our work we do not change the original GPC method but improve its scalability.

String clustering and cleansing is an active field of research. Depending on the task, specific clustering techniques like GPC or general techniques like partitional [8], hierarchical [9] and density-based [10, 11] clustering are used to cluster strings. If a dictionary is available, spelling techniques [12, 13] can be used to cleanse strings. In fuzzy string matching, the goal is to find strings that are similar to a given pattern. Chaudhuri *et al.* [14] introduced an algorithm that retrieves tuples that match a query string with a high probability. The SSJoin algorithm proposed by Arasu *et al.* [15] finds pairs of sets of high q-gram similarity in two set collections. The Ed-Join algorithm developed by Chuan Xiao *et al.* [16] uses mismatching q-grams to speed up the join process and reduce the computation time. In this paper we do not study new clustering techniques, but improve the scalability of an existing clustering approach, GPC.

6 Conclusions and Future Work

GPC is a string clustering method that automatically detects the cluster borders and was developed to cleanse non-dictionary string data [1]. To find the cluster border, the state-of-the-art methods compute a so-called proximity graph that is expensive to compute for datasets with long strings.

In this work, we proposed two new algorithms, PG-Skip and PG-Binary, that detect the cluster border in the partially computed proximity graph and thus reduce the computational costs. The PG-Skip method computes the *exact* GPC clustering, while PG-Binary approximates it, but performs much faster for long strings. We showed the optimality of PG-Skip among all prefix pruning algorithms and empirically evaluated our solutions in extensive experiments.

We plan to further improve the runtime of GPC by incrementally updating the neighborhoods in the proximity graph instead of computing each neighborhood from scratch.

References

1. Mazeika, A., Böhlen, M.H.: Cleansing databases of misspelled proper nouns. In: CleanDB (2006)
2. Kazimianec, M., Augsten, N.: Exact and efficient proximity graph computation. In: Catania, B., Ivanović, M., Thalheim, B. (eds.) ADBIS 2010. LNCS, vol. 6295, pp. 293–307. Springer, Heidelberg (2010)
3. Sarawagi, S., Kirpal, A.: Efficient set joins on similarity predicates. In: SIGMOD Conference, pp. 743–754 (2004)
4. Augsten, N., Böhlen, M., Dyreson, C., Gamper, J.: Approximate joins for data-centric XML. In: International Conference on Data Engineering (ICDE), Cancún, Mexico, pp. 814–823. IEEE Computer Society, Los Alamitos (2008)
5. Li, C., Lu, J., Lu, Y.: Efficient merging and filtering algorithms for approximate string searches. In: International Conference on Data Engineering (ICDE), Washington, DC, USA, pp. 257–266. IEEE Computer Society, Los Alamitos (2008)
6. Kazimianec, M., Mazeika, A.: Clustering of short strings in large databases. In: International Workshop on Database and Expert Systems Applications, pp. 368–372 (2009)
7. Li, C., Wang, B., Yang, X.: Vgram: improving performance of approximate queries on string collections using variable-length grams. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007, pp. 303–314, VLDB Endowment (2007)
8. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Cam, L.M.L., Neyman, J. (eds.) Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press, Berkeley (1967)
9. Kaufman, L., Rousseeuw, P.: Finding Groups in Data An Introduction to Cluster Analysis. Wiley Interscience, New York (1990)

10. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp. 226–231 (1996)
11. Ankerst, M., Breunig, M.M., Kriegel, H.-P., Sander, J.: Optics: Ordering points to identify the clustering structure. ACM SIGMOD Record 28(2), 49–60 (1999)
12. Kukich, K.: Techniques for automatically correcting words in text. ACM Comput. Surv. 24(4), 377–439 (1992)
13. Hodge, V.J., Austin, J.: A comparison of standard spell checking algorithms and a novel binary neural approach. IEEE Trans. on Knowl. and Data Eng. 15(5), 1073–1081 (2003)
14. Chaudhuri, S., Ganjam, K., Ganti, V., Motwani, R.: Robust and efficient fuzzy match for online data cleaning. In: SIGMOD, pp. 313–324 (2003)
15. Arasu, A., Ganti, V., Kaushik, R.: Efficient exact set-similarity joins. In: VLDB 2006: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 918–929, VLDB Endowment (2006)
16. Xiao, C., Wang, W., Lin, X.: Ed-join: an efficient algorithm for similarity joins with edit distance constraints. Proc. VLDB Endow. 1(1), 933–944 (2008)

An Effective Approach for Searching Closest Sentence Translations from the Web

Ju Fan, Guoliang Li, and Lizhu Zhou

Department of Computer Science and Technology,
Tsinghua University, Beijing 100084, China
fan-j07@mails.thu.edu.cn, {liguoliang,dcszlz}@thu.edu.cn

Abstract. There are large numbers of well-translated sentence pairs on the Web, which can be used for translating sentences in different languages. It is an interesting problem to search the closest sentence translations from the Web for high-quality translation, which has attracted significant attention recently. However, it is not straightforward to develop an effective approach, as this task heavily depends on the effectiveness of the similarity model which is used to quantify the similarity between two sentences. In this paper, we propose several optimization techniques to address this problem. We devise a phrase-based model to quantify the similarity between two sentences. We judiciously select high-quality phrases from sentences, which can capture the key features of sentences and thus can be used to quantify similarity between sentences. Experimental results show that our approach has performance advantages compared with the state-of-the-art sentence matching methods.

1 Introduction

With the rapid growth of the Web, hundreds of millions of *parallel sentences* (i.e., sentences with their well-translated counterparts) are now available on the Web, forming a rich source for translation reference. The following English sentence with its Chinese translation from a Chinese Web page is an example (the translation is in a pair of parentheses): "The president says he knows the problem too well (总统说他非常了解这个问题)". Recently, utilizing the parallel sentences from the Web to build a sentence-level translation-aid (STA) system for producing high-quality translations has attracted much attention. Some commercial search engines, e.g., Youdao (http://dict.youdao.com) and Iciba (http://dj.iciba.com), have incorporated STA systems into their services.

An overview of a typical STA system is shown in Figure 1. A huge amount of parallel sentences are crawled, extracted from the Web, and stored in a parallel-sentence database. Given a source sentence to be translated by a translator, the Sentence Matching component searches for the most similar (or closest) sentences with their translations from the database. The translator then chooses some of the retrieved sentences, revises their translation counterparts if necessary, and then obtains the appropriate translation for the source sentence by limited revisions. For example, given a source sentence, "My elder sister is one year older than her husband", the Sentence Matching component may find the following two sentences: 1) "My elder sister is one year older than me (我姐姐比我大年龄一岁)", and 2) "My elder sister is older than her husband (我姐姐比他丈夫

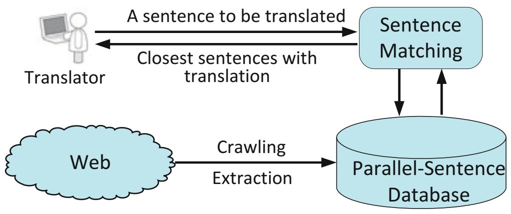

A sentence to be translated
Closest sentences with translation
Translator
Sentence Matching
Web
Crawling
Extraction
Parallel-Sentence Database

Fig. 1. An overview of a typical STA system

年龄大)". Although neither of the retrieved sentences is identical with the source sentence, the translator can reuse these two sentences to obtain the correct translation of the source sentence, "我姐姐比他丈夫年龄大一岁" from either of the Chinese translations of the two sentences with a very minor revision. Some translator surveys have shown that the STA systems are very effective and helpful to both translators and translation companies [5].

In this paper, we study the problem of sentence matching, i.e., searching the closest sentences with their translations for a source sentence, given a parallel-sentence database obtained from the Web. Note that we focus on English-to-Chinese parallel sentences with English as the source sentences to be translated for illustration purpose. Notice that it is not straightforward to develop a sentence matching approach with good performance, as the task heavily depends on the effectiveness of the similarity model that quantifies the similarity between two English sentences. Word-based models, e.g., Translation Models [10,12], the percentage of shared words [1] and edit distance [13], assume that words in the sentence are mutually independent, and thus they cannot capture the order of words and will lead to low performance. N-gram model [4], one of the Markov models, segments a sentence into a sequence of N-length grams, and makes use of the local dependencies among words. However, it does not consider the syntactic information, which is rather crucial for sentence matching for translation. Cancedda et al. [3] propose the word sequence kernel, which selects all subsequences of sentences and computes the inner product on top of the subsequences to measure the similarity. Although this model has the sound theoretical background, it is very expensive to select all subsequences. To address this problem, we devise a phrase-based similarity model between the source sentence to be translated and the sentences in the database, where a phrase is a subsequence of a sentence. We judiciously select high-quality phrases by considering both the dependency syntax [9] and the frequencies of phrases. The selected phrases can capture the syntactic and frequency features of sentences and thus can be used for effective sentence matching.

The contributions of our paper are summarized as follows: 1) We propose an effective method for searching sentence translations from the Web. 2) We introduce a phrase-based similarity model and propose a method to judiciously select the high-quality phrases. 3) We conducted extensive experiments on two real data sets from the Web, and the experimental results show that our method outperforms existing methods.

The rest of this paper is organized as follows. The related work of sentence matching techniques is briefly reviewed in Section 2. We discuss our similarity model in Section 3 and the phrase-selection algorithms in Section 4. In Section 5 we report our experiments, and finally we conclude in Section 6.

2 Related Work

Sentence matching, a.k.a., sentence retrieval, which retrieves sentences for some requirements, has been widely studied in the research community. Word-based sentence matching methods assume that words in the sentence are mutually independent [12,10], which is only a simplification in mathematics. Some studies discussed the relationship of words. Metzler and Croft [11] examined the sequential dependencies of words using a Markov assumption. However, the Markov assumption is also too strong. Therefore, some researchers used a syntax parser to analyze the structure of each sentence [6,2], and then computed the dependencies of words. Some similarity measures combine the above-mentioned ideas. They used sequences of words (i.e., phrases) to replace words by taking into account the order of words. The words in a phrase are closely dependent with each other, and phrases are assumed to be mutually independent. Cancedda et al. [3] proposed the word sequence kernel which extracts all n-length subsequences. Li et al. [7] proposed *VGRAM* (variable-length grams) to select high-quality grams to represent an string (This technique can also been applied to sentences). Hiroshi Ichikawa et al. examined syntactic impacts and manually annotated the syntactic structure of a sentence [6], which is not capable of a large number of sentences.

Our approach introduces the syntactic and frequency information to the similarity measure, and can be seen as an extension of the VGRAM based method [7] and word sequence kernel [3]. On the one hand, we generalize the variable-length grams to the phrases by allowing gaps between words. On the other hand, we incorporate the *syntactic gap constraint*, the *maximum constraint* and the *frequency constraint* to select high-quality phrases to avoid the exponential combination of words in [3].

3 Similarity Models for Sentence Matching

Sentence matching essentially involves a similarity model between a source sentence s to be translated and a sentence u in the parallel-sentence database $\mathcal{U}$. As we use an ordered set of phrases/words to represent a sentence, we use the similarity between two ordered sets to quantify the similarity between the two sentences. Given a source sentence to be translated, we compute the similarity score of every sentence in $\mathcal{U}$ to the source sentence, and suggest the sentences with the highest scores. In this paper, we concentrate on the effectiveness of sentence matching, and take the efficient matching problem as a future work.

Commonly used similarity models between two ordered sets are Jaccard coefficient, cosine distance, and edit distance. Note that we use the sentences in Table 1 as a running example. Suppose we obtain three parallel sentences, u_1, u_2 and u_3 from the Web, and store them in the database. We take s as a source sentence to be translated.

We firstly introduce the three commonly used similarity models. The Jaccard coefficient between s and u is defined as $Sim_J(s,u) = (\mid \mathcal{S}_s \cap \mathcal{S}_u \mid)/(\mid \mathcal{S}_s \cup \mathcal{S}_u \mid)$, where $\mathcal{S}_s$ ($\mathcal{S}_u$) denotes the set of phrases of sentence s (u). Cosine distance is summarized as $Sim_c(s,u) = (\sum_{\bar{f}_i \in \mathcal{S}_s \cap \mathcal{S}_u} w(\bar{f}_i))/(\sqrt{\mid \mathcal{S}_s \mid} \cdot \sqrt{\mid \mathcal{S}_u \mid})$, where $\bar{f}_i$ is an element in the ordered set, $\mathcal{S}_s \cap \mathcal{S}_u$, and $w(\bar{f}_i)$ is the weight of $\bar{f}_i$. The edit distance between two ordered sets is the minimum number of operations, such as insertion, deletion, and substation, required to transform one set into the other.

Table 1. An example (s is the source sentence and $u_1 \sim u_3$ are parallel sentences in the database)

	Sentence	**Translation**
s	He has eaten an apple	–
u_1	He has an apple	他有一个苹果
u_2	He ate a red apple	他吃了一个红苹果
u_3	He has a pencil	他有一支铅笔

The three models focus on common phrases/words but cannot capture the syntactic information of a sentence. Alternatively, we propose a phrase-based similarity model in Equation (1). The model is based on two factors. One is the percentage of shared phrases, and the other is the sum of their syntactic weights.

$$Sim_p(s,u) = \frac{|\mathcal{S}_s \cap \mathcal{S}_u|}{|\mathcal{S}_u|} \cdot \sum_{\bar{f}_i \in \mathcal{S}_s \cap \mathcal{S}_u} \phi(s,\bar{f}_i)\phi(u,\bar{f}_i)w(\bar{f}_i) \tag{1}$$

where $\frac{|\mathcal{S}_s \cap \mathcal{S}_u|}{|\mathcal{S}_u|}$ is the percentage of the shared phrases between sentences s and u, $\phi(s,\bar{f}_i)$ (or $\phi(u,\bar{f}_i)$) is the *importance* of $\bar{f}_i$ to the sentence s (or u), and $w(\bar{f}_i)$ is the weight of $\bar{f}_i$.

As phrase $\bar{f}_i$ can be discontinuous, its *importance* to s contains two factors: the matched words and the words in *gaps* which are composed of the unmatched words in the phrase. Take a sentence "`He has eaten an apple`" and a phrase "`has apple`" as an example. "`eaten an`" is a gap. We use λ_α to denote the syntactic weight of a matched word α and λ_β to denote the penalty of an unmatched word β in gaps. $\phi(s,\bar{f}_i)$ can be computed as $\phi(s,\bar{f}_i) = \prod_{1 \leqslant j \leqslant m} \lambda_{\alpha_j} \prod_{1 \leqslant k \leqslant n} \lambda_{\beta_k}$, where m is the number of matched words and n is the number of unmatched words in gaps.

Apparently, our model Sim_p is a generalization of cosine distance and Jaccard coefficient, and incorporates the syntactic feature $\phi(s,\bar{f}_i)$ (or $\phi(u,\bar{f}_i)$) into the computation of the similarity. We will experimentally prove that our proposed similarity model achieves the best performance in Section 5.

Now we discuss how to estimate the parameters introduced in the proposed model. Since λ_α is the syntactic weight of a matched word α, we employ the dependency syntax [9] to estimate this weight. In the dependency syntax, a sentence is parsed as a tree structure, where each node is a word in the sentence and an edge between the parent node and the child node represents a *dependency relationship* between the two corresponding words. More specifically, the child word is the modifier of the parent word. In particular, the root of the tree is the predicate of the sentence. For example, consider a sentence "`He has eaten an apple`'. In the parsed tree, the root is the predicate, "`eat`", and the root has three child nodes, i.e., "`he`", "`have`", and "`apple`". In addition, "`an`" is the child node of "`apple`", as the former is the article of the latter. We assume that the ancestors are more important than the decedents in terms of syntax. According to this rule, we introduce a decay factor d ($0 < d \leqslant 1$) to ensure that the weight of a child α_c is smaller than its parent α_p by the decay factor, i.e., $\lambda_{\alpha_c} = d \cdot \lambda_{\alpha_p}$. On the other hand, λ_β is set to a constant for penalizing the unmatched words.

In addition, given a phrase, we use its inverse document frequency (IDF) in $\mathcal{U}$ to estimate its weight, i.e., $w(\bar{f}_i) = \log \frac{|\mathcal{U}|}{|\{u_j | \bar{f}_i \in u_j\}|}$ where $|\mathcal{U}|$ is the size of $\mathcal{U}$, and $\bar{f}_i \in u_j$ means that u_j ($u_j \in \mathcal{U}$) contains the phrase $\bar{f}_i$.

4 High-Quality Phrase Selection

In this section, we discuss an effective method to select high-quality phrases from sentences. We firstly give the formulation of high-quality phrases in Section 4.1, and then develop efficient algorithms to generate such phrases from sentences in Section 4.2.

4.1 High-Quality Phrases

As selecting all subsequences of a sentence as phrases is rather expensive, we propose a heuristic approach to select *the infrequent phrases with syntactic gap constraints* as the high-quality phrases. We restrict the gaps of discontinuous phrases with syntactic rules: if two discontinuous words of a sentence are selected, they must have syntactic dependency relationships (Section 3). Consider the sentence "`he has eaten an apple`" and the phrase "`he eaten an apple`" that has a gap "`has`". The phrase is meaningful because "`he`" and "`eaten`" have a dependency relationship. In contrast, "`has apple`" should be eliminated because "`has`" is independent on "`apple`". We also select phrases according to its frequency. Phrases with high frequencies have small IDF scores, thus they are less important than the infrequent ones according to our similarity model. Therefore we eliminate them in order to improve the efficiency.

Now, we formally define the high-quality phrase. Let a sentence s be a word sequence $s = s_1 s_2 \dots s_{|s|}$, where $\mid s \mid$ is the length of s and s_i is the i-th word of s ($1 \leqslant i \leqslant \mid s \mid$). Notice that all words are stemmed and stop-words are eliminated (We maintain a subset of commonly used stop-word list where some playing important roles in syntax, i.e., "he" and "I", are excluded). We denote a phrase as $\bar{f}[I] = s_{i_1} s_{i_2}, \dots, s_{i_n}$ where $I = [i_1, i_2, \dots, i_n]$ ($1 \leqslant i_1 < i_2 < \cdots < i_n \leqslant \mid s \mid$). The infrequent phrase with syntactic gap constraints (i.e., high-quality phrases) is defined in Definition 1.

Definition 1 (High-Quality Phrases). *$\bar{f}[I]$ is a high-quality phrase if it satisfies:*

1. Gap constraint: If $i_{j+1} \neq i_j + 1$ ($1 \leqslant j < n$), $s_{i_{j+1}}$ and s_{i_j} must have a dependency relationship, and

2. Frequency constraint: $\bar{f}[I]$ is infrequent, that is, its frequency is smaller than a given threshold which is determined empirically, and

3. Maximum constraint: $\bar{f}[I]$ is not a prefix of any other phrases.

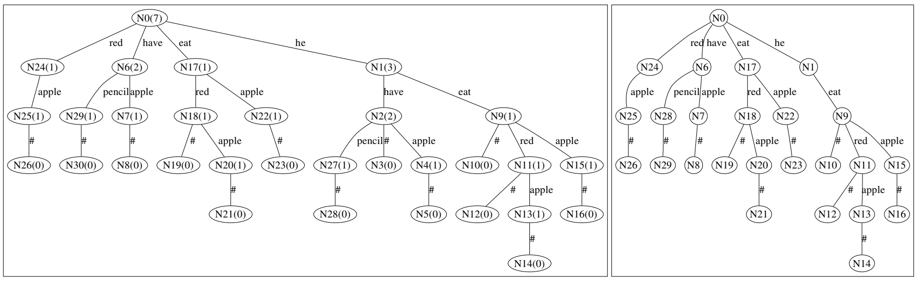
Take the sentence "`he ate a red apple`" in Table 1 as an example. We preprocess it into "`he eat red apple`". Suppose that the given frequency threshold is 2. Consider the phrase "`eat apple`". Firstly, it satisfies the gap constraint: "`apple`" is the object of the verb "`eat`". Secondly, the frequency of this phrase in u_1, u_2, and u_3 in Table 1 is 1, which validates the frequency constraint. Thirdly, it is the longest phrase started with "`eat`", thus the maximum constraint also holds. So this phrase is a high-quality phrase.

4.2 Generating High-Quality Phrases from a Sentence

Selecting Phrases with Gap and Maximum Constraints. As the high-quality phrases must satisfy gap constraint, we describe an algorithm to generate such phrases for a sentence. The aim of this model is extracting the sequential and syntactic relationships between words and modeling these relationships into a graph. A plain sentence model (a sequence of words) cannot reach this point because we allow gaps in the phrases. Instead, we use the Minipar [8] to preprocess the sentence s and model it as a S-Graph, a directed acyclic graph, where each node represents a word in the sentence s, and there is an edge between two nodes, say s_i and s_j, if $j = i + 1$ or s_i has a dependency relationship with s_j. Obviously, the longest path started with one node in the graph corresponds to a phrase with gap and maximum constraints.

Selecting Phrases with Frequency Constraint. As the phrases must satisfy frequency constraint, we have to consider all the sentences in the database $\mathcal{U}$ to generate high-quality phrases of a sentence. We use a *frequency-trie* structure to collect frequencies of all possible high-quality phrases. We develop a two-step algorithm to achieve our goal as follows. In the first step, we use a frequency-trie to collect the frequencies of the phrases, and then we traverse the trie to prune frequent phrases.

Step1: Collecting the Phrase Frequencies: Each node in the frequency-trie has a number to represent the frequency of the phrase corresponding to the path from root node to this node (Note that we set the value in a leaf node as 0). Figure 2 (a) displays an example. The value 2 on node N2 denotes that the frequency of "`he have`" is 2.



(a) The trie for all possible phrases. (b) The trie for infrequent phrases.

Fig. 2. The frequency tries corresponding to u_1, u_2 and u_3 in Table 1

We initialize the root node of the frequency-trie as 0. For each S-Graph $\mathcal{G}$ modeled from s, we collect all paths from the graph node n_i, which correspond to the phrases started with s_i. We examine whether these phrases are prefixes of the generated phrases. If not, they can be inserted into the phrase set P, and their prefixes in P which are previously generated should be removed. Thus the phrases in P satisfy both gap constraint and maximum constraint. Then, for each phrase in P, we insert it to the frequency-trie and increase the frequency for each node on the corresponding path. Next, for each node (excluding the root) on the path, we add a leaf node with an endmarker # to them if the leaf node does not exist to distinguish a phrase from its extended ones. Take u_2 in Table 1 as example. One longest phrase started with "`he`" is "`he eat apple`". It is inserted into the frequency-trie as shown in Figure 2 (a). The frequencies of nodes N1, N9 and N15 are increased and a new leaf node N10 is appended for the prefix "`he eat`".

Step2: Selecting Infrequent Phrases: The frequency-trie collects all the phrases satisfying the syntactic gap constraint and maximum constraint. In this step, we select the phrases based on the frequencies. The algorithm traverses the nodes of the frequency-trie in pre-order. For each node n with leaf node (which means there is a phrase corresponding to the path from the root to n), if its frequency is not smaller than δ, we delete this node and recursively delete its children; otherwise, we recursively visit all of its children. According to this algorithm, given the threshold $\delta = 2$, the frequency-trie in Figure 2 (a) is converted to the trie in Figure 2 (b). For example, node N2 and its descendants are eliminated since N2's frequency is not smaller than δ. N1 and N17 should not be deleted, because they are infrequent or have no leaf node.

5 Experiments

In this section, we evaluate the performance of our proposed sentence matching method. We obtained two sets of parallel sentences from the Web to evaluate our proposed methods. ICIBA (http://www.iciba.com), denoted as D_I, contains 520,899 parallel sentences, most of which are expressions for everyday use. The average length of the English sentences is 13.2 words. CNKI (http://dict.cnki.net), denoted as D_C, contains 800,000 parallel sentences extracted from the academic papers. The average length of the sentences is 20.5 words. For each data set, we randomly selected 100 sentences as source sentences to be translated, and took the other sentences in the database $\mathcal{U}$.

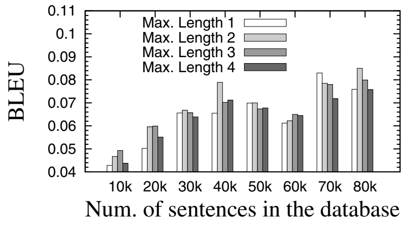
We compared our model with the three models on top of the selected phrase set: Jaccard coefficient, Cosine distance, and Edit distance. We also evaluated our proposed method with state-of-the-art sentence-matching methods. Translation Model based methods (TM) estimate the probability that one sentence is a translation of another by using a translation model, and take the probability as the similarity [10,12]. We implemented the state-of-the-art translation model with its optimal parameters in [12]. Variable-length gram based methods (VGRAM) select variable-length grams, rather than fixed-length grams, as features of sentences [7]. We implemented the VGRAM and computed the cosine similarity with TF-IDF weights between gram sets.

We exploited the most popular metric *BLEU* in machine translation to evaluate the sentence-matching effectiveness. BLEU measures the similarity between the translation output and a reference translation using the N-gram strategy. Thus, it can measure the revision efforts of translators that transform the translation of the retrieved sentences to the reference translation. The BLEU score was calculated by the NIST script with default settings (http://www.nist.gov/speech/tests/mt/2008/scoring.html).

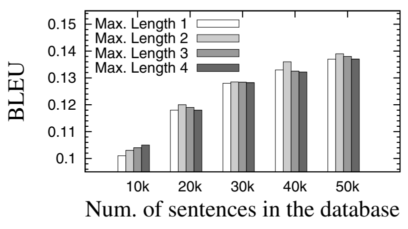
All the programs were implemented in JAVA and all the experiments were ran on a computer with an Intel Core 2 CPU 1.87 GHz, 2 GB RAM and 100 GB disk.

5.1 Effects of High-Quality Phrase Selection

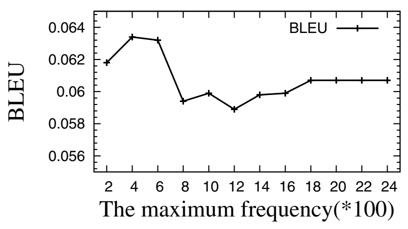
We first evaluated the effects of selecting phrases with different maximum length thresholds. Figures 3(a) and 3(b) illustrate the BLEU scores of phrases with different maximum length thresholds. We observe that selecting too long phrases does not yield too high performance. Surprisingly, the highest performance is achieved at the maximum length thresholds of 2 or 3. This can be explained by the maximum constraint introduced in Section 4.1. This constraint guarantees that each generated phrase is not a



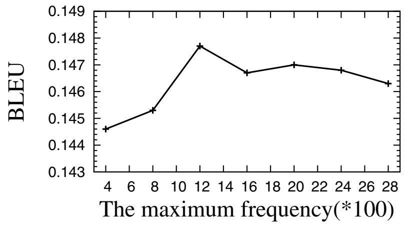
(a) Effect on max. lengths on D_I.



(b) Effect on max. lengths on D_C.



(c) Effect on freq. thresholds on D_I.



(d) Effect on freq. thresholds on D_C.

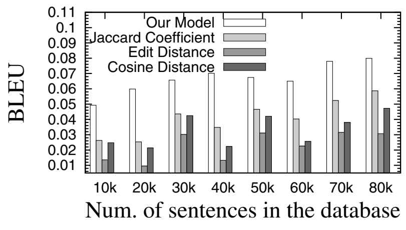
Fig. 3. Effect of Phrase Selection

prefix of others. Therefore selecting too long phrases will eliminate many other possible high-quality ones, which will affect the performance. As the phrases must satisfy the frequency constraint, we also examined the effect of different maximum frequency thresholds. Figures 3(c) and 3(d) give the experimental results on the two data sets. As the threshold increases, the performance first increases and then decreases at a fixed point (No phrase is eliminated given a large enough threshold). It indicates that selecting phrases within a maximum frequency indeed improves the performance of sentence matching. However, too many phrases will be eliminated if this threshold is too small.

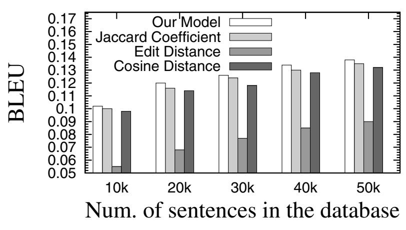
5.2 Effectiveness Comparisons

Comparison of Similarity Models. We compared the effectiveness of Jaccard coefficient, edit distance, cosine distance, and our model (Section 3) on top of the selected phrase sets. Observed from Figures 4(a) and 4(b), our model achieves the highest BLEU score for various numbers of sentences in the database. This is because our model introduces the syntactic importance of each phrases by considering the matched words and unmatched words in the gaps, compared with Jaccard coefficient and cosine distance which only count the number (or weights) of shared phrases. The performance of edit distance is the worst. Because taking the minimum number of operations of transforming one sentence to the other as the similarity is not very useful for translation.

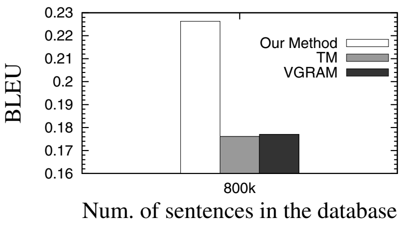
Comparison with Existing Matching Methods. We compared our proposed method with two baseline methods, TM and VGRAM. Figures 4(c) and 4(d) display their performance. Our method achieves the best BLEU scores and outperforms the baseline methods significantly. For example, the BLEU score of our method is 37% higher than that of TM in Figure 4(c), and is 30% higher than that of VGRAM in Figure 4(d). The



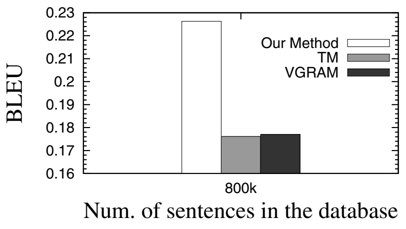
(a) Model comparison on D_I.



(b) Model comparison on D_C.



(c) Approach comparison on D_I.

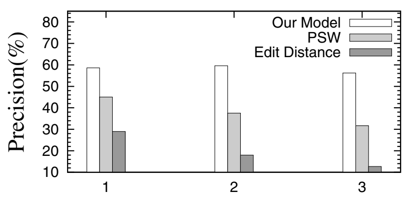


(d) Approach comparison on D_C.

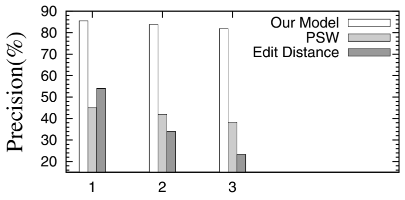
Fig. 4. Performance Comparison

improvement of our approach is mainly due to the selected high-performance phrases. TM assumes that each word translates by itself [12,10] and neglects the order of words. VGRAM only considers continuous words, and misses some discontinuous phrases that reflect syntactic features of a sentences for translation. For example, consider the sentence, “he have eaten an red apple”. The discontinuous words “he eat apple” is very important for translation. Unfortunately, they cannot be selected as features of the sentence by VGRAM. In contrast, our method allows discontinues words (i.e., phrases), and proposes effective rules to select high-quality phrases by considering syntactic and frequency information. The experimental results show that such information really plays an important role in sentence similarity measurement, and the selected phrases are more powerful in capturing the key features of sentences. Secondly, compared with the translation probability used by TM and the standard cosine model with the TF-IDF weight used by VGRAM, our proposed similarity model can further exploit the matched words and unmatched words in gaps in the selected phrases and then improve the sentence matching for translation.

User Studies. We conducted user studies to evaluate the usefulness of our sentence matching method. We compared with the PSW (Percentage of Shared Words) and the edit distance, which are widely used in commercial STA systems (e.g., Youdao and Trados) and Translation Memory systems [13] respectively. We asked 10 volunteers to annotate whether each retrieved sentence translation is helpful for the translation of the source sentence. Then, we averaged their results and computed the *precision* according to these labels. Observed from Figures 5(a) and 5(b), our method outperforms the two baseline approaches significantly. For example, the precision at position 1 (i.e., $P@1$) of our method is nearly 50% higher than that of baseline methods in Figure 5(b). The



(a) Precision over 3 positions on D_I.



(b) Precision over 3 positions on D_C

Fig. 5. The user study on the two data sets, D_I and D_C

results show that our proposed methods are effective and helpful for assisting users to produce high-quality translations.

6 Conclusion

In this paper, we have proposed an effective approach for searching closest sentence translations from the Web. We introduced a phrase-based sentence similarity model and selected high-quality phrases from sentences for effectively quantifying the similarity between sentences. We have implemented our method and built an STA system, where parallel sentences are crawled and extracted from the Web. We have conducted extensive experiments to benchmark our method, and the experiment results show that our approach achieves high result quality, and outperforms state-of-the-art methods.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under grant No. 61003004 and No. 60833003.

References

1. Biçici, E., Dymetman, M.: Dynamic translation memory: Using statistical machine translation to improve translation memory fuzzy matches. In: CICLing, pp. 454–465 (2008)
2. Cai, K., Bu, J., Chen, C., Liu, K.: Exploration of term dependence in sentence retrieval. In: ACL (2007)
3. Cancedda, N., Gaussier, E., Goutte, C., Renders, J.: Word sequence kernels. The Journal of Machine Learning Research 3, 1059–1082 (2003)
4. Damerau, F.: Markov models and linguistic theory: an experimental study of a model for English. Mouton De Gruyter, Berlin (1971)
5. Garcia, I.: Power shifts in web-based translation memory. Machine Translation 21(1), 55–68 (2007)
6. Ichikawa, H., Hakoda, K., Hashimoto, T., Tokunaga, T.: Efficient sentence retrieval based on syntactic structure. In: ACL (2006)
7. 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)

8. Lin, D.: Dependency-Based Evaluation Of Minipar. Treebanks: Building and Using Parsed Corpora (2003)
9. Mel'cuk, I.: Dependency Syntax: Theory and Practice. State University of New York Press (1988)
10. Metzler, D., Bernstein, Y., Croft, W.B., Moffat, A., Zobel, J.: Similarity measures for tracking information flow. In: CIKM, pp. 517–524 (2005)
11. Metzler, D., Croft, W.B.: A markov random field model for term dependencies. In: SIGIR, pp. 472–479 (2005)
12. Murdock, V., Croft, W.B.: A translation model for sentence retrieval. In: HLT/EMNLP (2005)
13. Planas, E., Furuse, O.: Multi-level similar segment matching algorithm for translation memories and example-based machine translation. In: COLING, pp. 621–627 (2000)

Finding the Sites with Best Accessibilities to Amenities

Qianlu Lin, Chuan Xiao, Muhammad Aamir Cheema, and Wei Wang

The University of New South Wales, Australia
{qlin,chuanx,macheema,weiw}@cse.unsw.edu.au

Abstract. Finding the most accessible locations has a number of applications. For example, a user may want to find an accommodation that is close to different amenities such as schools, supermarkets, and hospitals etc. In this paper, we study the problem of finding the most accessible locations among a set of possible sites. The task is converted to a top-k query that returns k points from a set of sites R with the best accessibilities. Two R-tree based algorithms are proposed to answer the query efficiently. Experimental results show that our proposed algorithms are several times faster than a baseline algorithm on large-scale real datasets under a wide range of parameter settings.

1 Introduction

Optimal location problems have received significant research attention in the past[9, 21]. In this paper, we study a new problem that finds the sites with the best accessibilities to amenities. Consider the example of a person who wants to rent an apartment. He may be interested in finding an apartment such that different amenities are close to it (e.g., it has a restaurant, a bus stop and a super market nearby). The person may specify different types of amenities that are of his interest. The accessibility of an apartment can be defined based on its closest amenity of each type. Furthermore, the person may define a scoring function such that it gives higher priority to certain types of amenities (e.g., a nearby bus stop may be more important than a nearby restaurant). We allow the users to define a monotonic scoring function to set such preferences. Formal definition is given in Section 2.

Similar to the existing work on the other versions of facility location problems [9, 21], our focus is on solving the problem in Euclidean space. Also, we focus on the case where the accessibility of a site depends on the closest amenity of each type. Nevertheless, we remark that the pruning rules presented in this paper can be extended to the case where the accessibility depends on m-closest amenities of each type.

Several approaches have been proposed to solve the all nearest neighbors problem (ANN) [22, 7, 10] and aggregate nearest neighbor problem [15, 14, 16]. However, these techniques only consider one type of amenity and cannot be efficiently applied to our problem. Nevertheless, we use these techniques to design a baseline algorithm and show that our proposed algorithms perform significantly better than the baseline algorithm.

We propose two efficient R-tree based algorithms. The first algorithm constructs indexes for different types of amenities in separate R-trees, and traverses the R-trees in parallel to progressively output top-k query results. The second algorithm indexes the different types of amenities in a single R-tree and demonstrates better performance in most of the settings. Both algorithms carefully exploit the lower bound of the accessibility scores with several non-trivial optimizations applied.

Another important feature of our algorithms is that we progressively report the best sites in an order of their accessibility scores. Such progressive/incremental answer may be useful in many interactive applications and a user may terminate the algorithm if he is satisfied with first j results (where $j < k$).

Below we summarize our contributions.

– We proposed two algorithms to find top-k accessible sites among a set of possible locations. Unlike traditional algorithms, our algorithms are able to compute the results progressively.
– We developed several non-trivial pruning and optimization techniques that can be integrated into the two proposed algorithms in order to reduce the I/O cost and running time.
– We performed experiments on several real datasets. Our proposed algorithms are shown to outperform the baseline algorithm in all settings.

The rest of the paper is organized as follows: Section 2 gives the problem definition and introduces a baseline algorithm to the problem. We propose two main algorithms in Section 3. Several optimizations to the main algorithms are presented in Section 4. We present and analyze experimental results in Section 5, and survey related work in Section 6. Section 7 concludes the paper.

2 Preliminaries

In this section, we define the problem and introduce a baseline algorithm based on R-trees.

2.1 Problem Definition

We define R and S as two spatial datasets, and each point s in S is assigned a type i. Let $|T|$ be the total number of types. A distance metric $d(r, s)$ measures the Euclidean distance between two points r and s. For a point $r \in R$, let $NN(r, S_i)$ be the nearest point of type i from r. The accessibility cost c_r of r is given by the formula below;

$$c_r = f(d(r, NN(r, S_1)), \cdots, d(r, NN(r, S_{|T|})). \tag{1}$$

where $f()$ is a monotonic scoring function that takes $|T|$ values as parameters and returns a single value[1]. Our goal is to find k points from R, such that their accessibility costs are the smallest among all the points in R.

2.2 All Nearest Neighbor Algorithm

An immediate solution to the proposed top-k problem is to borrow the techniques from all nearest neighbor queries [10, 22, 7, 12, 2, 8]. For each point r in R, we enumerate its nearest neighbors of each type in S, compute the accessibility cost for r, and then output the k points with the smallest accessibility costs. Existing algorithms for computing all nearest neighbor queries presume the data points are indexed by an R-tree [11] and use various optimizations to reduce the search space.

First, we build an R-tree I_R on the points in R, and for each type of points in S, we build a separate R-tree I_{S_i}. Hence there are $|T| + 1$ R-trees. We then probe these R-trees, starting from their roots.

In order to determine the access order of the nodes, we employ the following two data structures:

Local Priority Queue (LPQ). Each node u in I_R owns exactly one LPQ, a min-heap that maps its entries to the nodes in I_{S_i}. Each entry v in the priority queue has two values `mind` and `maxd`, indicating the minimum distance from u's MBR to v's MBR, and the maximum distance from u's MBR to v's MBR. Figure 1 shows an example of two MBRs and their `mind` and `maxd`. The priority queue orders its entries by increasing `mind` values, while `maxd` values are used for pruning unpromising entries. An LPQ also keeps two values `minmind` and `minmaxd`, representing the smallest of the `mind` values among its entries, and the smallest of the `maxd` values among its entries, respectively.

Global Priority Queue (GPQ). A min-heap maintains all the LPQs that are generated when accessing the R-trees, ordered by increasing `minmind`. `minmaxd` is used for pruning the LPQs that are guaranteed not to produce any nearest neighbors.

LPQ and GPQ allow us to access the nodes with smallest lower bound of distance first, which are the most promising nodes. Additionally, we apply pruning techniques to avoid accessing the nodes that cannot generate nearest neighbors to the points in R. We only allow one LPQ for each node in I_R, so that accessing duplicate nodes in I_R can be avoided.

The all nearest neighbor algorithm iterates through all the types in S. Within each iteration for a type i, it starts with the root of I_R and I_{S_i}, and expands the nodes in a bi-directional fashion [7, 19][2]. The nearest neighbors are returned

[1] For sake of simplicity, in rest of the paper, we consider that the monotonic scoring function returns the sum of these $|T|$ values. However, we remark that our algorithms can be applied on any monotonic scoring function.

[2] Although there exist alternative ways to expand nodes in R-trees, we choose bi-directional expansion because it is shown to outperform others in extensive experiments [7].

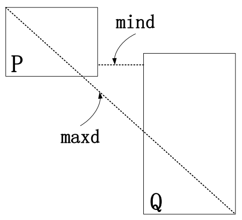


Fig. 1. Illustration of `mind` anx `maxd`

if the point level is reached in both I_R and I_{S_i}. After finish the tree expanding for all the types in S, the k points with smallest accessibility costs are output as final results to the top-k query.

The all nearest neighbor algorithm sequentially processes the points in S according to types. The drawback is that I_R will be traversed $|T|$ times, and we cannot obtain any results until all the types are processed. In addition, the algorithm does not exploit the abundant information provided by the top-k results, and hence the pruning power is very limited. In Section 3, we will show how to traverse the R-trees simultaneously and output top-k results progressively, as well as exploit the inherent information contained in these results.

3 Main Algorithm Frameworks

In this section, we give two basic algorithms to compute the k locations with the smallest accessibility cost.

3.1 Separate-Tree Method

The first algorithm is to choose the same indexes as the all nearest neighbor algorithm, but traverse these trees in a parallel fashion. Since different types of points in S are indexed in separate R-trees, we call this approach separate-tree algorithm.

We still choose LPQ and GPQ as the data structures in separate-tree algorithm. However, each entry u in I_R owns $|T|$ LPQs in separate-tree algorithm because we expand the entries in R-trees in a parallel way. We call these LPQs u's LPQ group, still denoted LPQ_u, with which we are able to estimate the lower bound of the accessibility costs for the points indexed in u. In u's LPQ group, we denote $LPQ_u[i]$ the LPQ that maintains entries from I_{S_i}, and calculate the lower bound as

$$LB_c = \sum_{i=1}^{|T|} LPQ_u[i].\texttt{minmind}.$$

A major difference from the all nearest neighbor method is that we arrange the LPQs pushed to GPQ by increasing order of LB_c. A fixed sized min-heap M is used to keep the top-k results seen so far, and $M[k]$ gives the temporary result with k-th smallest accessibility cost. Before expanding entries to form new LPQ groups, we compare the new LPQ groups' LB_c with $M[k]$'s accessibility cost, and allow only those whose LB_c are smaller than $M[k]$'s accessibility cost. Otherwise they are guaranteed not to produce any results that can beat the current temporary results.

Algorithm 1. SeparateTree (I_R, I_S)

```
1  for each $point\ r\ in\ R$ do
2      $c_r \leftarrow 0$;
3  $M \leftarrow$ InitializeTempResults ;   /* Store any $k$ points as initial results */
4  $GPQ \leftarrow \emptyset$;
5  for $i = 1$ $to$ $|T|$ do
6      $u \leftarrow I_R.root$; $v \leftarrow I_{S_i}.root$;
7      $LPQ_u[i] \leftarrow \emptyset$;
8      $LPQ_u[i]$.minmind $\leftarrow +\infty$; $LPQ_u[i]$.minmaxd $\leftarrow +\infty$;
9      SepTreePushAndUpdate($LPQ_u[i], v$);
10 $GPQ.push(LPQ_u)$;
11 while $GPQ \neq \emptyset$ do
12     $LPQ_u \leftarrow GPQ.pop()$;
13     SepTreeExpandTrees($LPQ_u, GPQ$);
```

Algorithm 1 describes this parallel algorithm. We initialize the min-heap M by choosing any k points as initial temporary results. These points are computed for their all type nearest neighbors and accessibility costs. Like the all nearest neighbor algorithm, the separate tree algorithm starts with the roots of I_R and all I_{S_i}, and then expands the nodes in a bi-directional fashion. The first LPQ group formed is owned by I_R's root, and the root of I_{S_i} is inserted into priority queues (Line 6 – 9). Then this LPQ group is pushed into a GPQ (Line 10). We iteratively select an LPQ group from the GPQ, and expand nodes in both R-trees I_R and I_{S_i}.

Algorithm 2. SepTreePushAndUpdate $(LPQ_u[i], v)$

```
1  if mind$(u, v) < LPQ_u[i]$.minmaxd then
2      $LPQ_u[i].push(v)$;
3      $LPQ_u[i]$.minmind $\leftarrow \min(LPQ_u[i]$.minmind, mind$(u, v))$;
4      $LPQ_u[i]$.minmaxd $\leftarrow \min(LPQ_u[i]$.minmaxd, maxd$(u, v))$;
5      $LPQ_u.LB_c \leftarrow \sum_{i=1}^{|T|} LPQ_u[i]$.minmind
       ;                    /* update lower bound of accessibility cost */
```

The expansion algorithm is shown in Algorithm 3. Given a node u in I_R, the entries in u's LPQ group are popped, according to the order of `mind`. We identify the nearest neighbor, add the distance to the the accessibility cost, and update

Algorithm 3. SepTreeExpandTrees (LPQ_u, GPQ)

```
1  if $u$ is a point then
2      for $i = 1$ to $|T|$ do
3          while $LPQ_u[i] \neq \emptyset$ do
4              $v \leftarrow LPQ_u[i].pop()$;
5              if $v$ is a point then
6                  $c_u \leftarrow c_u + d(u, v)$;
7                  if $u$'s NNs of all types are found and $c_u < M[k].cost$ then
8                      $M.add(u, c_u)$ ;                    /* update temp results */
9                  return
10             else
11                 for each $v' \in v$ do
12                     SepTreePushAndUpdate($LPQ_u[i], v'$);
13     if $LPQ_u.LB_c < M[k].cost$ then $GPQ.push(LPQ_u)$;
14 else
15     for each $u' \in u$ do
16         $LPQ_u[i]' \leftarrow \emptyset$; $LPQ_u[i]'$.minmind $\leftarrow +\infty$; $LPQ_u[i]'$.minmaxd $\leftarrow +\infty$;
17     for $i = 1$ to $|T|$ do
18         while $LPQ_u[i] \neq \emptyset$ do
19             $v \leftarrow LPQ_u[i].pop()$;
20             if $v$ is a point then
21                 for each $u' \in u$ do
22                     SepTreePushAndUpdate($LPQ'_u[i], v$);
23             else
24                 for each $v' \in v$ do
25                     for each $u' \in u$ do
26                         SepTreePushAndUpdate($LPQ'_u[i], v'$);
27     for each $u' \in u$ do
28         if $LPQ_u.LB_c < M[k].cost$ then $GPQ.push(LPQ_u)$;
```

temporary results when point level is reached in both I_R and I_{S_i} (Line 8). Otherwise, the children of both u and popped entry v paired to form new LPQ groups. Specifically, for each type, we expand u and create a group for each of its children u', and the children of v is then inserted to the LPQ of u'. To avoid accessing the nodes that cannot generate any nearest neighbors for the points in u, we compare the nodes' `mind` to u' with the `minmaxd` of u'. Only if its `mind` is smaller than u''s `minmaxd`, we insert this node to u''s LPQ (Line 1, Algorithm 2). The values of `minmind` and `minmaxd` of the LPQ are updated once an entry is inserted, and finally we check the LB_c of new formed LPQ group before inserting it into the GPQ (Line 13 and 28), since the temporary results in M can be used to prune unnecessary LPQ groups. In addition, a temporary result is confirmed as a final result if its accessibility cost is smaller than the LB_c of the LPQ group popped from GPQ. The results are progressively output with the execution of the algorithm.

3.2 One-Tree Method

The above separate-tree method adopts the same indexing scheme as the all nearest neighbor algorithm. Here, we consider building indexes for the various types of points in S in one tree. Although the estimation of LB_c will be looser due to multiple types indexed in the nodes of R-trees, we are able to achieve a more efficient node expansion and hence better runtime performance.

We call this method one-tree algorithm. To record the type information, we add an attribute to the nodes of the R-tree built on S. This new attribute maintained in each node of I_S is a *type bitmap* B of length $|T|$ that indicates which types of points are contained in (the descendants of) the node. The bit i is set to 1 if the node or its descendants contain at least one point of type i, or 0 otherwise.

The one-tree algorithm follows the separate-tree algorithm framework, but differs in the generation of LPQs and lower bound estimations of accessibility costs. Now we allow an LPQ to store entries of various types. In order not to miss any real results due to abuse of types, the values of `minmaxd` are broken down into specific types. We use the notation `minmaxd`$[i]$ to capture the smallest of the `maxd` values among its entries that contain points of type i. Before inserting an entry to the LPQ, we check the type bitmap of the entry, and allow only the entries whose `mind` is smaller than `minmaxd`$[i]$ on at least one type i.

Similarly, `minmind`$[i]$ stores the smallest of the `mind` among the entries that contain points of type i. This is to estimate the lower bound of accessibility cost, as given by the following equation:

$$LB_c = \sum_{i=1}^{|T|} LPQ_u.\mathtt{minmind}[i].$$

Algorithm 4 captures the pseudo-code of forming new LPQs and updating LB_c in the one-tree algorithm. Before inserting entry v into u's LPQ, we check the types contained in v. If the pair (u, v) can produce final top-k results, there must be at least one type i such that v's `mind` is smaller than LPQ_u's `minmaxd` [i]. We use a boolean variable flag to capture whether such type can be found. If it is set to **true**, we insert v into u's LPQ, and make necessary updates.

4 Optimizations on Existing Algorithms

In this section, we introduce several pruning and optimization techniques that can be integrated into the two basic algorithm frameworks proposed in Section 3.

Algorithm 4. OneTreePushAndUpdate (LPQ_u, v)

```
1 $flag \leftarrow$ **false**;
2 **for each** type $i$ in $B_v$ **do**
3     **if** $\texttt{mind}(u, v) < LPQ_u.\texttt{minmaxd}[i]$ **then** $flag \leftarrow$ **true**;
4 **if** $flag =$ **true then**
5     $LPQ_u.push(v)$;
6     **for each** type $i$ in $B_v$ **do**
7         $LPQ_u.\texttt{minmind}[i] \leftarrow \min(LPQ_u.\texttt{minmind}[i], \texttt{mind}(u, v))$;
8         $LPQ_u.\texttt{minmaxd}[i] \leftarrow \min(LPQ_u.\texttt{minmaxd}[i], \texttt{maxd}(u, v))$;
9     $LPQ_u.LB_c \leftarrow \sum_{i=1}^{|T|} LPQ_u.\texttt{minmind}[i]$;
```

4.1 Break Ties in Priority Queues

In both separate-tree and one-tree algorithms, the entries in LPQs are arranged by increasing order of mind. Consider a node u in I_R, and an entry in its LPQ, v. If u's and v's MBRs are overlapped, the value of mind will become zero. This often happens for the high-level nodes in R-trees. To alleviate this problem, we break ties by choosing the one with the smaller maxd if two entries have the same mind.

The same problem may occur in GPQ. We compute the upper bound of the accessibility costs for the points indexed in u:

$$UB_c = \sum_{i=1}^{|T|} LPQ_u.\texttt{minmaxd}[i].$$

The LPQ with the smaller UB_c is chosen to break ties if there exist multiple LPQs that have the equal value of LB_c in GPQ.

4.2 Early Check to Avoid Unnecessary Expansion

In the basic separate-tree and one-tree algorithm, an entry v' is checked for its mind before it is inserted to u''s LPQ. The LPQ admits only the entries whose mind is smaller than the LPQ's current minmaxd.

Since the entry v' is expanded from its parent node v, we can check the mind between u' and v, and safely prune v' from u''s LPQ given that the $\texttt{mind}(u', v)$ is larger than or equal to the minmaxd of u'. This is because $\texttt{mind}(u', v) \leq \texttt{mind}(u', v')$.

We apply this optimization before expanding v by calculating the mind between u' and v, so that the access to v's children can be avoided if the minimum distance between the two MBRs is too large.

4.3 Pre-update Temporary Results

The third major optimization is based on the observation that all the points indexed in a node u may have smaller accessibility cost than the temporary result M, and the number of points indexed in u exceeds k. We can verify this by comparing UB_c, the upper bound of the accessibility costs for the points indexed in u, and the k-th temporary result's cost. If the upper bound is smaller, the top-k temporary results are to be updated in future expansions. In this case, the k-th temporary result's cost is updated beforehand, although the expansions are not done yet. We assign the value of UB_c to the k-th temporary result's accessibility cost, because the cost of the final k-th result is *at most* as large as UB_c.

5 Experiments

In this section, we report our experimental results and analysis.

5.1 Experiment Setup

The following algorithms are compared in the experiment.

ANN is the all nearest neighbor algorithm described in Section 2.
Sep-Tree is separate-tree top-k search algorithm proposed in Section 3.
One-Tree is one-tree top-k search algorithm proposed in Section 3.

All optimizations described in Section 4 are applied to separate-tree and one-tree unless specified otherwise.

All algorithms are implemented as in-memory algorithms. They are implemented in C++ and performed on a PC with Pentium D 3.00GHz CPU and 2GB RAM. The operating system is Debian 4.1. The algorithms are complied using GCC 4.1.2 with `-O3` flag. We used two publicly available real datasets in

Table 1. Statistics of Datasets

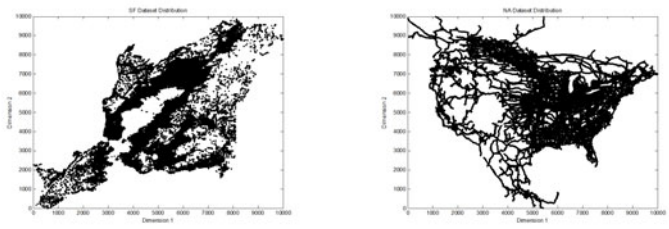
Dataset	$\vert S\vert$	$\vert R\vert$
NA	174,956	17,000
SF	175,813	17,000

the experiment, San Francisco Road Network (SF) and Road Network of North America (NA)[3]. Some important statistics and data distribution are shown in Table 1 and Figure 2. We split each dataset into $|T|$ types, and the default $|T|$ is set to 20 unless specified otherwise. For each data point in the dataset, a random type is assigned.

To generate the set of possible locations R, we choose 10% of the points from the original dataset, and shift the x and y coordinates by a random number in the range of $[-5, 5]$.

We measure the number of internal nodes expanded and the number of leaf nodes expanded in the R-trees, as well as the processing time. The processing time measured does not include the time for constructing R-tree indexes.

[3] `http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm`



(a) San Francisco Road Network (SF)

(b) Road Network of North America (NA)

Fig. 2. Data Distribution

5.2 Effect of Optimization

We first study the effect of optimizations, and run the two proposed algorithms on both NA and SF datasets with several optimization techniques applied. We compare four optimizations:

No-Opt. The basic separate-tree (one-tree) algorithm with no optimization techniques applied.

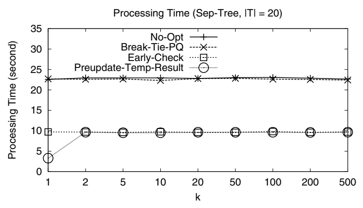
Break-Tie-PQ. The above algorithm equipped with the "break ties in priority queues" optimization. In LPQs, we arrange entries by increasing order of `mind`, and then `maxd`. In GPQs, we arrange entries by increasing order of LB_c, and break ties by UB_c.

Early-Check. The above algorithm equipped with the "early check" optimization technique to avoid redundant node expansions. Before expanding a node v in I_S, we check the `mind` between the LPQ owner u' and v, and prune v if the value of `mind` is no smaller than the `minmaxd` of u''s LPQ.

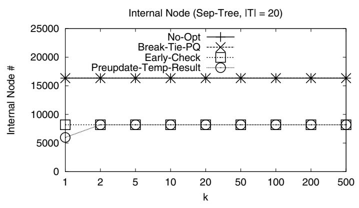
Preupdate-Temp-Result. The above algorithm equipped with the "preupdate temporary result" technique. If a node u in I_R contains at least k points in R, and its LPQ (group)'s UB_c is smaller than the current k-th temporary result's accessibility cost, we then regard UB_c as the k-th temporary result's accessibility cost so as to improve the algorithm's pruning power.

Figures 3(a) – 3(c) show the processing time, the number of internal node expansions, and the number of leaf node expansions using separate-tree on NA dataset. The performance on SF dataset displays similar trends and thus is not shown here in the interest of space. It can be observed that Early-Check is the most effective optimization for separate-tree algorithm. It reduces the processing time by 57%. The main reason is that the number of internal nodes is reduced by 50%, and the number of leaf nodes expanded is reduced by 63% after applying Early-Check. The other two optimization techniques exhibit minor improvements to the separate-tree algorithm.

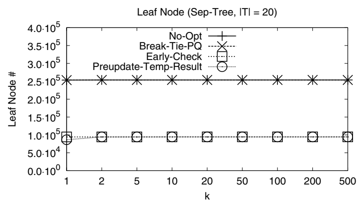
Figures 3(d) – 3(f) show the processing time, number of internal node and leaf node expansions using one-tree algorithm on NA dataset. Break-Tie-PQ is the most significant optimization technique for one-tree algorithm. It reduces the



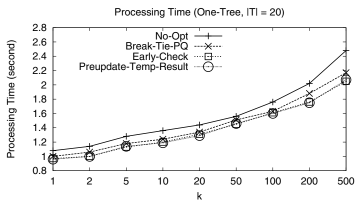
(a) Processing Time



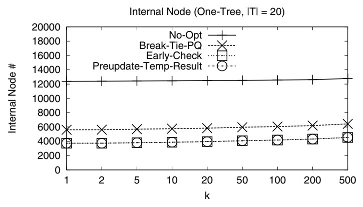
(b) Internal Node Expansion



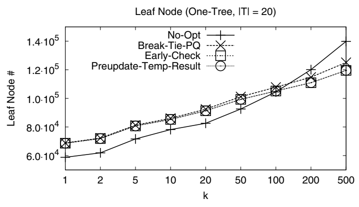
(c) Leaf Node Expansion



(d) Processing Time



(e) Internal Node Expansion



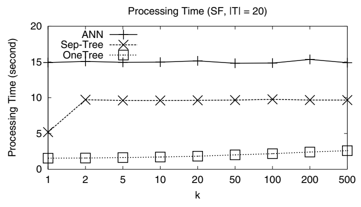
(f) Leaf Node Expansion

Fig. 3. Effect of Optimization (NA)

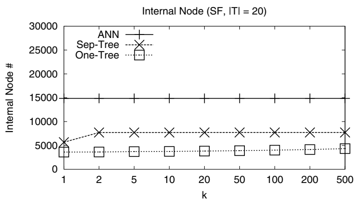
processing time by about 10% when k is small and about 20% when k is large. The other two optimizations can further reduce the processing time. The reason why the effect of Break-Tie-PQ is more remarkable on one-tree algorithm than on separate-tree algorithm is that one-tree algorithm constructs indexes by putting the points with various types in one node, and therefore yields looser estimation of lower bound of accessibility cost. There are more LPQs sharing equal values of LB_c, and therefore it is necessary to break them by introducing the upper bound of accessibility cost.

5.3 Comparison with All Nearest Neighbor Algorithm

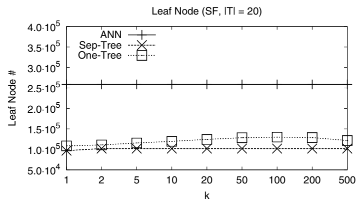
We run the three algorithms with various numbers of returned objects (k) and types of points ($|T|$). Figures 4(a) – 4(c) show the performance on SF dataset with respect to different k. The general trend is that running time of both separate-tree and one-tree algorithm slightly increases when we move k towards larger values; the running time of ANN algorithm is irrelevant to the change of k because it always computes the nearest neighbors for all the possible locations. The result shows that separate-tree algorithm is 1.5 times as fast as the ANN algorithm, and the speed-up of one-tree algorithm can be up to 5.7x. The speed-up is mainly due to more efficient internal node and leaf node expansion. As can be seen, separate-tree reduces the the number of internal nodes by 48%, and one-tree reduces the number by additional 23%. In terms of leaf node expansion, separate-tree and one-tree display similar performance, and the reduction can be up to 61%, compared to the ANN algorithm.



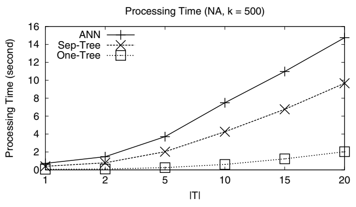
(a) Processing Time (SF)



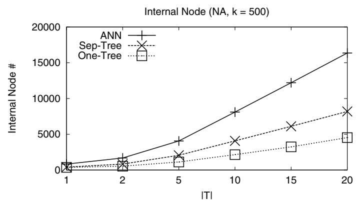
(b) Internal Node Expansion (SF)



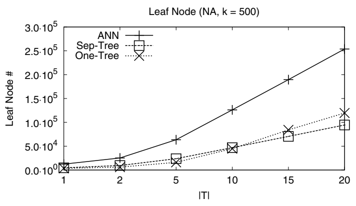
(c) Leaf Node Expansion (SF)



(d) Processing Time (NA)



(e) Internal Node Expansion (NA)



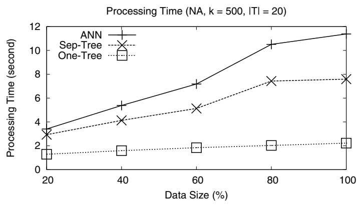
(f) Leaf Node Expansion (NA)

Fig. 4. Comparison with ANN

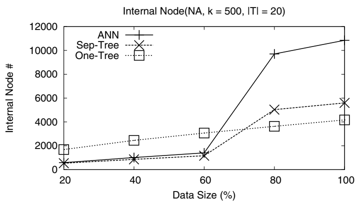
We study the performance of the algorithms with respect to varying number of types $|T|$, and plot the results on NA dataset in Figures 4(d) – 4(f). k is set to 500 in this set of experiments. As shown in the figures, the processing time and the node expansion grow linearly while the number of types $|T|$ is increasing. Both separate-tree and one-tree have slower increasing rates than ANN.

5.4 Scalability against Data Sizes

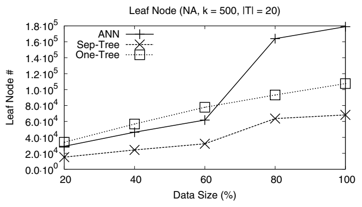
We study how the proposed algorithms perform on different size of datasets. Figure 5 shows the performance on NA dataset with respect to different number of data points. The number of points in R is fixed at 17,000, and k is set to 500



(a) Processing Time



(b) Internal Node Expansion



(c) Leaf Node Expansion

Fig. 5. Scalability (NA)

for this set of experiments. We observe that when data size grows, the processing time of the algorithm grows linearly, When data size is 20% of the original S, one-tree algorithm is about twice as fast as separate-tree algorithm. When data size is 100%, one-tree algorithm is more than three times as fast. Although separate-tree expands three times fewer nodes than one-tree when data size is small, one-tree is still faster under this scenario. This is because one-tree accesses less number of entries in the point level of R-trees, while separate-tree needs to access $|T|$ times for each point to create its LPQ group and compute the accessibility cost.

5.5 Index Sizes

Table 2 shows the size of index on the two datasets using different algorithms, and the size of the original datasets. separate-tree algorithm uses the same amount of disk memory as ANN algorithm, which is 2.6 times as large as the original datasets. one-tree uses about 13% more disk space to store the R-trees because it needs to keep an additional bitmap in each node to indicate what types are assigned to the points indexed by this node and its descendants.

Table 2. Index Size

Dataset	ANN	separate-tree	one-tree	Original Dataset
NA	11.36MB	11.36MB	12.80MB	4.35MB
SF	11.34MB	11.34MB	12.69MB	4.33MB

Summary. Considering the runtime performance and the space usage of the three algorithms, we find that one-tree algorithm achieves the best runtime performance while occupying similar amount of space with the other two. We recommend users to select the highest accessible locations using one-tree algorithm.

6 Related Work

Facility location problem, also known as location analysis, is to find optimal placement of facilities respective to cost to a given set S. The problem we propose in this paper is one of the variations.

Another variation is Minsum facility location [9]. It is to seek a location that minimizes the sum of distances from a set of points to the selected location. [9] proposed three tree-based algorithm to solve it. Among these three proposed algorithms, Virtual OL-tree is the most efficient algorithm. Virtual OL-tree is an extension of k-d-B tree [17]. In [21], an variant of the Minsum facility location problem was studied. This paper proposed a partition-based algorithm. It recursively partitions candidate cell of the query region to smaller cells.

Other related studies include [4, 3, 20]. In [4, 3], the optimization problem is defined as: Given two sets S and P, find the point s in S that satisfies: 1) number of bichromatic reverse nearest neighbours (BRNN) of s in P is maximum; 2) the maximum distance of the BRNN of s is minimum; 3) the minimum distance of the BRNN of s is maximum. [20] solves the problem to find a region Q such that when placing s in Q, its BRNN size is maximum.

The problem we propose is related to k nearest neighbor (NN) query as well. kNN query has been extensively studied by spatial database community and many spatial indexes were proposed to solve this problem [18, 13, 5]. Among these indexes, R-tree [11] and its variations [1] are most popular ones. Two variations of NN query, group nearest neighbor queries and all nearest neighbor, have been recently studied. [6] provided a detailed survey of work related to these two types of queries.

7 Conclusion

In this paper, we study the problem finding k best locations that are close to various types of facilities. We focus on Euclidean space and measure the accessibility using the sum of distances to nearest neighbors. Two algorithms are proposed to efficiently find the top-k answers, with several non-trivial optimizations applied to reduce the number of node expansion and improve runtime performance. The separate-tree algorithm creates indexes for different types of points in separate R-trees, while the one-tree algorithm indexes all the points in a single R-tree. The experiment results show that both proposed algorithms outperform the baseline algorithm with a speed-up up to 5.7 times.

Acknowledgement. The authors would like to thank the anonymous reviewers for their insightful comments. Wei Wang is supported by ARC Discovery Projects DP0987273 and DP0881779.

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 Conference, pp. 322–331 (1990)

2. Böhm, C., Krebs, F.: The -nearest neighbour join: Turbo charging the kdd process. Knowl. Inf. Syst. 6(6), 728–749 (2004)
3. Cabello, S., Díaz-Báñez, J.M., Langerman, S., Seara, C., Ventura, I.: Reverse facility location problems. In: CCCG, pp. 68–71 (2005)
4. Cabello, S., Díaz-Báñez, J.M., Langerman, S., Seara, C., Ventura, I.: Facility location problems in the plane based on reverse nearest neighbor queries. European Journal of Operational Research 202(1), 99–106 (2010)
5. Chaudhuri, S., Gravano, L.: Evaluating top- selection queries. In: VLDB, pp. 397–410 (1999)
6. Cheema, M.: Circulartrip and arctrip: Effective grid access methods for continuous spatial queries
7. Chen, Y., Patel, J.M.: Efficient evaluation of all-nearest-neighbor queries. In: ICDE, pp. 1056–1065 (2007)
8. 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)
9. Du, Y., Zhang, D., Xia, T.: The optimal-location query. In: Anshelevich, E., Egenhofer, M.J., Hwang, J. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 163–180. Springer, Heidelberg (2005)
10. Emrich, T., Graf, F., Kriegel, H.-P., Schubert, M., Thoma, M.: Optimizing all-nearest-neighbor queries with trigonometric pruning. In: Gertz, M., Ludäscher, B. (eds.) SSDBM 2010. LNCS, vol. 6187, pp. 501–518. Springer, Heidelberg (2010)
11. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD Conference, pp. 47–57 (1984)
12. Hjaltason, G.R., Samet, H.: Incremental distance join algorithms for spatial databases. In: SIGMOD Conference, pp. 237–248 (1998)
13. Hjaltason, G.R., Samet, H.: Distance browsing in spatial databases. ACM Trans. Database Syst. 24(2), 265–318 (1999)
14. Li, H., Lu, H., Huang, B., Huang, Z.: Two ellipse-based pruning methods for group nearest neighbor queries. In: GIS, pp. 192–199 (2005)
15. Mouratidis, K., Hadjieleftheriou, M., Papadias, D.: Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In: SIGMOD Conference, pp. 634–645 (2005)
16. Papadias, D., Shen, Q., Tao, Y., Mouratidis, K.: Group nearest neighbor queries. In: ICDE, pp. 301–312 (2004)
17. Robinson, J.T.: The k-d-b-tree: A search structure for large multidimensional dynamic indexes. In: SIGMOD Conference, pp. 10–18 (1981)
18. Seidl, T., Kriegel, H.-P.: Optimal multi-step k-nearest neighbor search. In: SIGMOD Conference, pp. 154–165 (1998)
19. Shin, H., Moon, B., Lee, S.: Adaptive multi-stage distance join processing. In: SIGMOD Conference, pp. 343–354 (2000)
20. Wong, R.C.-W., Özsu, M.T., Yu, P.S., Fu, A.W.-C., Liu, L.: Efficient method for maximizing bichromatic reverse nearest neighbor. PVLDB 2(1), 1126–1137 (2009)
21. Zhang, D., Du, Y., Xia, T., Tao, Y.: Progressive computation of the min-dist optimal-location query. In: VLDB, pp. 643–654 (2006)
22. Zhang, J., Mamoulis, N., Papadias, D., Tao, Y.: All-nearest-neighbors queries in spatial databases. In: SSDBM, pp. 297–306 (2004)

Audio Lifelog Search System Using a Topic Model for Reducing Recognition Errors

Taro Tezuka and Akira Maeda

College of Information Science and Engineering
Ritsumeikan University
{tezuka,amaeda}@media.ritsumei.ac.jp

Abstract. A system that records daily conversations is one of the most useful types of lifelogs. It is, however, not widely used due to the low precision of speech recognizers when applied to conversations. To solve this problem, we propose a method that uses a topic model to reduce incorrectly recognized words. Specifically, we measure relevancy between a term and the other words in the conversation and remove those that come below the threshold. An audio lifelog search system was implemented using the method. Experiments showed that our method is effective in compensating recognition errors of speech recognizers. We observed increase in both precision and recall. The results indicate that our method has an ability to reduce errors in the index of a lifelog search system.

Keywords: Lifelog, topic model, information retrieval, audio data, hierarchical Bayes.

1 Introduction

A lifelog is a system that digitally stores the user's daily activities through various recording devices. Among various types of lifelogs, the audio lifelog, which records daily conversations, is probably the one with the clearest merit. Looking through past discussions with colleagues often provides beneficial business ideas. The lifelog also makes it possible to find a piece of information that was once heard in a casual conversation.

Despite its high potential, such a system is not widely used outside of research laboratories. One reason is that the speech recognition for daily conversations is not precise enough. Until recently, speech recognition systems were targeted at recognizing formal speech, such as lectures and presentations. A conversation is a more difficult target, since the speed of speech varies, and words are often less clearly spoken. Transcriptions by speech recognizers used with conversations, therefore, contain too many errors, making them unsuitable for retrieval or text mining.

Our approach is to use contextual information about the conversation to correct words that are possibly recognition errors. The approach is based on the observation that speech recognizer errors are often phonetically similar to the correct word but semantically far from other words appearing in the conversation. We developed a lifelog search system that uses a topic model to rank terms according to its relevance to the whole context. In other words, we rank terms using semantic context of the conversation. The goal is to filter out recognition errors and to increase the search precision.

In this paper, we mainly discuss a method of reducing recognition errors that frequently occur when applying speech recognition to daily conversations. The rest of the paper consists of the following sections. Section 2 gives related work. Section 3 describes our method in detail. Section 4 illustrates implementation, and Section 5 describes the evaluation results. Section 6 is the conclusion.

2 Related Work

We discuss related work on lifelog search, speech recognition for conversations, and the use of global semantic context for speech recognition.

2.1 Lifelog Search

Lifelogs can include images, videos, sound, locations, documents, physiological data, and all other types of sensory data. There are now a vast number of studies on lifelogs, although only a few of them are actually being used outside research laboratories. Sellen and Whittaker give an overview of possible future directions of lifelogs [1]. They point out that the system should aim to meet the user's specific goals in using the lifelogs, rather than just trying to store everything.

2.2 Speech Recognition for Conversation

Using speech recognition with conversation is difficult, and there have been various approaches. It is a typical case of a large-vocabulary continuous speech recognition (LVCSR) problem [2,3]. When the vocabulary set is large, estimation of the parameters of a language model becomes high dimensional, and it becomes increasingly difficult to obtain an accurate model. Because the size of the training corpus is limited, it is often difficult to obtain good parameter estimation.

In addition to the difficulties of LVCSR, conversations present more difficulties due to disfluencies, hesitations, or false starts [4]. Speaking rate (speaking speed) is likely to vary, and the speech of other speakers may come in as a noise. It is easier if the domain of conversation is limited, but this is not feasible for lifelogs, since it can cover a wide range of domains.

2.3 Global Semantic Context for Speech Recognition

In our proposed method, we use global semantic context in a conversation to reduce recognition errors. Unlike widely used n-gram-based language models that consider only local context, we consider the global topic of the conversation.

Bellegarda did extensive work on applying latent semantic analysis (LSA) to improve speech recognition [5,6]. LSA has been criticized for assuming continuous variables for term appearance, since the appearance of a term in a document is a discrete feature. Instead, we use a model that assumes discrete variables for terms and documents. Also, his aim was different from ours in that his goal was to incorporate the

LSA model into the language recognition process, whereas our goal is to improve precision of terms in the inverted index.

Wick et al. used a topic model to improve precision of OCR (optical character recognition)[7]. In estimating the topic of a given document, their method uses words that were judged to be highly reliable by the OCR system. Words that were judged to be less reliable are corrected based on the topic indicated by the highly reliable words. On the other hand, our method does not require any external information source to find out which words are reliable and which are not. In this sense our problem setting is harder than the one handled by Wick et al.

3 Method

The output of a speech recognizer is called a *transcription*. A lifelog search system must search through a database of transcriptions and find conversations that are relevant to the query given by the user. Unfortunately, the state-of-the-art speech recognition engines contain many recognition errors in its transcriptions, resulting in too many irrelevant terms added to the index. On the other hand, since the task is to search, fully transcribed results are not necessary. A set of terms relevant to each part of the conversation is what we need. We therefore developed a method that uses a topic model to rank terms and to filter out speech recognition errors. The basic idea is that a term can be filtered out from the speech recognition results, if it is semantically distant from other terms in the same conversation.

A topic model is constructed using statistical distributions of terms in a training corpus. Specifically, we use a topic model to measure relevancy between a term and the transcription it is contained in.

3.1 Topic Model

There is a wide range of choice regarding topic models, but we used Latent Dirichlet Allocation (LDA), which is considered one of the most effective text modeling methods [8]. The parameters of LDA can be learned using a Gibbs sampler [9]. Figure 1 is a graphical model of LDA. It is a generative model of words in documents. It considers the probability distribution of a topic mixture for each document. Once a topic mixture is generated, words are generated from the mixture using a conditional distribution. Through topics, LDA expresses a relationship between words based on their occurrences in a large set of text.

In the following formulation, “term” refers to an element of a vocabulary, and “word” refers to an instantiation of a term in a specific position in a document. Therefore, a term t can appear more than once in a transcription, meaning $w_i = w_j = t$. In Figure 1, M is the number of documents, N_m is the number of words in a document m, K is the number of latent topics, and $w_{m,n}$ is a discrete random variable representing a word at a position n of a document m. The value of $w_{m,n}$ is represented by t, which is an element of vocabulary V. The z is a discrete variable representing a topic in a position n of a document m. It is a latent variable and cannot be observed, but probabilistically determines the value of a word $w_{m,n}$. θ_m is a K dimensional parameter vector of a multinomial distribution $P(z_{m,n})$. It represents an underlying topic allocation of a document m. The ϕ_k is a V dimensional parameter vector of a multinomial distribution $P(w_{m,n}|z_{m,n} = k)$, where V is the size of a vocabulary and k is a topic. ϕ_k represents a probability distribution of term appearances when the topic is k. The α and β are hyperparameter vectors.

From the outcome of the training, we use the parameters of multinomial distributions $P(w|z)$ and $P(z)$ to measure relevancy between a term and a transcription. $P(w = t|z = k)$ is equal to ϕ_{kt} where k is a topic and t is a term. Since $P(z = k)$ is equal to $\sum_m P(z = k|d = m)P(d = m)$, assuming the probability of the appearance of a document m as equal, we can use $\sum_m \theta_{mk}$ for $P(z = k)$.

3.2 Speech Recognition

Many speech recognition engines (speech recognizers) can provide multiple candidate transcriptions ranked in decreasing order of likelihood. Our system uses not only the result with the highest likelihood, but also the lower-ranked results. It often happens that the top-ranked transcription does not contain the terms actually spoken by the speaker, but the lower ranked transcriptions do. We therefore store terms that are contained in top-κ transcriptions, where κ is an arbitrary number. Since our goal is a search system, it is not necessary to store results as sentences. We therefore store all terms appearing in top candidates to indices, and then filter out irrelevant ones. The grammatical structure is lost, but it will suffice for the search purpose.

3.3 Term Ranking and Filtering

When a recognition error occurs during transcription, the error is usually phonetically similar but semantically far from the correct term. For example, the term "year" may be recognized as "ear." In such a case, the semantic meanings of two terms are totally different. It is therefore natural to use semantic information to make corrections. People are also likely to use such a mechanism while listening to someone else talking, since whenever terms irrelevant to the context appear, we are not good at understanding them.

By filtering out semantically isolated terms, the search result is expected to give higher precision. Since we are storing top-κ transcriptions provided by the speech recognizer, if the terms in the top-ranked transcription are filtered out, the terms in the lower ranked transcriptions are the only ones to be added to the index. In this sense, our method is correcting the recognition errors by using the semantic context, rather than just filtering out error terms.

3.4 Relevancy Measure between Term and Transcription

To rank terms and filtering out semantically isolated terms, a relevancy measure between a term and a transcription must be defined. We defined it using the parameters of the probability distributions $P(w|z)$ and $P(z)$ obtained in Subsection 3.1. One way to define the relevancy of a term t to a transcription $\tilde{m}$ is to use a conditional probability $P(w = t|d = \tilde{m})$. It can be calculated using $P(w|z)$ and $P(z|d)$.

$$\begin{aligned} r(t,\tilde{m}) &= P(w=t|d=\tilde{m})c(t,\tilde{m}) \\ &= \sum_{k=1}^{K} P(w=t|z=k)P(z=k|d=\tilde{m})c(t,\tilde{m}) \end{aligned} \tag{1}$$

$P(w=t|d=\tilde{m})$ is the probability of term t appearing in transcription $\tilde{m}$. $c(t,\tilde{m})$ is the maximum score attached to term t in the transcription $\tilde{m}$, provided by the speech recognizer. Conditional independence indicated by Figure 1 is used in factorizing $P(w=t|d=\tilde{m})$.

From LDA, we have parameters for $P(w=t|z=k)$. The problem is that since the transcription $\tilde{m}$ is not a part of the original corpus used for training LDA, $P(z=k|d=\tilde{m})$ is not available. We therefore calculate it based on the terms contained in the transcription $\tilde{m}$. Using the Bayes' theorem, we can rewrite $P(z=k|d=\tilde{m})$ in the following way.

$$\begin{aligned} P(z=k|d=\tilde{m}) &= \frac{P(d=\tilde{m}|z=k)P(z=k)}{P(d=\tilde{m})} \\ &\propto P(d=\tilde{m}|z=k)P(z=k) \end{aligned} \tag{2}$$

Assuming conditional independency of words in a transcription, $P(d=\tilde{m}|z=k)$ can be factorized as follows.

$$P(d=\tilde{m}|z=k) = \prod_{i=1}^{n_{\tilde{m}}} P(w_i=t_i|z=k) \tag{3}$$

In Equation 3, w_i is the word at position i, and t_i is the term appearing there in the transcription $\tilde{m}$. The number of terms in $\tilde{m}$ is represented by $n_{\tilde{m}}$. When the number of terms in a dictionary is large, $P(w_i=t_i|z=k)$ is small. Multiplying $n_{\tilde{m}}$ of them results in an extremely small amount. One way to cope with it is to use the logarithm of $\prod_{i=1}^{n_{\tilde{m}}} P(w_i=t_i|z=k)P(z=k)$, but in that case it becomes difficult to compute marginalization by z in Equation 1. To avoid this problem, we used the following value $Q(\tilde{m},k)$ instead of $P(z=k|d=\tilde{m})$. Specifically, we took the arithmetic average of $P(w_i=t_i|z=k)$, the probability that the term t_i appears, and multiplied it by $P(z=k)$.

$$Q(\tilde{m},k) = \frac{1}{n_{\tilde{m}}} \sum_{i=1}^{n_{\tilde{m}}} P(w_i=t_i|z=k)P(z=k) \tag{4}$$

Using this value, we defined a new measure of relevancy, $\hat{r}(t,\tilde{m})$, as follows.

$$\hat{r}(t,\tilde{m}) = \sum_{k=1}^{K} P(w=t|z=k)Q(\tilde{m},k)c(t,\tilde{m}) \tag{5}$$

Terms appearing in a transcription are sorted using the value of $\hat{r}(t, \tilde{m})$. If a term is ranked below the ratio ρ of the whole set, it is filtered out. In other words, a set of terms that are added to the inverted index is expressed by Equation 6. $n_{\tilde{m}}$ is the number of terms in the transcription $\tilde{m}$ and $R(t)$ is the rank of the term t. Inequality $R(t) < R(t')$ means that $\hat{r}(t, \tilde{m}) > \hat{r}(t', \tilde{m})$. Ranking starts from 1, which means $R(t) \geq 1$.

$$\{t_j | R(t_j) < n_{\tilde{m}} \cdot r_0\} \tag{6}$$

If a term that is not contained in the vocabulary appears, it will not be filtered out. The terms that were not filtered out are added to the *inverted index* of the lifelog search system. By changing the threshold value ρ, we can obtain different values for recall and precision. In the evaluation section, we compare it to the result based on the index created using the top-ranked transcription from the speech recognizer.

4 Implementation

Using the proposed method, we implemented LifeRecycle, a lifelog search system developed by ourselves for storing and searching audio lifelogs. It consists of a recording module, indexing module, and search interface. Each component is described in more detail in the following subsections. We used Java for implementation. For database management, we used MySQL. For now, we have implemented the system using Japanese. The method is, however, independent of language, so it can be applied to other languages as well. Audio data is recorded using a digital voice recorder. We use a headset microphone for recording.

4.1 Document Set

Estimating the parameters of a topic model requires a large set of documents. Such a set of documents is called a corpus. Since everyday conversation covers a wide range of topics, it is preferable that the content of the corpus cover as many topics as possible. To meet this need, we chose to use Wikipedia, one of the largest online encyclopedias on the Web[11]. Articles on Wikipedia are submitted and revised by volunteer contributors. There are more than 670,000 articles in the Japanese version of Wikipedia.

From each of the top 10,000 categories (in terms of the number of contained articles) in Wikipedia, we sampled a single article. To set the sizes of documents nearly equal, 1,000 nouns from the beginning of each article were extracted to form a document. This is to avoid too much influence from long articles. By using nouns only, the grammatical structures in the original articles are lost, but since our model uses the bag-of-word assumption, it is permissible. We call this set of documents a *corpus*. There are of course some differences between the content of Wikipedia articles and the content of daily conversations. We assume that we can cope with these differences, since by the nature of LDA, we do not necessary use all topics appearing in Wikipedia, but only the ones that are relevant to the words appearing in a conversation.

Using the set of documents obtained as described, we extracted a set of terms fulfilling the conditions described below. We call it a *vocabulary*. Terms that appeared in more than half of the documents in the corpus were excluded from the vocabulary.

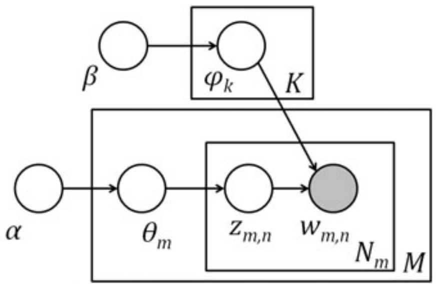


Fig. 1. Graphical model for LDA

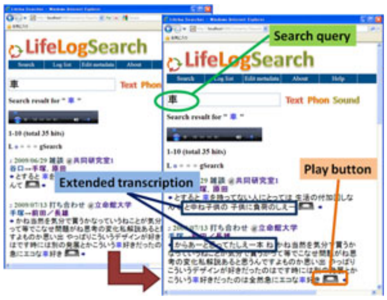


Fig. 2. Search functions in LifeRecycle

These are common terms and do not characterize a document. The terms that appear in more than δ documents were added to the vocabulary set. The δ is an arbitrary threshold. If all the terms were added to the vocabulary, the dimensions of the word vector would be too large, and the estimation process would be computationally too intensive.

From this corpus, we used terms that appear in more than two documents but less than 50% of the documents. This resulted in a vocabulary of 22,131 words. We have set K, the number of latent topics, to 50. For the hyperparameters of the prior density, we used symmetric values: $\alpha = 1$, $\beta = 0.1$. Following the recommended values of hyperparameters in Griffiths and Steyvers, we used $\alpha_k = 50/K$ and $\beta_t = 0.1$, which is in this case $\alpha_k = 1$ and $\beta_t = 0.1$[9].

4.2 Speech Recognition and Indexing

For speech recognition, we used Julius[12]. It is an open source speech recognition system consisting of a preprocessor for sound file and a speech recognizer. For a phonetic and language model, we used a model built using the CSJ corpus[13]. Since these models are built mainly on lectures and presentations, the speed of speech is different from that of conversation. Therefore, we used an adjustment method using multiple frame shifts and window sizes, described in Subsection 3.2. For the frame shifts and window sizes, we used the following pairs: 160 - 400, 152 - 380, 144 - 360.

We used $\kappa = 100$, which means terms contained in the top 100 candidates were stored in the transcription hash and used in the filtering phase. Using the method described in Subsection 3.4, the output of Julius was sent to the filter. Terms that were not filtered are added to the inverted index.

4.3 Search Interface

We have implemented a lifelog search system in which the user can find the content of a past conversation by sending a query consisting of terms that were spoken during the conversation. Figure 2 shows our search interface. We implemented it so that the user can access our search system from a regular web browser, since this is one of the most familiar search interfaces. It has an input box to type in a query at the top. A set of keywords divided by spaces is interpreted as an AND query, just like in an ordinary web search engine. Search results are ranked in order of relevance. In each transcription, the matched transcription and its surrounding transcriptions are presented to the user.

5 Evaluation

We have performed experiments to test the effectiveness of our proposed method. We evaluated precision and recall of the terms added to the inverted index by altering the threshold value ρ. For a comparison, we measured precision and recall for LSA (Latent Semantic Analysis), which is a widely used topic model in existing papers.

5.1 Precision and Recall

Experiments to check precision and recall of our proposed method were performed using texts from the CSJ (Corpus of Spontaneous Japanese)[13]. This corpus consists of audio data from spontaneous speech. It has a part called "core," which contains manually transcribed texts to speech data. The core of the CSJ contains 45 hours of conversations, about 500,000 words. The reason for using the CSJ rather than conversations obtained by our recording device is because manually transcribing conversations is a tremendously laborsome task. Since the CSJ contains the transcribed texts, it is ready for evaluation.

Speech data in the CSJ comes from the following sources. We measured precision and recall for each of them. "A" is for an academic presentation speech. "D01" is for an interview after a simulated public speaking. "D02" is for a task oriented dialogue, in which is a conversation between two speakers performing a specific task designed by the editors of the CSJ. "D04" is for an interview after an academic presentation speech.

Although the phonetic and language model we used in the speech recognizer were trained using the CSJ, our proposed method is independent from these models, therefore it is not a problem to use the part of the corpus as test data. We tested our method using talks and conversations in the core of the CSJ, since it contains manually transcribed texts. Terms appearing in these texts are considered as the correct terms.

Precision and recall are defined as follows. Let N indicate a set of terms obtained using our proposed method and M indicate a set of terms contained in the manually transcribed text. In other words, M is the set of correct terms. Precision is defined as $\frac{|N \cap M|}{|N|}$. Recall is defined as $\frac{|N \cap M|}{|M|}$.

Figures 3-6 illustrate precision-recall curves for each source, using different values of threshold ρ. The precision and recall are average values obtained using all files in each source. We used the following values for ρ: 0, 0.005, 0.01, 0.015, 0.02, 0.025, 0.03, 0.05, 0.075, 0.1, 0.15, 0.2, 0.3, 0.4, and 0.5. A rectangular marker ("semantic ranking") indicates the result using our proposed method that ranks and filters terms obtained from top-κ transcriptions. A triangular marker ("ASR score") indicates to the result using the scores provided by the automatic speech recognizer (ASR) for ranking and filtering the same set of terms. An X marker ("LSA") indicates the result based on a topic model obtained by Latent Semantic Analysis.

The result indicates that our system is effective in increasing the average precision and recall of the inverted index. The content of source D01, D02, and D04 are more

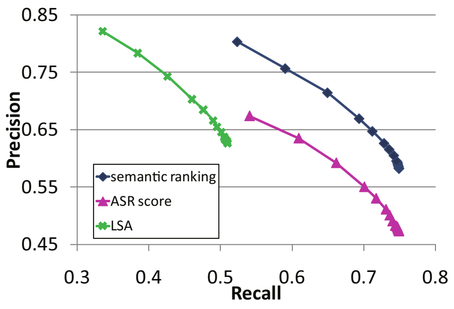


Fig. 3. P-R curve for source A (academic presentation speech)

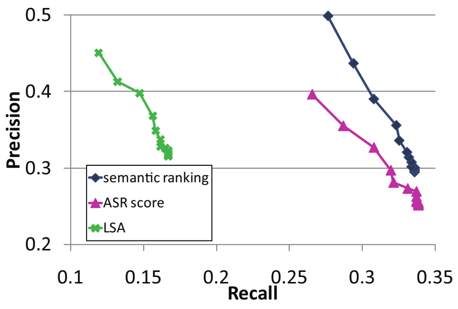


Fig. 4. P-R curve for source D01 (interview after simulated public speaking)

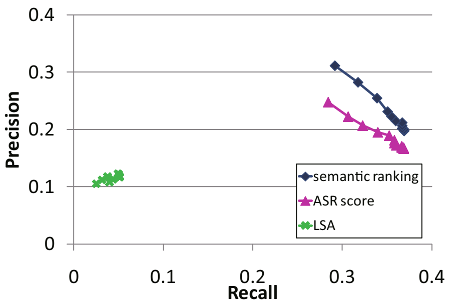


Fig. 5. P-R curve for source D02 (task oriented dialogue)

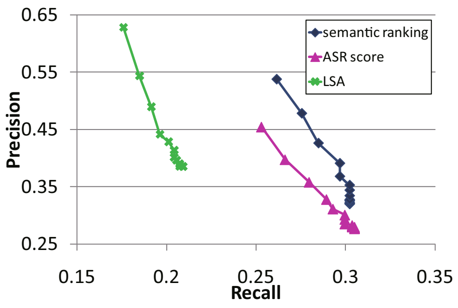


Fig. 6. P-R curve for source D04 (interview after academic presentation speech)

casual than that of A. The result shows that our method is effective for both formal and casual talks.

Figures 3-6 also compare our method with the result using only the scores $c(t, \tilde{m})$ provided by the speech recognizer. Terms are ranked according to the score and those that came below the threshold rank are removed. The corresponding P-R curve is worse than that of our method. Our method is much better than using LSA also. We can assume that LSA did not represent the topic structures of conversations well enough. One possible reason is that while LSA models the frequency of a term as a continuous probabilistic variable, in reality it can only take discrete values, therefore resulting in an inadequacy of the model.

6 Conclusion

Low precision and recall in an automatic speech recognition engines is a problem that prohibit its wider use in storing and searching past conversations. We proposed a method of ranking terms in a transcription by their semantic appropriateness and filtering out those that are likely to be errors. We applied a topic model constructed from

Wikipedia articles to cope with a wide range of topics covered in everyday conversations. The result showed improvement of precision and recall for various types of speeches. Using our method, we have implemented an actual audio lifelog search system. We plan to extend our system to handle more complex queries, including search facility using a phrase.

Acknowledgments

This work was supported in part by a MEXT Grant-in-Aid for Strategic Formation of Research Infrastructure for Private University "Sharing of Research Resources by Digitization and Utilization of Art and Cultural Materials" (Grant Number: S0991041) and a MEXT Grant-in-Aid for Young Scientists (B) "Object Identification System using Web Image Collection and Machine Learning" (Leader: Taro Tezuka, Grant Number: 21700121), and IPA Exploratory IT Human Resources Project (The MITOH Program) "LifeRecycle - A search engine for life". (Leader: Taro Tezuka, 2008 second semester).

References

1. Sellen, A., Whittaker, S.: Beyond total capture: a constructive critique of lifelogging. Communications of the ACM 53(5), 70–77 (2010)
2. Rabiner, L., Juang, B.H.: Fundamentals of speech recognition. Prentice Hall, Englewood Cliffs (1993)
3. Ney, H., Ortmanns, S.: Dynamic Programming Search for Continuous Speech Recognition Contents. IEEE Signal Processing Magazine 16, 64–83 (1999)
4. Holmes, J., Holmes, W.: Speech synthesis and recognition. Taylor & Francis, Abington (2001)
5. Bellegarda, J.R.: Exploiting latent semantic information in statistical language modeling. Proc. of the IEEE 88(8), 1279–1296 (2000)
6. Bellegarda, J.R.: Statistical language model adaptation: review and perspectives. Speech Communication 42, 93–108 (2004)
7. Wick, M.L., Ross, M.G., Learned-Miller, E.G.: Context-Sensitive Error Correction: Using Topic Models to Improve OCR. In: Proc. of the 9th International Conference on Document Extraction and Analysis, pp. 1168–1172 (September 2007)
8. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
9. Griffiths, T.L., Steyvers, M.: Finding Scientific Topics. Proc. of the National Academy of Sciences of the United States of America 101, 5228–5235 (2004)
10. Heinrich, G.: Parameter estimation for text analysis, Technical Note, ver 2.4 (2008), http://www.arbylon.net/publications/text-est.pdf
11. Wikipedia, http://wikipedia.org
12. Julius - Open-Source Large Vocabulary CSR Engine, http://julius.sourceforge.jp/en_index.php
13. The Corpus of Spontaneous Japanese (CSJ Corpus), http://www.kokken.go.jp/katsudo/seika/corpus/public/

Towards Web Search by Sentence Queries: Asking the Web for Query Substitutions

Yusuke Yamamoto[1,2] and Katsumi Tanaka[1]

[1] Graduate School of Informatics, Kyoto University, Japan
{yamamoto,tanaka}@dl.kuis.kyoto-u.ac.jp
[2] JSPS Research Fellow

Abstract. In this paper, we propose a method to search the Web for sentence substitutions for a given sentence query. Our method uses only lexico-syntactic patterns dynamically generated from the input sentence query to collect sentence substitutions from the Web on demand. Experimental results show that our method works well and can be used to obtain sentence substitutions for rare sentence queries as well as for popular sentence queries. It is also shown that our method can collect various types of sentence substitutions such as paraphrases, generalized sentences, detailed sentences, and comparative sentences. Our method searches for sentence substitutions whose expressions appear most frequently on the Web. Therefore, even if users issue the sentence query by which Web search engines return no or few search results for some reasons, our method enables users to collect more Web pages about the given sentence query or the sentences related to the query.

1 Introduction

Web search engines like Google and Bing are great tools for searching the Web. People can efficiently obtain what they want by conveying their information needs as queries to Web search engines. There are three possible ways to generate queries for Web search engines: keyword query, phrase query, and sentence query. Using keyword queries or phrase queries, people can obtain many Web pages containing the queries, but sometimes many irrelevant Web pages are also collected. In contrast, when using sentence queries, people can convey their information needs in more detail, expecting to obtain very relevant Web pages.

Although sentence queries are very useful for clearly representing information needs, people rarely use them because the sentence queries often cause users failure to get Web pages about the queries. There are two possible cases not to obtain Web pages well using sentence queries. The first case is that the meaning of sentence queries is correct but the expressions of the queries are rare on the Web. For example, when users want to search for Web pages describing that Germany is famous for beer and they issue sentence query “beer is famous in Germany”, if the expression of the sentence query appears in few Web pages, Web search engines return few results. The second case is that what sentence queries means is wrong or rare on the Web. For example, if users misunderstand that the capital of *Austria* is Canberra and they issue sentence query "the capital of *Austria* is Canberra", Web search engines do not return any results although they can returns Web pages describing "the capital of *Australia* is Canberra". The both cases result from the cause that if expressions of sentence queries rarely appear on the Web, search engines cannot return any relevant Web pages.

As for keyword query search, most Web search engines provide query substitution functions to deal with the cases that users' queries have miss spelling, that the queries' expression or meaning is rare or abstract, or that the queries are partially wrong because of users' misunderstandings [1,4]. Users can obtain more Web pages which they want to browse using suggested keyword queries. However, unfortunately, there is no Web search engines provide query substitution functions for sentence queries as far as we know. For more flexible Web search with natural language, query substitution for sentence query is important.

In this paper, we propose a method to search for sentence substitutions for sentence queries, for obtaining more Web search results for the initial sentence queries and sentences related to them. Our main idea is to search the Web for sentence substitutions of a sentence query by considering popular expressions and popular topics on the Web. In our method, given a sentence query, we first collect paraphrases for the sentence query from the Web by issuing keywords consisting of the sentence query to Web search engines. After that, we extract sentence substitutions from Web search engines' indices by applying lexico-syntactic pattern mining with the sentence query and its paraphrases. Our proposed method does not require huge corpora in advance because necessary and fresh corpora are collected by using Web search engines on demand. The method also does not need language dictionaries or tools like POS taggers or parsers. Therefore, our method can search for sentence substitutions for any type of sentence query.

2 Related Work

One possible output of our method is paraphrase. Many studies have been done on paraphrasing in the field of natural languages processing. Qiu et al. presented a framework to recognize paraphrases from text corpora, focusing on the dissimilarity between sentences [6]. Kaji et al. proposed a method to paraphrase from expressions for written language to ones for spoken language based on occurrence in written and spoken language corpora [5]. As in these studies, most approaches are based on off-line processing through machine learning or deep natural-language processing, and they require huge corpora for paraphrasing in advance. Moreover, most are focused on only paraphrases based on types of phrase substitution. In contrast, our method collects not only paraphrases but also related sentences (generalized sentences, specified sentences, comparative sentences, and so on) as sentence substitutions for a given sentence query, and these are collected from Web search engine indices on demand.

In the field of information extraction, lexico-syntactic patterns are often used for entity extraction [3,7]. KnowItAll is a system for searching the Web

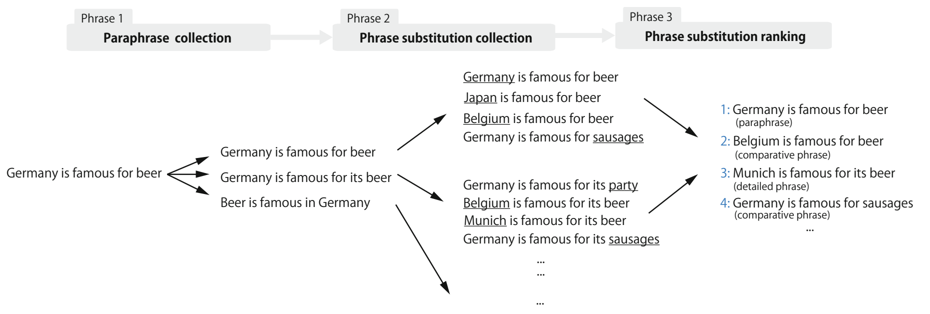

Phrase 1
Paraphrase collection
Phrase 2
Phrase substitution collection
Phrase 3
Phrase substitution ranking
Germany is famous for beer
Germany is famous for beer
Germany is famous for its beer
Beer is famous in Germany
Germany is famous for beer
Japan is famous for beer
Belgium is famous for beer
Germany is famous for sausages
Germany is famous for its party
Belgium is famous for its beer
Munich is famous for its beer
Germany is famous for its sausages
1: Germany is famous for beer (paraphrase)
2: Belgium is famous for beer (comparative phrase)
3: Munich is famous for its beer (detailed phrase)
4: Germany is famous for sausages (comparative phrase)

Fig. 1. Workflow of our method for the sentence query, *"Germany is famous for beer"*

for entity names in the same class as a given example using lexico-syntactic patterns like *"such as"* and *"and other"* [2]. KnowItAll learns effective syntactic patterns for entity extraction in advance from many relevant and irrelevant terms for expected entity names. In our previous work, we used lexico-syntactic pattern mining techniques to develop HontoSearch, a system which collects comparative sentences for a given sentence that helps users check the credibility of a given sentence [8]. The goal of this study is to comparative sentences for credibility judgment on a given sentence. On the other hand, our sentence substitution method provides paraphrases, generalized sentences, and specialized sentences for a given sentence as well as comparative sentences, for the purpose of assisting users to efficiently obtain Web pages by sentence queries..

3 Method

Given sentence query q, we wish to search the Web for sentence substitution s of q. We define this as $q \mapsto s$. The goal of our work is to collect sentence substitutions $S = \{s|q \mapsto s\}$ from the Web and rank them using ranking function $\text{rank}(s|q)$. The workflow of our approach is shown below (Fig.1 illustrates the workflow when sentence *Germany is famous for beer* is given): We first collect paraphrases $P = \{p_1, p_2, .., p_m\}$ of q from the Web using core terms of q (Phase 1). The core terms are the ones which consist of the sentence query and are not stopwords. After that, we obtain Web search results by using P and collect q's sentence substitutions S from the search results (Phase 2). To collect S from the Web search results without using specific language parsers such as POS taggers, we use a combination of multiple lexico-syntactic patterns which we can generate with S. After collecting S, we rank each sentence substitution $s \in S$ for q through $\text{rank}(s|q)$, which evaluates s's frequency of appearance on the Web and its relevance for q (Phase 3).

3.1 Searching for Paraphrases

Given sentence query q, we first collect sentences that contain all core terms in sentence query q and that have a low edit distance between them and q. We regard such sentences as paraphrases of q. For example, given sentence query $q =$"*Germany is famous for beer*", "*Germany is famous for its beer*" contains all of the core terms {*Germany, beer, famous*} of q, and the edit distance between it and q is low. Therefore we regard sentence "*Germany is famous for it beer*" as one of possible paraphrases of q.

We search the Web for paraphrases P of sentence query q as follows:

Phase 1. Searching the Web for paraphrases

1. The given q is divided into terms. Stop words are then omitted from a list of the terms. The remaining terms are denoted as $T_q = \{t_1, t_2, ..., t_n\}$ and we call T_q the *core terms*.
2. Text contents (snippets) that contain all terms in T_q are gathered by issuing query "$t_1 \wedge t_2 \wedge ... \wedge t_n$" to a conventional Web search engine. They are denoted as $\mathrm{Doc}(T_q)$.
3. The system extracts the strings that contain all terms in T_q and are minimal-length from each split sentence in $\mathrm{Doc}(T_q)$.
4. For each p_c of the extracted strings, the term-based edit distance $\mathrm{dist}_{edit}(q, p_c)$ is calculated. If $\mathrm{dist}_{edit}(q, p_c)$ is lower than threshold θ_{edit}, p_c is added to a set of paraphrases P.

3.2 Searching for Sentence Substitutions

Given a sentence query, we suppose that its sentence substitutions have the following features: *Lexical-syntactic patterns of the sentence substitutions are similar to that of the sentence query or those of its paraphrases.* For example, given sentence query $q =$"*Germany is famous for beer*", sentence "*Munich is famous for beer*" has the same lexico-syntactic pattern as that of q (*X is famous for beer*). Also, sentence "*Munich is famous for its beer*" has the same syntactic pattern as that of q's paraphrase "*Germany is famous for its beer*" (*X is famous for its beer*). This indicates that we can collect sentence substitutions using lexico-syntactic patterns of the sentence query and its paraphrases.

In the next phase, we search the Web for sentence substitutions for a given sentence based on the above hypothesis. Here, given sentence q, we denote the lexico-syntactic pattern to focus on term t in q as $\mathrm{pt}(q, t)$. For example, given sentence $q =$ *"Germany is famous for beer"*, $\mathrm{pt}(q,$ "*Germany*$'')$ represents *"(*) is famous for beer"*. We use only lexico-syntactic patterns of sentence query q and its paraphrases P to search for sentence substitutions. The following is a workflow of the method.

Phase 2. Searching for Sentence substitutions

1. Given q and core term $t \in T_q$, the system generates lexico-syntactic patterns of each of collected paraphrases P, $\mathrm{Pt}(P, t) = \{\mathrm{pt}(p, t) | p \in P\}$, by replacing term t in each paraphrase with asterisk.

2. The system issues each $\mathrm{pt}(p,t) \in \mathrm{Pt}(P,t)$ as a sentence query to a conventional Web search engine, and then the system gathers Web search results $\mathrm{Doc}(\mathrm{pt}(p,t))$ for each pattern.
3. For each $\mathrm{pt}(p,t) \in \mathrm{Pt}(P,t)$, the substrings that match the asterisk of $\mathrm{pt}(p,t)$ are extracted from snippets of $\mathrm{Doc}(\mathrm{pt}(p,t))$. We denote extracted substrings as $E = \{e_1, e_2, .., e_m\}$. Moreover, for each $e \in E$, the number of e extracted by only $\mathrm{pt}(p,t)$ is temporarily retained as $\mathrm{num}(e, \mathrm{pt}(p,t))$.
4. For $e \in E$, the replaceability of e for t is scored by considering $\mathrm{num}(e, \mathrm{pt}(p,t))$ for $p \in P$ and the characteristics of the paraphrases used to extract e.
5. If the replaceability of e for t is higher than threshold θ_{rep}, we replace t with e in the paraphrases with which e is extracted. After that, the replaced sentences are added to a list of sentence substitutions S.
6. For all $t \in T_q$, the operations from Step 1 to 5 are executed.

In Steps 1 and 2, we focus on a certain core term $t \in T_q$ and prepare a corpus for collecting sentence substitutions for $p \in P$ from the Web. For example, given sentence $q =$"*Germany is famous for beer*", we get $P = \{$*Germany is famous for beer, Germany is famous for its beer, Germany famous for beer*$\}$ in Phase 1. We now wish to obtain sentence substitutions of q focusing on the core term $t =$ *"Germany"*. We issue sentence queries, *"* is famous for beer"*, *"* is famous for its beer"*, and *"* famous for beer"* to a conventional Web search engine.

In Step 3, we extract all substrings that match asterisk of $\mathrm{pt}(p,t)$ in Doc $(\mathrm{pt}(p,t))$. For example, when we have a snippet *"Prague, the Czech Republic is famous for its beer, and they have the ..."* for $\mathrm{pt}(p, \text{"}Germany'')$ = *"* is famous for its beer"*, we can extract substrings, *Republic*, *Czech Republic*, *the Czech Republic*, *, the Czech Republic*, and *Prague, the Czech Republic* from the snippet. These are then added to a list of substrings E.

In Step 4, we score the replaceability of extracted substring $e \in E$ for core term t using the following functions:

$$\mathrm{rep}(e|t) = \sum_{p_i,p_j} \min(\mathrm{num}(e,\mathrm{pt}(p_i,t)), \mathrm{num}(e,\mathrm{pt}(p_j,t))\cdot \mathrm{sim}_{edit}(q,p_i)\cdot \mathrm{sim}_{edit}(q,p_j) \tag{1}$$

where function $\mathrm{sim}_{edit}(q,p) = 1 - \mathrm{dist}_{edit}(q,p)$.

Formula 1 means that we estimate the replaceability of the extracted substrings for a core term in a give sentence query by considering the following hypotheses: (1) The more similar a paraphrase used for substring extraction is to a sentence query, the more replaceable an extracted substring can be with a core term of the sentence query. (2) The more kinds of paraphrase the substring is more frequently extracted through, the more replaceable the substring can be with the core term of the sentence query. In function min of Formula 1, we check the frequency of substrings that can be mutually obtained through pairs of paraphrases. This operation enables us to extract only the terms or the sentences which are grammatically replaceable for parts of paraphrases from Web snippets without using lexical analyzers or syntax analyzers.

3.3 Scoring of Sentence Substitutions

In the next phase, we score sentence substitutions. In our hypothesis, if a sentence substitution is important for a given sentence query, the substitution should meet the following conditions: (1) The sentence substitution appears frequently on the Web. (2) The context in which the substitution appears on the Web is similar to the context in which the sentence query appears on the Web.

Based on the above hypothesis, the score of sentence substitution s for sentence query q is calculated using the following formula rank$(s|q)$:

$$\text{rank}(s|q) = \text{WebCount}(s) \cdot \text{sim}_{context}(q, s). \tag{2}$$

WebCount(s) is the total number of Web pages that a search engine returns for sentence query s. $\text{sim}_{context}(q, s)$ is the context similarity between sentence q and s on the Web. This similarity is defined as the cosine similarity between feature vectors of q and s. The feature vector of a sentence is generated as follows: First, text contents which contain the sentence are gathered by issuing the sentence as query to conventional Web search engines. We regard the collected text snippets as a document featuring the sentence. In this paper, features of the sentence vectors are defined as terms which appear in the *documents*, and a tf/idf algorithm is used to weight each feature.

4 Experiments and Results

We conducted experiments to evaluate how effective our method is for searching the Web for sentence substitutions. For this experiment, we prepared a set of 50 sentence queries and divided them into two classes. Class 1 contained 20 popular sentences, each of which appeared on at least six Web pages. Class 2 contained 30 rare sentences, each of which appeared on at most five Web pages. Table 1 shows examples from our test set.

We subjectively compared the performance of our method against that of a baseline method. The baseline method searches the Web for sentence substitutions simply by using lexico-syntactic patterns of a given sentence query. For example, given sentence query "*Germany is famous for beer*", the system generates patterns "** is famous for beer*", "*Germany is * for beer*", and "*Germany is*

Table 1. Examples of sentence queries in the test set. Each number in parentheses indicates how many Web pages the sentence query appears in.

Class	Sentence query
Popular sentence	Obama is the current president of the United States (200)
	The mouse was invented by Apple (35)
	Canberra is the capital of Australia (1378)
Rare sentence	Germany is very famous for beer (1)
	Europa was discovered by Galileo Galilei in 1610 (3)
	Sodium leads high blood pressure (0)

Table 2. Accuracy and Web access cost of two methods for popular sentence queries and rare ones. Numbers in front of the slash are for popular sentence queries, and numbers behind the slash are for rare queries.

Method	@1	@3	@5	@10	@20	Access
Our method	100/92.3	98.3/92.3	95.0/89.2	90.5/84.2	83.0/70.6	76.2/79.4
Baseline	100/88.9	93.3/79.7	88.0/76.6	81.0/65.6	70.5/46.4	3.5/3.7

*famous for **". The system then collects the sentences matching these patterns by using the method in Step 3 of Phase 2. Collected sentence substitutions are ranked according to the number of them extracted.

In our implementation, we used Yahoo! Search Boss APIs[1] to access Web search engine indices. In Phase 1, the number of search results for collecting paraphrases was fixed at 1000, and threshold θ_{edit} was set to 0.5. In Phase 2, The number of search results for collecting sentence substitutions was fixed at 50, and threshold θ_{rep} was 1. In Phase 3, we collected 50 Web documents to generate the feature vector of each sentence substitution.

4.1 Performance of Searching for Sentence Substitutions

We evaluated our algorithm from four viewpoints: (1) the accuracy of sentence substitutions ranking, (2) the processing time, and (3) the number of valid sentence substitutions. To evaluate the accuracy of the substitutions ranking, we checked the average percentage of valid sentence substitutions in the top N of the ranking. The value is denoted as @N. Here valid sentence substitutions means valid paraphrase or valid alternative sentences (generalized sentences, specified sentences, or comparative sentences) for a given sentence. We subjectively checked whether obtained substitutions are valid paraphrases or not. Note that for some sentence queries in the test set, the system failed to obtain more than N sentence substitutions. In such cases, when we obtained k results for any of such sentences ($k < N$), we supposed that (N-k) irrelevant results were additionally obtained and we calculated @N. To evaluate the processing time of each method, we counted the average number of Web accesses.

Table 2 shows the accuracy and the Web access cost of our method and the baseline method for both popular sentence queries and rare ones. For popular queries, both our method and the baseline method provided high accuracy for any @N. In contrast, for rare sentences, the baseline method provided low accuracy for @10 and @20, compared to our method. Regarding Web access cost, the baseline method was far less than our method. These results suggest that if we ignore results under the top 10 ranking, the baseline method might be better than our method because the baseline method can provide reasonably high accuracy with a low processing cost. However when we checked the number of valid sentence substitutions of the two methods for searching valid sentence substitutions, the relative situation changed.

[1] http://developer.yahoo.com/search/boss/

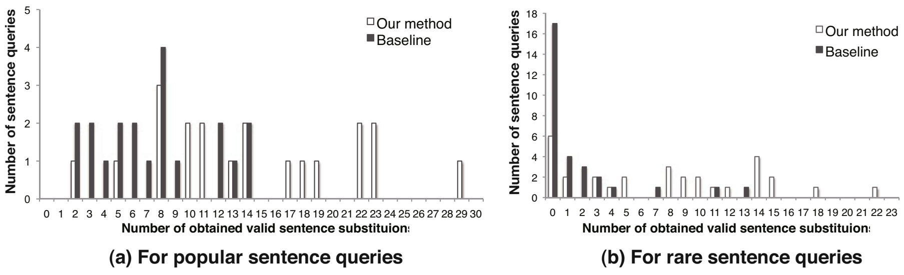


Fig. 2. Histogram of the number of obtained valid sentence substitutions. Graph (a) and (b) are histograms for popular sentence queries and rare ones, respectively.

We checked the number of valid sentence substitutions obtained by both methods. Fig. 2 shows histograms of the number of the valid sentence substitutions obtained through our method and the baseline method for popular sentence queries and rare ones. For both the popular sentence queries and the rare ones, our method on average collected more kinds of valid sentence substitution than the baseline method. Notably, our method collected many sentence substitutions for most rare queries, while the baseline method collected few or no sentence substitutions for most rare queries. When our proposed method is used, similar sentences (paraphrase candidates) of a given sentence query are generated, and the system then searches the Web for sentence substitutions using multiple similar sentences. Fig. 2 indicates that this operation worked well for collecting more sentence substitutions, expecially for rare sentence queries.

These results indicate that our method is more robust than the baseline method. Although Table 2 shows that the baseline method could be better than our method in some respects, people do not always search with popular sentence queries that appear on a lot of Web pages. Therefore, our method should be useful for rare sentence queries even if the processing cost is somewhat high.

4.2 Discussion

There are various types of sentence substitution. Therefore, we checked 874 obtained sentence substitutions through our method and manually categorized them into five classes: *paraphrases, generalized sentences, detailed sentences, comparative sentences, and irrelevant sentences for sentence queries.* Paraphrases are sentences semantically similar to sentence queries, but whose expression differs from one of the sentence queries. Generalized sentences are sentences whose meaning is broader or more general than the meaning of the sentence queries. Detailed sentences are those whose meaning is more specific or more detailed that of the sentence queries. Comparative sentences are useful for comparison with sentence queries from specific aspects.

Table 3 shows the percentage of obtained sentence substitutions belonging to each of the five classes. About 42% of the obtained sentence substitutions were paraphrases for the sentence queries. The second and third most obtained

Table 3. Percentage of sentence substitutions belonging to each of the following classes: paraphrases, generalized sentences, detailed sentences, comparative sentences, or irrelevant sentences

Class	Ratio	Examples
Paraphrase	41.6%	*Europa was discovered by Galileo Galilei in 1610* $\mapsto$ *Galileo Galilei discovered Europa in 1610*
Comparative	27.7%	*Oil will be depleted by 2050* $\mapsto$ *Oil reserves will be depleted by 2030*
Detailed	17.2%	*Kyoto was the capital of Japan in the past* $\mapsto$ *Kyoto was the capital of Japan over 1000 years*
Generalized	7.8%	*2014 winter olymics will be held in Sochi* $\mapsto$ *2014 winter olympics are to be held in Russia*
Irrelevant	5.7%	*Ozone is potent greenhouse gas* $\mapsto$ *Ozone and is a potent greenhouse gas*

sentence substitutions were comparative sentences and detailed sentences. In our method, the system searches the Web for sentence substitutions with the lexico-syntactic patterns generated from sentence queries and their paraphrases. These results indicate that our approach worked well and the system succeeded in obtaining both comparative sentences and detailed sentences for the sentence queries. On the other hand, the system obtained far fewer generalized sentences than comparative sentences and detailed sentences. This suggests that our approach does not work well for collecting generalized sentences. One possible reason for this is that the lexico-syntactic patterns generated from sentence queries keep the context of the sentence queries and so they have greater potential for collecting paraphrases, comparative sentences, and detailed sentences which include the context of the sentence query rather than generalized sentences.

5 Conclusion and Future Work

In this paper, we have proposed a method to search the Web for sentence substitutions for a given sentence query. If users issue the sentence substitutions to Web search engines, users can obtain more Web pages about the query than the initial query. Our proposed method obtains sentence substitutions from the Web using lexico-syntactic patterns generated from the input sentence and its paraphrases. Our method does not need language tools such as POS taggers or parsers. Also, huge corpora do not need to be prepared in advance because the method collects fresh corpora through Web search engines on demand. Experimental results have shown that our method can accurately collect many and various sentence substitutions, especially for rare sentences on the Web.

Several problems remain regarding the search for sentence substitutions. For example, we need a method to segment core sentences of a given sentence query to generate effective lexico-syntactic patterns. Furthermore, we need to deal with cases where input sentence queries contain inappropriate or unpopular keywords on the Web so that we can still robustly collect sentence substitutions. In the future, we plan to develop a method to automatically classify obtained sentence substitutions into classes such as paraphrases, generalized sentences, detailed sentences, and comparative sentences. We believe that such a method will enhance the ability to search for Web pages using sentence queries and knowledge mine using lexico-syntactic sentence patterns.

Acknowledgments

This work was supported in part by the following projects and institutions: Grants-in-Aid for Scientific Research (No. 18049041) from MEXT of Japan, a Kyoto University GCOE Program entitled "Informatics Education and Research for Knowledge-Circulating Society," the National Institute of Information and Communications Technology, Japan, and Grants-in-Aid for Scientific Research (No. 09J01243) from JSPS.

References

1. Craswell, N., Szummer, M.: Random Walks on the Click Graph. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), pp. 239–246 (2007)
2. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D., Yates, A.: Web-Scale Information Extraction in KnowItAll (Preliminary Results). In: Proceedings of the 13th International Conference on World Wide Web (WWW 2004), pp. 100–110 (2004)
3. Hearst, M.A.: Automatic Acquisition of Hyponyms from Large Text Corpora. In: Proceedings of the 14th Conference on Computational Linguistics (ACL 1992), pp. 539–545 (1992)
4. Jones, R., Rey, B., Madani, O., Greiner, W.: Generating Query Substitutions. In: Proceedings of the 15th International Conference on World Wide Web (WWW 2006), pp. 387–396 (2006)
5. Kaji, N., Okamoto, M., Kurohashi, S.: Paraphrasing Predicates from Written Language to Spoken Language Using the Web. In: Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004), pp. 241–248 (2004)
6. Qiu, L., Kan, M.Y., Chua, T.S.: Paraphrase Recognition via Dissimilarity Significance Classification. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006), pp. 18–26 (2006)
7. Ravichandran, D., Hovy, E.: Learning Surface Text Patterns for a Question Answering System. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL 2002), pp. 41–47 (2002)
8. Yamamoto, Y., Tanaka, K.: Finding Comparative Facts and Aspects for Judging the Credibility of Uncertain Facts. In: Vossen, G., Long, D.D.E., Yu, J.X. (eds.) WISE 2009. LNCS, vol. 5802, pp. 291–305. Springer, Heidelberg (2009)

The DISTARNET Approach to Reliable Autonomic Long-Term Digital Preservation

Ivan Subotic[1], Heiko Schuldt[2], and Lukas Rosenthaler[1]

[1]Imaging & Media Lab
[2]Databases and Information Systems Group
University of Basel, Switzerland
firstname.lastname@unibas.ch

Abstract. The rapidly growing production of digital data, together with their increasing importance and essential demands for their longevity, urgently require systems that provide reliable long-term preservation of digital objects. Most importantly, these systems have to ensure guaranteed availability over a long period of time, as well as integrity and authenticity of the preserved data and their metadata. This means that all kinds of technical problems need to be reliably handled and that the evolution of data formats is supported. At the same time, systems need to scale with the volume of data to be archived. In this paper, we present DISTARNET, a fully distributed system that reliably executes pre-defined workflows for long-term preservation. Moreover, DISTARNET is designed as an open system that allows the curators of digital objects to specify new processes to cope with additional challenges.

1 Introduction

Digital data, either digitized or digital born, are increasingly gaining importance in our everyday life. As a consequence, a large spectrum of applications require that data is preserved over long periods of time — up to several years due to legal constraints in business applications or for scientific data, for the duration of the lifetime of a human in medical applications, up to potentially unlimited time spans for the preservation of cultural heritage. Approaches to digital long-term preservation are constrained by the enormous and ever growing volumes of data. In addition, long-term data archiving and preservation also needs to take into account that data has to outlive the hardware on which they are stored and the data formats in which they are represented.

Metadata is the key to providing long-term digital preservation. We will use the term *Information Object* to denote the digital data (bit-stream) and the corresponding representation information, or some other kind of metadata.

Further, the Open Archival Information System (OAIS) reference model [1] also explicitly considers the migration of digital data to new data carriers and/or data formats in periodic intervals in order to escape from the technological obsolescence of specific hardware and software products.

In short, digital long-term preservation can be defined as the task of preserving information objects, despite potential changes of the formats in which objects are stored and the underlying hardware environment. Therefore, a software system for digital long-term preservation has to support preservation processes that guarantee i.) *integrity*: the information captured by data is not altered in any way; ii.) *authenticity*: provenance information is properly linked to each stored object by means of appropriate metadata; iii.) *chain of custody*: location and management controls are tracked within the preservation environment; iv.) *completeness*: everything that was initially stored is also available in the future and finally v.) *ease of access*: the necessary means are provided to find and identify the stored digital objects. Moreover, an essential requirement for viable long-term preservation systems is their capability to do the necessary maintenance and failure recovery in an *autonomous* way, e.g., to automatically identify when a pre-defined replication level is no longer reached and to trigger corrective actions (deploy new replicas) without human intervention.

To cope with all these challenges, we are currently developing DISTARNET (DIstributed STorage ARchival NETwork), a system for digital long-term preservation of information objects as required by archives, museums, research communities or the corporate sector. The goal is to provide a software system based on which deploying institutions can, on their own or through collaboration with others, build an autonomous, reliable, replicated, geographically distributed, Internet-based digital archiving system. Maintenance and recovery from software or hardware failures is handled in DISTARNET by means of dedicated processes that are automatically executed in a reliable way. Some of these processes encompass the necessary steps for format conversions, while other processes address failure recovery, for instance, by deploying new replicas or by migrating content from an unreliable host to a more stable one.

In the following we briefly sketch two use case scenarios that highlight the broad applicability of DISTARNET.

Scenario 1: Jim, a digital archivist at the National Museum of History & Native Art in a small European country wants to implement a new archiving solution for preserving his country's cultural heritage. This new solution should enable him to have redundant off-site replicas, although the museum itself has only one site available that can be used to deploy such a solution. However, there is a collaboration agreement between the national museums of different countries that includes access to the other institutions' computation and storage resources for deploying replicas, together with the enforcement of access restrictions on these shared data (pretty much like in a virtual organization known in the context of grid computing). Each museum deploys a DISTARNET node. When data is ingested, they will be handled according to the policies specified by their owner and will automatically be distributed across the storage resources of different museums. DISTARNET will also periodically instantiate maintenance processes and launch recovery processes when necessary.

Scenario 2: The cloud storage provider Stratocumulus Inc. plans to release DA^3S (Data Archiving as a Service), a new data management service. Essentially, DA^{3}S offers customers the option for long-term digital preservation of their digital assets, with dedicated quality of service guarantees on data availability (replication), integrity (regular checks) and authenticity. For this, Stratocumulus Inc. installs DISTARNET on each of their data-centers. In this setting, DISTARNET will be a layer underneath the cloud, so that Stratocumulus Inc. can provide the services offered by DISTARNET fully transparent to their end users. As optional services for DA^{3}S for an extra cost, Stratocumulus Inc. offers automated data format migration. Moreover, storage location constraints can be defined through preservation policies that will allow to restrict where data will be stored since Stratocumulus Inc. runs multiple data-centers around the world.

This paper introduces the flexible and reliable DISTARNET approach to digital preservation, and in particular its metadata model and the processes that are pre-defined for maintenance and failure handling purposes. We present the challenges and risks that need to be addressed by a system for long-term digital preservation. The main contribution of the paper is the detailed analysis of DISTARNET's self-$*$ capabilities, i.e., how the system is able to automatically adjust itself to changing environments (both in terms of volumes of digital content to be archived and the available storage resources) and how it automatically recovers from different kinds of failures. The autonomic behavior also includes the compliance to quality of service guarantees for the DISTARNET users (digital archivists) such as a predefined level of data availability or specific constraints on the locality of data and replica placement.

The remainder of this paper is organized as follows. In Section 2, we summarize the challenges of digital long-term preservation. Section 3 introduces DISTARNET, in particular its metadata model and processes. In Section 4 we analyze how these processes can take care of maintenance and failure recovery in digital preservation. Section 5 discusses related work, Section 6 concludes.

2 Distributed Digital Long-Term Preservation: Challenges

The challenges that distributed digital long-term preservation systems are faced with are fault tolerance, scalability, load balancing, and security in addition to integrity of complex information objects, authenticity, data format obsolescence, long-term readability, and ease of access.

Fault Tolerance and Failure Management. The failure of one or more components in a distributed preservation system should not endanger the whole system, and should only have isolated effects. Failure or disaster situations resulting in destruction or corruption of some of the stored information objects should not lead to a complete loss of the archived data. Automated replication mechanisms should maintain a minimum number of geographically dispersed replicas (number and location defined by preservation policies) of the stored information objects. Any data loss event should trigger automated recovery processes that will reestablish the minimum number of geographically dispersed replicas. This should be done by either using the repaired failed storage nodes, or by using other available and suitable storage nodes found through resource discovery.

Management of Complex Information Objects. The long-term preservation of digital data requires the management of complex information objects, i.e. information objects that are comprised of or are part of other information objects. The challenge lies in the automated management of such complex objects in a distributed setting. Preserving the integrity of complex objects is a twofold problem. First, the integrity of the referential information needs to be maintained, and second, the integrity of the objects themselves. Referential and object integrity checking needs to be automated. Any loss of integrity needs to trigger automated processes that will restore the integrity of the information object. If the information object cannot be repaired solely by the information it carries itself, other remote replicas need to be used. Besides, preserving integrity is an important challenge when information objects evolve (e.g., annotations or collection/subcollection information).

Scalability. The growing production of digital data that needs to be archived requires a scalable distributed preservation system that should work efficiently even with an increasing number of users and rapidly growing volumes of data that need to be stored. The addition of storage resources should enhance the performance of the system. This requires that the processes supporting the archiving operations be automated and scalable themselves.

Openness and Extensibility. A long-term preservation system should provide clearly separated and publicly available interfaces to enable easy extensions to existing components and the possibility of adding new components. The system should be able to be adapted to arising new challenges, by allowing curators of digital objects to specify new processes to cope with additional challenges.

Resource Discovery, Load Balancing and Remote Execution. In a distributed preservation system, the discovery of newly available resources, together with the monitoring and management of existing resources is very important and should be handled efficiently. The information gathered is important for the functioning of processes that provide automated replication of the information objects to suitable remote storage nodes, constrained through preservation policies. The system should be able to distribute the replicas among the available resources for improving performance based on availability, access speed, higher security, and/or reliability. Dynamically incorporating new resources or correctly handling the loss of existing resources (temporarily or permanently) should also be provided via automated processes with transactional semantics.

Security. Access to resources should be secured to ensure only known users are able to perform allowed operations. In addition, in a distributed where different institutions cooperate and share storage resources, only the data owning institution should be able to access and manage its data. However, cooperating institutions should be able to access other institution's meta-data and be granted access to the content of interest after having been authorized by the data owner.

3 DISTARNET

DISTARNET is a fully distributed system consisting of a network of collaborating nodes. The DISTARNET network consists of nodes organized together into virtual organizations (VO) [3]. The structure of the network inside the VOs is in itself organized in a P2P fashion, and thus creates what we call a *P2P-VO*. A DISTARNET node can be part of one or more P2P-VOs. Resources provided by nodes within a P2P-VO can only be accessed by other member nodes. Within a P2P-VO access restriction management can be used to define the allowed access characteristics to the stored content. The discovery of new resources, monitoring and management of existing resources will be done in a P2P fashion.

3.1 DISTARNET Data Model

In the context of DISTARNET the term *DISTARNET Archival Object* (DAO) will be used to denote a container holding an *Information Object* consisting of a Data Object (e.g., image, audio/video, text document, etc.) and the corresponding representation information, or some other kind of metadata. This metadata can provide additional descriptions for an information object (e.g., annotations), the descriptions of relationships between information objects (links) or information about collection/subcollection sets. Furthermore, also metadata on the object's storage and deployment can also be part of the DAO.

DISTARNET distinguishes between mutable and immutable content [12]: First, the read-only digital objects that are to be archived (e.g., images, audio/video, text documents, etc.) are considered immutable, i.e., thex cannot be modified once created. Second, the metadata of the archived digital objects is usually mutable and may exist in several versions and which can be modified (e.g., annotations pertaining to some archived digital object).

The data model used in DISTARNET allows the archiving of complex data objects. Figure 1 shows the DISTARNET logical data model in UML. In DISTARNET, every container stores one information object characterized by its type. To represent for example an annotation for an archived image, we will create a DISTARNET Archival Object of the corresponding type which will contain the annotation and make a link to the DAO containing the image – note that an annotation can be anything from text to a full-fledged DAO. The container storing the information objects corresponds to the Archival Information Package (AIP) described in the OAIS Reference Model. Although information such as annotations which are generated over time are not the original data objects that where archived, they are treated equally since they provide additional description information and need to be preserved as such alongside the originally archived data objects.

3.2 DISTARNET Processes

The goal of DISTARNET is to provide dynamic replication, automated consistency checks, and recovery of the archived digital objects utilizing autonomic

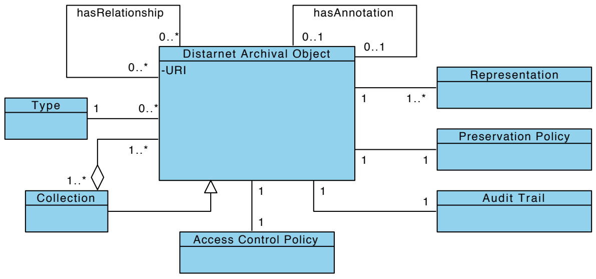


Fig. 1. Logical Data Model for DISTARNET using UML notation

behavior and predefined processes, governed by preservation policies without any centralized coordinator in a fully distributed network. Rather, the system provides two distributed repositories, namely a distributed *Replica Location Repository* that stores the degree of replication and the replica locations for each DISTARNET DAO, and a distributed *Node Information Repository* with relevant metadata on the participating nodes in a P2P-VO. Based on the information stored in these repositories, DISTARNET provides several processes that exhibit self-* properties needed for automatically dealing with the challenges of long-term digital preservation.

Self-Configuration

In DISTARNET self-configuration manifests itself in the ability of the system to automatically detect changes in the network. Events such as new nodes joining or nodes leaving are being constantly monitored and taken into account.

Node Joining Process (NJP). A node joins the network after the node credentials are configured and a number of seed nodes are added to the Neighbor Node List. The NJP then contacts the seed nodes and starts to gather information about other nodes in the network (e.g., populate the Node Information Repository). The Neighbor Node List is a dynamic list, which will contain a few *near* nodes defined as such by the P2P overlay metric.

Periodic Neighbor-Node Checking Process (PNCP). Every node will check periodically its neighbors (from the Neighbor Node List) by sending a message to which the receiver has to reply in a defined time. If the receiver does not reply, this node is marked, after some defined period of time, as lost. After that, the system will begin with the self-healing behavior.

Automated Dynamic Replication Process (ADRP). The ADRP is responsible for finding suitable storage nodes, estimating the optimal number of replicas and initiating the creation of replicas. For this, the ADRP will find – using the Node Information Repository – suitable geographically dispersed nodes for storing the replicas by taking into account possible policy-based geographical restrictions. The system will estimate the optimal number of replicas needed by taking into account the availability of nodes (based on statistics on the individual availability collected in the past) used to store a DAO and via the preservation policy imposed availability threshold of the DAO itself. This estimate will be used to raise the number of replicas if needed. To optimize the access performance the system will create if necessary additional replicas by analyzing the usage patterns of the digital objects. After evaluating a DAO regarding its overall availability in the network, ADRP will initiate if needed the *Reliable Copying Process* (RCP) and create new replicas. The reliable copying process is a BitTorrent-like transfer mode that uses existing replicas in the network for creating new ones in a secure and efficient manner. At process level, transactional semantics according to the model of transactional processes [8] will be applied.

Self-Healing

Due to the continuous monitoring of nodes, the DISTARNET system will detect abnormal conditions or problems that may harm its proper functioning (e.g., in the case of a *Node-Lost Event* or a *Corrupted DAO Event*) and it will be able to automatically recover from the following situations.

Node-Lost Event. The system will automatically react and initiate countermeasures by reevaluating the DAOs affected by the disappeared node by the ADRP and if needed create new replicas so that the policy-defined redundancy and availability requirements are upheld again.

Corrupted DAO Event. Periodic integrity checks is done by the *Periodic Integrity Checking Process (PICP)* and if integrity is breached, will automatically trigger countermeasures like finding healthy replicas in the network and by using the reliable copying process to copy them in place of the corrupted DAOs to rectify the problem.

Obsolete Data-Format. Data formats of the archived data objects are constantly monitored and warnings are issued if a given data format is becoming obsolete. The *Data-Format Migration Process (DMPP)* can be used to automatically migrate data formats by following a predefined migration path.

Self-Learning

All the mentioned properties until now can only be provided if the system has the needed information on which it can act upon. As a consequence, the DISTARNET system must know its environment, especially the available resources, and track their changes over time. This knowledge will be continuously gathered and disseminated throughout the network and be used to autonomously manage and maintain resource allocation through ADRP (e.g., finding suitable nodes where data can be replicated to, automatic policy-based geographical distribution of data, etc.) and other processes needed for the operation of DISTARNET. ADRP and PICP are triggered periodically. The parameters that trigger these processes will be adapted dynamically by the system. They will be prolonged in the case that for a longer period of time there where no changes in the network, or shortened if there where recent changes.

4 DISTARNET Maintenance and Recovery

In what follows, we analyze how DISTARNET addresses the challenges introduced in Section 2. which can be broadly categorized in issues involving *Maintenance* and *Recovery* processes.

4.1 Maintenance

DISTARNET enforces *referential integrity* (e.g., links between DAOs, collection/subcollection information) of its complex information objects using PICP.

Authenticity, *chain of custody* and *completeness* of the archived objects are supported by the data model and supporting processes through maintaining an *audit trail*. DISTARNET processes create automatically an audit record for every operation done on an DAO, with details about who, what, when, where and why the operation was executed.

The dynamic and autonomic nature of DISTARNET encounters *hardware and software obsolescence* that at the bitstream level endanger the DAOs by means of automated "media" migration, by allowing to simply turn off the old hardware and turn on the new hardware.

The *interpretability* of the logical representation of the DAOs is guarantied by processes providing automated data format migration and is supported by the data model and the collection of extended format descriptions by using format identification and characterization tools.

Scalability, *resource discovery* and *load balancing* are supported by a fully distributed design of the network, adaptability of the triggers to lower the overall load on the nodes caused by the maintenance processes, and by the ADRP which also optimizes the usage of the storage resources provided in the network.

Openness and *extensibility* are very important attributes which DISTARNET supports by allowing the curators to define new process, and by providing a flexible data model that can adapt to new needs that can arise in the future (e.g., in case of new data formats, new metadata standards).

4.2 Recovery

Due to the continuous monitoring of nodes, the DISTARNET system will detect abnormal conditions or problems that may harm its proper functioning and it will be able to automatically recover from those situations, again by means of predefined processes.

Hardware problems caused by failure (e.g., power failure, hardware failure, etc) or disaster (e.g., natural disaster, fire) can result in destruction of the whole node, or in destruction or corruption of the stored DAO. The loss of a node is discovered through the *PNCP* which triggers a *node-lost event* after some predefined period of time, because at first a *network problem* is assumed. In the case of a ode-lost event, this information is stored in the *Node Information Repository*, the *Replica Location Repository* is updated, and both are propagated throughout the network. Subsequently, the ADRP is started for the DAOs that are affected (network wide redundancy) by the lost node.

Network problems caused by lost network connection or an intermittent network connection are detected through the *PNCP*. For a lost connection to a node to be classified as a network problem, upon the return of a node it has to be verified that the node was running, only the network was down, and all DAOs are accounted for. Such discovered network problems trigger an entry into the Node Information Repository and are taken into consideration (e.g., reliability of a node) by the ADRP.

Problems with the *content* of the archive caused through the corruption (e.g., hardware problems, malicious acts, etc.) of the DAOs is discovered through the *PICP*. Any detected corruption is logged correspondingly in the *Node Information Repository* where it speaks about the reliability of a specific node, in the *Replica Location Repository*, and in the audit trail of the DAO for future reference. Also subsequently the ADRP is triggered to resolve the problem.

DISTARNET processes are able to *self recover* from problems encountered during their execution. Every DISTARNET process is composed of single atomic tasks or other (sub)processes. The workflow engine responsible for the execution the processes, monitors and logs every step. In the event of failure of any step, the corresponding recovery process, either of the main process or of the subprocess is triggered. Further we divide the processes in *repeatable* and *non-repeatable* processes. The recovery of *repeatable* processes consist in the repeated execution of those processes either until the process succeeds or a predefined number of retries is reached. In the case of *non-repeatable* process, the recovery process consist of clean-up tasks to end the process in a consistent state.

5 Related Work

For the design of digital archives, two different approaches can be distinguished. First, a centralized approach which is followed by Kopal [5] and SHERPA DP [4]. This approach allows to easily control the archived content but implies rather high infrastructure and maintenance costs and the inflexibility to cope with rapidly increasing loads if they exceed the initial design. Second, a distributed approach followed by LOCKSS [7], Cheshire 3 [13], and SHAMAN [9] where the archiving infrastructure is distributed across the participants' sites. These systems have lower infrastructure and maintenance costs, the ability for virtually unlimited growth, and allow for a higher degree of availability and reliability. DISTARNET is built following the distributed approach.

Fedora Commons [6] and DSpace [11] are open source projects that allow the creation of Digital Repositories for managing digital objects. Those two initiatives have been merged in the context of DuraCloud [2] which offers both storage and access services, including content replication and monitoring services that can span multiple cloud-storage providers. The idea is that Digital Repositories using DuraCloud can expand their systems and provide long-term preservation capabilities, or individuals can use directly use DuraCloud as a long-term preservation system. DuraCloud and DISTARNET are similar as both provide processes for the management of digital objects necessary for digital long-term preservation. The difference lies in the flexibility of the DISTARNET processes that manage digital objects between DISTARNET nodes by using local storage or, if necessary, by using cloud based storage.

Hoppla [10] is an archiving solution that combines back-up and fully automated migration services for data collections in environments with limited resources for digital preservation. Similar to Hoppla, DISTARNET also provides processes that support data format migration for the archived digital objects.

6 Conclusion and Future Work

In this paper, we have presented DISTARNET's self-healing and self-adaptation features. DISTARNET is an ongoing effort to build a long-term digital preservation system that will provide a fully distributed, fault tolerant archiving environment with autonomic behavior governed by preservation policies. DISTARNET autonomously provides dynamic replication, automated consistency checking and recovery of the archived digital objects. As part of DISTARNET, we have developed a highly flexible data model that takes into account user generated annotations or collections and arbitrary links between objects. Based on that, we have specified sophisticated management processes that are triggered automatically when a violation of one of the preservation policies is detected.

In our future work, we plan to run a series of test cases that will simulate the real world usage of the DISTARNET network and evaluate the autonomic behavior of the system. A further emphasis will be put on testing the scalability and performance of the system under load.

References

1. CCSDS: Reference Model for an Open Archival Information System (OAIS). CCSDS 650.0-B-1, Consultative Committee for Space Data Systems
2. Duracloud, http://duraspace.org/duracloud.php
3. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. Int'l Journal of Supercomputer Applications 15(3) (2001)
4. Knight, G.: SHERPA DP: Establishing an OAIS-Compliant Preservation Environment for Institutional Repositories. In: Digital Repositories, pp. 43–48 (2005)
5. Kopal, http://kopal.langzeitarchivierung.de/
6. Lagoze, C., Payette, S., Shin, E., Wilper, C.: Fedora: an Architecture for Complex Objects and their Relationships. Int'l Journal on Digital Libraries 6, 124–138 (2006)
7. Reich, V., Rosenthal, D.S.H.: LOCKSS (Lots Of Copies Keep Stuff Safe). The New Review of Academic Librarianship 6, 155–161 (2000)
8. Schuldt, H., Alonso, G., Beeri, C., Schek, H.J.: Atomicity and Isolation for Transactional Processes. ACM TODS 27(1), 63–116 (2002)
9. Shaman, http://shaman-ip.eu/shaman/
10. Strodl, S., Petrov, P., Greifeneder, M., Rauber, A.: Automating Logical Preservation for Small Institutions with Hoppla. In: Lalmas, M., Jose, J., Rauber, A., Sebastiani, F., Frommholz, I. (eds.) ECDL 2010. LNCS, vol. 6273, pp. 124–135. Springer, Heidelberg (2010)

11. Tansley, R., Bass, M., Smith, M.: DSpace as an Open Archival Information System: Current Status and Future Directions. In: Koch, T., Sølvberg, I.T. (eds.) ECDL 2003. LNCS, vol. 2769, pp. 446–460. Springer, Heidelberg (2003)
12. Voicu, L.C., Schuldt, H., Akal, F., Breitbart, Y., Schek, H.J.: Re:GRIDiT - Coordinating Distributed Update Transactions on Replicated Data in the Grid. In: 10th IEEE/ACM International Conference on Grid Computing, pp. 120–129 (2009)
13. Watry, P., Larson, R.: Cheshire 3 Framework White Paper. In: International Symposium on Mass Storage Systems and Technology, pp. 60–64 (2005)

A Unified Algorithm for Continuous Monitoring of Spatial Queries

Mahady Hasan, Muhammad Aamir Cheema, Xuemin Lin, and Wenjie Zhang

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

Abstract. Continuous monitoring of spatial queries has gained significant research attention in the past few years. Although numerous algorithms have been proposed to solve specific queries, there does not exist a unified algorithm that solves a broad class of spatial queries. In this paper, we first define a *versatile top-k query* and show that various important spatial queries can be modeled to a versatile top-k query by defining a suitable scoring function. Then, we propose an efficient algorithm to continuously monitor the versatile top-k queries. To show the effectiveness of our proposed approach, we model various inherently different spatial queries to the versatile top-k query and conduct experiments to show the efficiency of our unified algorithm. The extensive experimental results demonstrate that our unified algorithm is several times faster than the existing best known algorithms for monitoring constrained k nearest neighbors queries, furthest k neighbors queries and aggregate k nearest neighbors queries.

1 Introduction

With the availability of inexpensive mobile devices, position locators and cheap wireless networks, location based services are gaining increasing popularity. Some examples of the location based services include fleet management, geo-social networking (also called location-based networking), traffic monitoring, location-based games, location based advertisement and strategic planning etc. Due to the popularity of these services, various applications have been developed that require continuous monitoring of spatial queries. For example, a person driving a car may issue k nearest neighbors (kNN) query to continuously monitor k closest restaurants. Similarly, a taxi driver might issue a query to continuously monitor the passengers that are within 5 Km of his location. Cabspotting[1] and Zhiing[2] are two examples of such applications.

Driven by such applications, continuous monitoring of spatial queries has received significant research attention in the past few years. For example, several algorithms have been proposed to answer kNN queries [14,22,20], range queries [8,12,18,3], constrained kNN queries [7,9] and aggregate kNN queries [14].

[1] http://cabspotting.org/faq.html
[2] http://www.zhiing.com/how.php

J.X. Yu, M.H. Kim, and R. Unland (Eds.): DASFAA 2011, Part II, LNCS 6588, pp. 104–118, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Although various algorithms have been proposed to solve each of these spatial queries, to the best of our knowledge, there does not exist a unified algorithm that solves all the above mentioned queries. In this paper, we propose a unified algorithm to monitor a broad class of spatial queries including the above mentioned spatial queries.

In Section 3.1, we first define a *versatile top-k query* and then show that various spatial queries can be modeled to the versatile top-k query (Section 3.2). Given a set of objects, a versatile top-k query reports k objects with the lowest scores. The score of each object is computed by using a *versatile scoring function $vsf()$* (the properties of a versatile scoring function are formally defined in Section 3). Various spatial queries can be modeled to the problem of a versatile top-k query by choosing a suitable scoring function. For example, a kNN query can be modeled to a versatile top-k query by choosing a scoring function such that $vsf(o) = dist(o, q)$ where $dist(o, q)$ returns the Euclidean distance between the object o and a reference point q (called query point in spatial queries).

We present an efficient algorithm to continuously monitor versatile top-k queries. Our unified algorithm can efficiently monitor any spatial query that can be modeled to versatile top-k queries by defining a suitable scoring function. To monitor any of these spatial queries, the only change we need to make in the implementation is to add a new scoring function for that specific spatial query. The unified algorithm then uses this scoring function to monitor the spatial query.

To show the effectiveness of our approach, we choose various inherently different spatial queries and model these queries to versatile top-k queries. Then, we conduct experiments to show the efficiency of our unified algorithm. More specifically, we show the performance of our algorithm for monitoring constrained kNN queries, furthest k neighbor queries and aggregate kNN queries.

Below we summarize our contributions in this paper.

- We define versatile top-k queries and show that various spatial queries can be modeled to a versatile top-k query by choosing a suitable scoring function.
- To the best of our knowledge, we are first to present a unified algorithm to efficiently monitor various spatial queries. We prove that our algorithm is optimal in number of grid cells it visits to monitor the spatial queries.
- We conduct extensive experiments to study the performance of our unified algorithm for monitoring various inherently different spatial queries. The experimental results show that our unified algorithm outperforms existing best known algorithms for these specific queries.

2 Background

2.1 Preliminaries

In this sectiom, we formally define few inherently different spatial queries.

k nearest neighbors query. A kNN [10,16,11,14,22,20] query returns the k closest objects from the query point q. Given a set of objects O and a query point q, the kNN query returns a set N of k objects such that for each $n_i \in N$ the $dist(n_i, q) \leq dist(o', q)$ where $o' \in O - N$.

kNN queries have a wide range of applications. For example, a person may issue a kNN queries to find k closest restaurants to his location.

Constrained k nearest neighbors query. A constrained kNN query [7,9] returns k objects closest to the query point q among the objects that lie inside a constrained region CR. Given a set of objects O, a query point q and a constrained region CR, the constrained kNN query returns a set N containing k objects such that for each $n_i \in N$, $dist(n_i, q) \leq dist(o', q)$ where $o' \in O - N$ and both o' and n_i lie in the constrained region CR.

Consider that a user wants to find k post offices closest to his location from a suburb named Randwick. He may issue a constrained kNN query where the constrained region corresponds to the suburb Randwick.

Furthest k neighbors query. A furthest k neighbors query [17,6,1] returns k furthest objects from the query point q. Given a set of objects O and a query point q, the furthest k neighbors query returns an answer set N containing k objects such that for each $n_i \in N$, $dist(n_i, q) \geq dist(o', q)$ where $o' \in O - N$.

Consider that a town planner wants to establish a service center in a town. Before establishing this service center, the town planner may need to find the furthest service user to identify the maximum range the service would need to cover.

Aggregate k nearest neighbors query. Given a set of objects O and a query set Q with m numbers of instances $\{q_1, ..., q_m\}$, the aggregate kNN query [21,15,13] returns k objects with the minimum aggregate distance from Q. Let $aggdist(o, Q)$ be the aggregate distance of an object o from Q. An aggregate kNN query returns an answer set N containing k objects such that for each $n_i \in N$, $aggdist(n_i, Q) \leq aggdist(o', Q)$ where $o' \in O - N$. Below we define the aggregate distance functions for some common aggregate kNN queries.

1. Sum-aggregate kNN query uses $aggdist(o, Q) = \sum_{q_i \in Q} dist(o, q_i)$.
2. Max-aggregate kNN query uses $aggdist(o, Q) = \max_{q_i \in Q}(dist(o, q_i))$.
3. Min-aggregate kNN query uses $aggdist(o, Q) = min_{q_i \in Q}(dist(o, q_i))$.

Consider that a group of friends wants to meet at a restaurant such that the total distance traveled by them to reach the restaurant is minimum. A sum-aggregate kNN query ($k = 1$) returns the location of such a restaurant.

2.2 Related Work

[22,20,14] are some notable techniques that use grid-based index to monitor spatial queries. Conceptual Partitioning Monitoring (*CPM*) technique [14] organizes the grid cells into conceptual rectangles and assigns each rectangle a direction and a level number. The direction is *R, D, L, U* (for right, down, left, up) and the level number is the number of cells in between the rectangle and q as shown in Fig. 1.

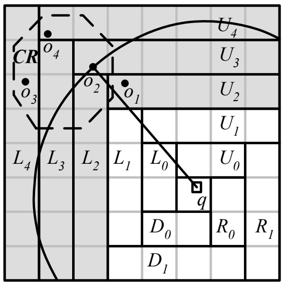


Fig. 1. CPM Constrained NN

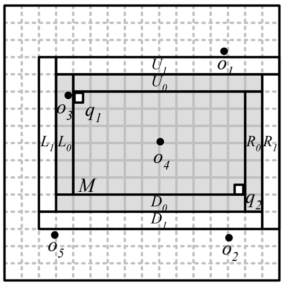


Fig. 2. Cells accessed by CPM

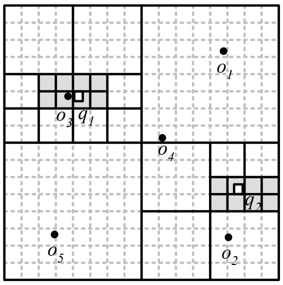


Fig. 3. Cells accessed by an optimal algorithm

CPM first initializes an empty min-heap H. It inserts the query cell c_q with key set to zero and the level zero rectangles (R_0, D_0, L_0, U_0) with the keys set to minimum distances between the query q and the rectangles into H. The entries are de-heaped iteratively. If a de-heaped entry e is a cell then it checks all its objects and updates $q.kNN$ (the set of kNN for the query q) and $q.dist_k$ (the distance of current k^{th}NN from q). If e is a rectangle, it inserts all the cells inside the rectangle and the next level rectangle in the same direction into the heap H. The algorithm stops when the heap becomes empty or when e's distance is greater than $q.dist_k$.

The update of each object is handled as follows. (1) Incoming update: CPM removes the k^{th} NN and inserts the object in $q.kNN$. (2) Outgoing update: CPM removes the object from $q.kNN$ and finds the next NN by visiting the remaining entries in the heap. In case the query moves, CPM starts from the scratch. CPM outperforms both YPK-CNN [22] and SEA-CNN [20]. CPM can also be extended to solve few other spatial queries.

CPM can be used to answer continuous *constrained kNN* queries by making a small change. More specifically, the algorithm inserts only the rectangles and the cells that intersect the constrained region into the heap. Figure 1 shows an example where the constrained region is a polygon R. The constrained NN is o_2 and the rectangles/cells shown shaded are inserted into the heap by CPM.

CPM can also be extended to answer *Aggregate kNN* queries. In case of a conventional kNN query, the algorithm starts with the query cell c_q. For aggregate kNN query, the algorithm computes a rectangle M such that all query instances lie in M. At the initial phase, the algorithm inserts the rectangle M and the zero level rectangles in the heap with their aggregated distance $aggdist(e, Q)$ (e.g., $\min_{q_i \in Q} dist(e, q_i)$) from the query set Q. For instance, in the Fig. 2 the min-aggregate NN is o_3 and all rectangles shown in solid lines are inserted in the heap to compute o_3. Please note that the algorithm inserts all the shaded cells into the heap. Note that CPM inserts many un-necessary cells into the heap. Fig. 3 shows the minimum number of cells (shown shaded) that are needed to be inserted in heap by an optimal algorithms. We show that our approach significantly reduces the number of cells inserted in the heap.

CircularTrip [5] and iSEE [19] also efficiently monitor kNN queries. CircularTrip visits the cells around query point round by round until all NNs are retrieved. On the other hand iSEE computes a visit order list around the query point to efficiently answer the kNN query. However, extension of these algorithms for other spatial queries (e.g., aggregate kNN query) is non-trivial.

3 Problem Definition

Let p be a point and R and R_c be two hyper-rectangles in a d-dimensional space $\mathbb{R}^d$. If R contains R_c (i.e., R_c is inside the hyper-rectangle R) then R_c is called the child of R. Consider a function $f(p)$ that returns the score of a given point p. An upper bound score $S^U(R)$ of a hyper-rectangle R is defined as,

$$S^U(R) = \max_{p \in R}(f(p))$$

where, $p \in R$ denotes a p that lies in the hyper-rectangle R. Similarly, the lower bound score $S^L(R)$ is defined as,

$$S^L(R) = \min_{p \in R}(f(p))$$

Versatile Scoring function: A function $f()$ is called a versatile scoring function iff $S^U(R) \geq S^U(R_c)$ and $S^L(R) \leq S^L(R_c)$ for any R and R_c where R_c is a child rectangle of R. We denote the versatile scoring function as $vsf()$. The versatile score of a given point p is denoted as $vsf(p)$.

Consider a function $f(p) = dist(p, q)$ where $dist(p, q)$ is the Euclidean distance[3] between the point p and a given point q. Hence, the upper bound score $S^U(R)$ is the maximum Euclidean distance between the rectangle R and the fixed point q. Similarly, the lower-bound score $S^L(R)$ is the minimum Euclidean distance between the rectangle R and the fixed point q. Note that $f(p) = dist(p, q)$ is a versatile scoring function.

3.1 Versatile Top-k Queries

Consider a set of objects $O = \{o_1, \ldots, o_N\}$ where o_i denotes the spatial location of ith object. Also, consider a versatile scoring function $vsf()$ to compute the score of the objects. A top-k query returns a set of k objects $N = \{n_1, \ldots, n_k\}$ such that $vsf(n_i) \leq vsf(o')$ for any $n_i \in N$ and any $o' \in O - N$. Hence, top-k query returns k objects having smallest scores.

In this paper we study the continuous monitoring of top-k query where the top-k results are updated with the changes in the datasets. We follow *timestamp* model where the results are required to be updated after every t time units.

[3] Other L_p distance metrics can also be used.

3.2 Modeling Spatial Query to Top-k Query

We can model various spatial queries to a versatile top-k query by defining a suitable versatile scoring function. The versatile scoring functions for some popular spatial queries are given below.

k nearest neighbors queries:

$$vsf(o) = dist(o, q)$$

Here, the $dist(o, q)$ is the Euclidean distance between an object o and the query point q.

Furthest k neighbors queries:

$$vsf(o) = -dist(o, q)$$

Please not that the object furthest from the query q has the smallest score. Hence, the further objects are preferred in this case.

Aggregate k nearest neighbors queries:

Below we define the scoring functions for Sum, Max and Min aggregate k nearest neighbors queries, respectively.
i) Sum-Aggregate k nearest neighbors queries: $vsf(o) = \sum_{q_i \in Q} dist(o, q_i)$
ii) Max-Aggregate k nearest neighbors queries:$vsf(o) = \max_{q_i \in Q}(dist(o, q_i))$
iii) Min-Aggregate k nearest neighbors queries: $vsf(o) = \min_{q_i \in Q}(dist(o, q_i))$

Constrained kNNs queries:

$$vsf(o) = \begin{cases} dist(o, q), & \text{if } o \text{ lies inside the constrained region } CR; \\ \infty, & \text{otherwise.} \end{cases}$$

Note that we can also define the versatile scoring functions for other queries like constrained furthest neighbors query and constrained aggregate kNNs query etc.

Next, we define the versatile scoring function to model another spatial query which is not essentially a top-k query. This demonstrates that our proposed unified algorithm can be applied to answer several other queries that are not top-k queries.

Circular Range queries: Given a set of objects O, a query point q and a positive value r. A circular range query [8,12,3] returns every object $n \in O$ that lies within distance r of the query location q (i.e., every object such that $dist(n, q) \leq r$). We call such query a circular range query because the search space is a circle around the query point q with the radius r. Below we define the versatile scoring function for the circular range query.

$$vsf(o) = \begin{cases} 1, & \text{if } dist(o, q) \leq r; \\ \infty, & \text{otherwise.} \end{cases}$$

Here, r is the radius of the circular range query.

To model the circular range query to a versatile top-k query we need to make the following small changes: i) every object with score equal to kth object's score must also be reported: ii) an object with score ∞ must not be reported. Note that we if the range contains more than k objects then all the objects inside the range are reported. On the other hand, if the range contains less than k objects then objects outside the range are not reported.

4 Technique

4.1 Conceptual Grid Tree

In this section, we briefly describe the conceptual grid tree which we introduced in [4] and later used to answer other spatial queries in [3,9]. The conceptual grid tree is the backbone of our approach. First, we briefly describe the grid index and then we describe the conceptual grid tree which is a conceptual visualization of the grid index.

In a grid based index ,the whole space[4] is divided into a number of cells, where the size of each cell is $\delta \times \delta$. Hence, the extent of each cell in a dimension is δ. A particular cell is denoted as $c[i, j]$ where i is the column number and j is the row number. The lower left cell of the grid is $c[0, 0]$. An object o with the position (x, y) is located into the cell $c[\lfloor x/\delta \rfloor, \lfloor y/\delta \rfloor]$. I.e., a cell $c[i, j]$ contains all the objects with x-coordinate in the range $i.\delta$ to $(i + 1).\delta$ and y-coordinate in the range $j.\delta$ to $(j + 1).\delta$.

In our proposed conceptual grid tree structure we assume a grid that consists of $2^n \times 2^n$ cells[5]. The grid is treated as a conceptual tree where the *root* contains all $2^n \times 2^n$ grid cells. Each intermediate entry e in a level l (for root $l = 0$) is recursively divided into four children of equal sized rectangles such that each child of an entry e contains $x/4$ cells where x is the number of cells contained by the intermediate entry e. I.e., if an entry e at level l contains $2^{n-l} \times 2^{n-l}$ cells then each child of the entry e will contain $2^{n-l-1} \times 2^{n-l-1}$ cells. Every leaf level entry contains four grid cells.

The root, intermediate entries and the grid cells are shown in Fig. 4. In Fig. 4 the grid size is 4×4 (i.e., $2^2 \times 2^2$ grid cells). Hence, the root contains all $2^2 \times 2^2$ cells. An intermediate entry with level 1 contains $2^{2-1} \times 2^{2-1}$ cells (i.e., four cells).

Please note that the Grid-tree is just a conceptual visualization of the grid and it does not exist physically (i.e., we do not need pointers to store entries and its children). In Fig. 4 the rectangles with dotted lines are considered as conceptual structure and the rest are physical structure. Therefore, root and the intermediate entries are conceptual only and they are not stored in the memory.

[4] For the ease of demonstration of our algorithm we use two dimensional space, although our technique can be applied to higher dimensional space.

[5] If the grid size is not $2^n \times 2^n$, it can be divided into several smaller grids such that each grid is $2^i \times 2^i$ for $i > 0$. For example, a 8×10 grid can be divided into 5 smaller grids (i.e., one 8×8 grid and four 2×2 grids).

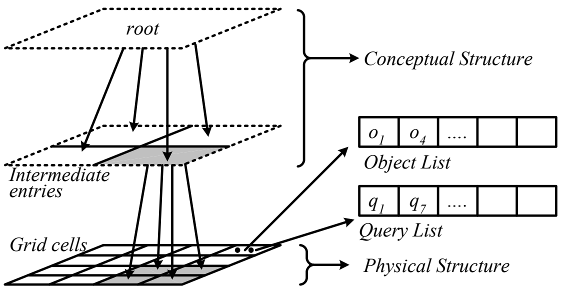


Fig. 4. Conceptual Grid Tree Structure

To retrieve the children of an entry (or root), we divide its rectangle into four equal sized rectangles such that each child has side length $d/2$ where d is the side length of its parent. A rectangle with side length equal to δ (the width of a grid cell) refers to a cell $c[i, j]$ of the grid.

4.2 Unified Algorithm

Initial computation. Most of the spatial queries algorithms that can be applied on other tree structure (e.g., R-tree) can easily be applied on the conceptual grid tree. The advantage of using this grid tree over previously used grid based access methods is that if an intermediate entry of the tree lies in the pruned region then none of its cells are accessed.

Algorithm 1. CGTree-based Unified Initial Computation

```
Input:  $q$: query point with the versatile scoring function($vsf()$); $k$: an integer
Output: top-$k$ query results
 1: $q.score_k = \infty$; $q.kA = \phi$; $H = \phi$
 2: Initialize a min-heap $H$ with root entry of the conceptual grid tree
 3: while $H \neq \phi$ do
 4:    de-heap an entry $e$
 5:    if $S^L(e) \geq q.score_k$ then
 6:       return $q.kA$
 7:    if $e$ is a cell in the grid then
 8:       update $q.kA$ and $q.score_k$ by the objects in $e$
 9:    else
10:       for each of the four children $c$ do
11:          if $S^L(c) \leq q.score_k$ then
12:             insert $c$ into $H$ with key $S^L(c)$
13: return $q.kA$
```

The initial computation of the unified algorithm using the Conceptual Grid-Tree is presented in Algorithm 1. The main idea is similar to that of applying BFS search [11] on R-tree based data structure. Specifically, the algorithm starts by inserting the root of the Grid-tree into a min-heap. The algorithm iteratively de-heaps the entries. If a de-heaped entry e is a grid cell then it visits the cell and updates $q.kA$ and $q.score_k$ where $q.kA$ is the answer set and $q.score_k$ is the kth smallest score of objects in $q.kA$ (line 8). If $|q.kA| < k$ (i.e, the size of the answer set is less then k) then $q.score_k$ is set to infinity. Please recall that the width of a cell is δ. So, the algorithm checks the width of each entry e to identify whether e is a grid cell or not (line 7).

If the de-heaped entry e is not a grid cell, then the algorithm inserts its children into the heap H with their lower bound scores (lines 10 to 12). The algorithm terminates when the heap becomes empty (line 3) or when a de-heaped entry e has its lower bound score $S^L(e) \geq q.score_k$ (line 5). This guarantees the correctness of the algorithm. This is because any cell c for which $S^L(c) \geq q.score_k$ cannot contain an object that has a score smaller than $q.score_k$ (and cannot be the answer for this reason). When the de-heaped entry e has its lower bound score $S^L(e) \geq q.score_k$, every remaining entry e' in the heap H has its lower bound score $S^L(e') \geq q.score_k$ because the entries are accessed in ascending order of their lower bound scores.

Continuous Monitoring. Before we present the continuous monitoring algorithm, we introduce the data structure that is used for efficient update of the results.

The system stores a query table and an object table to record the information about the queries and the objects. An object table stores the id and location of all objects. The query table stores the query id, query location, the answer set $q.kA$ and the *cellList* (cells that the query has visited to retrieve all objects in the answer set $q.kA$).

Each cell of the grid stores two lists namely *object list* and *query list*. The object list of a cell c contains the object id of every object that lies in c. The query list of a cell c contains the id of every query q that has visited c (by visiting c we mean that it has considered the objects that lie inside it (line 8 of Algorithm 1)). The query list is used to quickly identify the queries that might have been affected by the object movement in a cell c.

Handling a single update: Assume that an object o reports a location update and o_{old} and o_{new} correspond to its old and new locations, respectively. The object update can affect the results of a query q in the following three ways;

1. *internal update:* $vsf(o_{old}) \leq q.score_k$ and $vsf(o_{new}) \leq q.score_k$; clearly, only the order of the answer set may have been affected, so we update the order of $q.kA$ accordingly.
2. *incoming update:* $vsf(o_{old}) > q.score_k$ and $vsf(o_{new}) \leq q.score_k$; this means that o is now a part of $q.kA$. Hence, o is inserted in $q.kA$.
3. *outgoing update:* $vsf(o_{old}) \leq q.score_k$ and $vsf(o_{new}) > q.score_k$; i.e., o is not part of the answer set anymore. Therefore, we delete o from $q.kA$.

The complete update handling module: The update handling module consists of two phases. In the first phase, we receive the object updates. For each object update, we reflect its effect on the results according to the three scenarios described earlier. In the second phase, we compute the final results. Algorithm 2 presents the details.

Algorithm 2. Continuous Monitoring

```
Input:    location updates
Output:   $q.kA$
Phase 1: receive updates
 1: for each object update $o$ do
 2:    Affected queries $Q_{aff} = c_{o_{old}}.q\_list \cup c_{o_{new}}.q\_list$
 3:    for each query $q$ in $(Q_{aff})$ do
 4:       if internal update; update the order of $q.kA$
 5:       if incoming update; insert $o$ in $q.kA$
 6:       if outgoing update; remove $o$ from $q.kA$
Phase 2: update results
 7: for each query $q$ do
 8:    if $|q.kA| \geq k$; keep top $k$ objects in $q.kA$ and update $q.score_k$
 9:    if $|q.kA| < k$; expand $q.kA$
10: return $q.kA$
```

Phase 1: First, we receive the object updates and for each object update, we identify the queries that might have been affected by this update. It can be immediately verified that only the queries in the query lists of c_{old} and c_{new} may have been affected where c_{old} and c_{new} denote the old and new cells of the object, respectively. For each affected query q, the update is handled (lines 3 to 6) as mentioned previously (e.g., internal update, incoming update or outgoing update).

Phase 2: After all the updates are received, the results of the queries are updated as follows; if $q.kA$ contains more than k objects in it (more incoming updates than the outgoing updates), the results are updated by keeping only the top k objects. Otherwise, if $q.kA$ contains less than k objects, we expand the search region so that $q.kA$ contains k objects. The expansion is similar to the Algorithm 1 except the following changes. Any entry e that has $S^U(e) \leq q.score_k$ are not inserted into the heap. This is because such entries have already been explored. The stopping criteria is same as the initial computation i.e., we stop when a de-heaped entry e has $S^L(e) \geq q.score_k$.

If a query changes its location the versatile score become invalid. Hence, the results are computed by calling the Algorithm 1 (i.e., we compute the result for the query from the scratch).

Proof of optimality and correctness. Before we prove the optimality, we define two terms; *accessing* and *visiting* a cell. We say that a cell has been accessed if the algorithm inserts it in the heap (e.g., line 12 of Algorithm 1). If a cell is de-heaped from the heap and the algorithm retrieves the objects in this cell, we say that the cell has been visited by the algorithm (e.g., line 8 of Algorithm 1). Please note that the cost of visiting a cell is usually significantly higher than the cost of accessing a cell.

We prove that our algorithm is opitmal in number of visited cells (i.e., it does not visit any unnecessary cell to answer the query). To prove the correctness, we show that our algorithm visits all the cells that must be visited to compute the correct results.

Proof. Let $q_{old}.score_k$ and $q_{new}.score_k$ be the scores of k^{th} object before and after the update, respectively. Consider the case when $q_{old}.score_k \geq q_{new}.score_k$ (i.e., the number of incoming updates is at least equal to the number of outgoing updates). This implies $|q.kA| \geq k$ (line 8 of Algortihm 2) and we do not need to visit any new cell to update the result. Therefore, we only need to consider the case when $q_{old}.score_k < q_{new}.score_k$ (line 9 of Algorithm 2). Below, we prove the optimality and correctness of our algorithm for this case.

Let C be the set of minimum cells that have to be visited in order to guarantee the correct results. First, we identify C and show that our algorithm does not visit any unnecessary cell c' such that $c' \notin C$. A cell c' for which $S^U(c') \leq q_{old}.score_k$ is not required to be visited. This is because all the objects in this cell have been considered earlier. Similarly, a cell c' for which $S^L(c') \leq q_{new}.score_k$ is not required to be visited. This is because every object in such cell has score at least equal to $q_{new}.score_k$. Therefore, the set C of minimum cells consists of every cell c that satisfies the following two inequalities.

$$S^U(c) > q_{old}.score_k \qquad (1)$$
$$S^L(c) < q_{new}.score_k \qquad (2)$$

Please note that in our update handling algorithm, we ignore the cells that have $S^U(c) \leq q_{old}.score_k$ and terminate the algorithm when $S^L(c) \geq q_{new}.score_k$ (see Section 4.2 Phase 2). Thus, we satisfy both of the above inequalities. Therefore, our algorithm does not visit any un-necessary cell and is optimal in the number of visited cells.

Please note that the initial computation can be considered as a special case of update handling where $q_{old}.score_k$ is set to zero.

As a proof of correctness, we show that our algorithm visits all the cells in the set C. Recall that we maintain the cells in a heap based on their lower bound scores. Therefore, the cells are visited in the ascending order of their lower bound scores and it guarantees that every cell c for which $S^L(c) < q_{new}.score_k$ is visited.

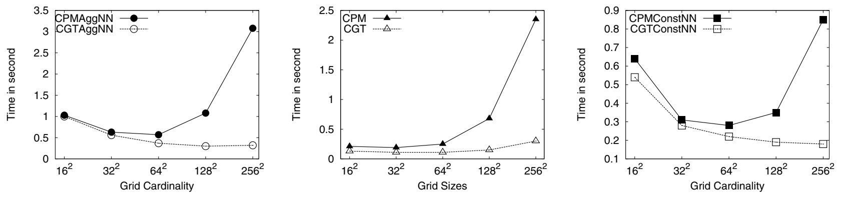
5 Experiments

We choose three inherently different spatial queries and run experiments to evaluate the efficiency of our unified algorithm. More specifically, we run the experiments for the constrained kNN queries, the aggregate kNN queries and the furthest k neighbors queries. Since our algorithm is based on the **C**onceptual **G**rid **T**ree, we refer to it as CGT.

We compare our algorithm with CPM [14]. As mentioned by the authors, it can be modified to answer constrained kNN and aggregate kNN queries. We extend CPM to answer furthest k neighbors query as follow. We compute the furthest conceptual rectangles from the query cell c_q in all four directions (i.e., right, down, left, up). Initially, we insert the furthest rectangles in the heap with their keys set to the maximum distances between the query and the rectangles.

Table 1. System Parameters

Parameter	**Range**
Number of objects (×1000)	20, 40, 60, 80, **100**
Number of queries	100, 500, **1000**, 2500, 5000
Value of k	2, 4, 8, **16**, 32, 64, 128
Object/query agility (in %)	10, 30, **50**, 70, 90
Aggregate function (for aggregate queries only)	**sum**, max, min
Number of query instances (for aggregate queries only)	5, **10**, 25, 50, 100



(a) Aggregate *k*NN (b) Furthest *k* neighbor (c) Constrained *k*NN

Fig. 5. Effect of grid cardinality

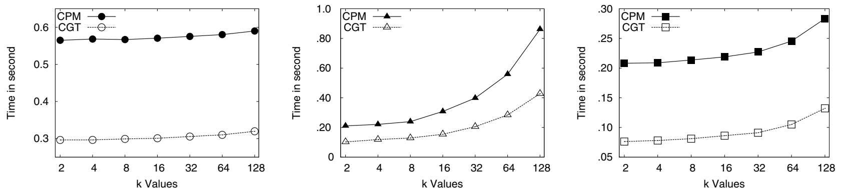
After de-heaping a rectangle, the previous level (closer to the query cell) rectangle in the same direction is inserted in the heap. We use a max heap and thus retrieve the rectangles in descending order of their maximum distances from the query.

Our experiment settings are similar to those used in [14]. More specifically, we use Brinkhoff data generator [2] to generate objects moving on the road network of Oldenburg, a German city. The queries are generated similarly. Each query is monitored for 100 time stamps and the experiment figures show the total computation time for a single query for the duration of 100 time stamps. Table 1 shows the parameters used in our experiments and the default values are shown in bold. Agility corresponds to the percentage of objects and queries that issue location updates at a given timestamp.

First we study the effect of grid cardinality. We vary the grid size and compare the algorithms for each of the three queries in Fig. 5. In accordance with previous work that use grid based approach, the performance degrades if the grid size is too small or too large. More specifically, if the grid has too low cardinality, the cost of the queries increases because each cell contains larger number of objects. On the other hand, if the grid cardinality is too high then most of the cells are empty and the cost increases because the number of visited cells becomes large.

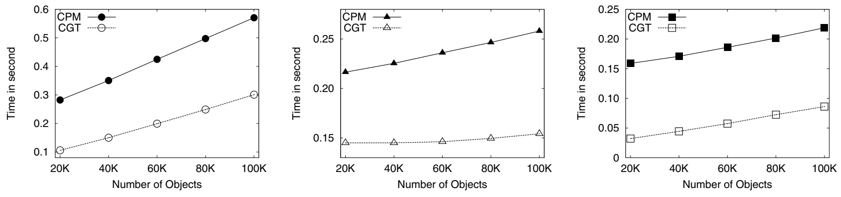
Based on Fig. 5, we choose the default grid sizes for both of the algorithms. More specifically, the default grid size selected for CPM is 64×64 and for CGT is 128×128. In the remaining experiments, we choose these default grid sizes for both of the algorithms.

In Fig. 6, Fig. 7, Fig. 8 and Fig. 9, we study the effect of k, the number of data objects, the number of queries and the agility of the datasets, respectively.



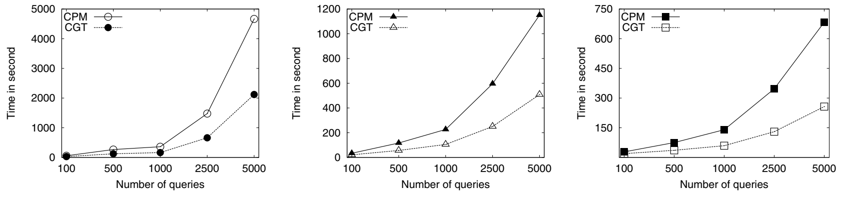
(a) Aggregate *k*NN (b) Furthest *k* Neighbor (c) Constrained *k*NN

Fig. 6. Effect of the value of *k*



(a) Aggregate *k*NN (b) Furthest *k* Neighbor (c) Constrained *k*NN

Fig. 7. Effect of number of objects



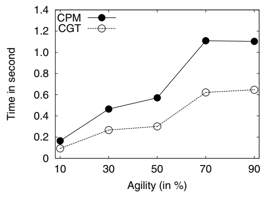
(a) Aggregate *k*NN (b) Furthest *k* Neighbor (c) Constrained *k*NN

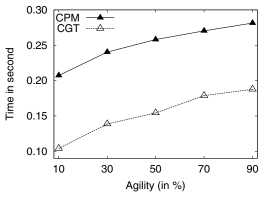
Fig. 8. Effect of number of queries

Although our algorithm is unified and does not require modification for different queries, it outperforms CPM for all different settings.

Since aggregate *k*NN queries has two extra parameters (the number of query instances and the aggregate function), we conduct more experiments to evaluate the performance of our algorithm for these parameters. Fig. 10(a) shows the effect of number of query instances. As expected, the cost of each algorithm increases when the number of query instances is large. This is because the cost of aggregate function increases with the increase in the number of query instances.

Fig. 10(b) studies the effect of different aggregate functions (i.e., Sum, Max and Min). Our algorithm outperforms CPM for each of the aggregate functions. The percentage on top of each group represents the percentage of the time taken by our unified algorithm with respect to CPM.


(a) Aggregate kNN


(b) Furthest k Neighbor

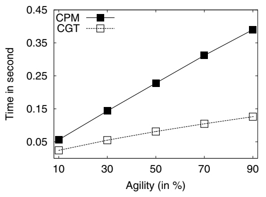
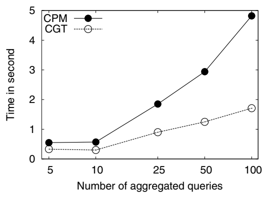

(c) Constrained kNN

Fig. 9. Effect of data agility


(a) # of instances for a query

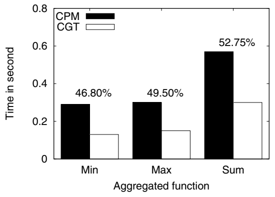

(b) Aggregate function

Fig. 10. Aggregate kNN effect

6 Conclusion

We are first to present a unified algorithm to answer a broad class of spatial queries. Our proposed algorithm is optimal in the sense that it visits minimum number of cells throughout the life of a continuous query. Our extensive experimental results demonstrate that for each inherently different spatial queries our unified algorithm significantly outperforms existing best known algorithm.

Acknowledgments. Xuemin Lin was supported by the ARC Discovery Grants (DP110102937, DP0987557 and DP0881035).

References

1. Bae, S.W., Korman, M., Tokuyama, T.: All farthest neighbors in the presence of highways and obstacles. In: Das, S., Uehara, R. (eds.) WALCOM 2009. LNCS, vol. 5431, pp. 71–82. Springer, Heidelberg (2009)
2. Brinkhoff, T.: A framework for generating network-based moving objects. GeoInformatica 6(2), 153–180 (2002)
3. Cheema, M.A., Brankovic, L., Lin, X., Zhang, W., Wang, W.: Multi-guarded safe zone: An effective technique to monitor moving circular range queries. In: ICDE, pp. 189–200 (2010)
4. Cheema, M.A., Lin, X., Zhang, Y., Wang, W., Zhang, W.: Lazy updates: An efficient technique to continuously monitoring reverse knn. VLDB 2(1), 1138–1149 (2009)

5. Cheema, M.A., Yuan, Y., Lin, X.: CircularTrip: An effective algorithm for continuous *k*NN queries. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 863–869. Springer, Heidelberg (2007)
6. Chen, Z., Ness, J.W.V.: Characterizations of nearest and farthest neighbor algorithms by clustering admissibility conditions. Pattern Recognition 31(10), 1573–1578 (1998)
7. Ferhatosmanoglu, H., Stanoi, I., Agrawal, D.P., Abbadi, A.E.: Constrained nearest neighbor queries. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 257–278. Springer, Heidelberg (2001)
8. Gedik, B., Liu, L.: Mobieyes: Distributed processing of continuously moving queries on moving objects in a mobile system. In: EDBT, pp. 67–87 (2004)
9. Hasan, M., Cheema, M.A., Qu, W., Lin, X.: Efficient algorithms to monitor continuous constrained *k* nearest neighbor queries. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5981, pp. 233–249. Springer, Heidelberg (2010)
10. Henrich, A.: A distance scan algorithm for spatial access structures. In: ACM-GIS, pp. 136–143 (1994)
11. Hjaltason, G.R., Samet, H.: Ranking in spatial databases. In: Egenhofer, M.J., Herring, J.R. (eds.) SSD 1995. LNCS, vol. 951, pp. 83–95. Springer, Heidelberg (1995)
12. Lazaridis, I., Porkaew, K., Mehrotra, S.: Dynamic queries over mobile objects. In: Jensen, C.S., Jeffery, K., Pokorný, J., Šaltenis, S., Hwang, J., Böhm, K., Jarke, M. (eds.) EDBT 2002. LNCS, vol. 2287, pp. 269–286. Springer, Heidelberg (2002)
13. Luo, Y., Chen, H., Furuse, K., Ohbo, N.: Efficient methods in finding aggregate nearest neighbor by projection-based filtering. In: Gervasi, O., Gavrilova, M.L. (eds.) ICCSA 2007, Part III. LNCS, vol. 4707, pp. 821–833. Springer, Heidelberg (2007)
14. Mouratidis, K., Hadjieleftheriou, M., Papadias, D.: Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In: SIGMOD, pp. 634–645 (2005)
15. Papadias, D., Tao, Y., Mouratidis, K., Hui, C.K.: Aggregate nearest neighbor queries in spatial databases. ACM Trans. Database Syst. 30(2), 529–576 (2005)
16. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In: SIGMOD, pp. 71–79 (1995)
17. Suri, S.: Computing geodesic furthest neighbors in simple polygons. J. Comput. Syst. Sci. 39(2), 220–235 (1989)
18. Wu, K.L., Chen, S.K., Yu, P.S.: Incremental processing of continual range queries over moving objects. IEEE Trans. Knowl. Data Eng. 18(11), 1560–1575 (2006)
19. Wu, W., Tan, K.L.: isee: Efficient continuous k-nearest-neighbor monitoring over moving objects. In: SSDBM, p. 36 (2007)
20. Xiong, X., Mokbel, M.F., Aref, W.G.: Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In: ICDE, pp. 643–654 (2005)
21. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate nearest neighbor queries in road networks. IEEE Trans. Knowl. Data Eng. 17(6), 820–833 (2005)
22. Yu, X., Pu, K.Q., Koudas, N.: Monitoring k-nearest neighbor queries over moving objects. In: ICDE, pp. 631–642 (2005)

Real-Time Monitoring of Moving Objects Using Frequently Used Routes

Yutaka Ohsawa[1], Kazuhisa Fujino[1], Htoo Htoo[1], Aye Thida Hlaing[1], and Noboru Sonehara[2]

[1] Graduate School of Science and Engineering, Saitama University
[2] National Institute of Informatics

Abstract. Recently, real-time monitoring of moving objects (MOs), such as cars and humans, has attracted interest. In these systems, tracking trajectories and positions of MOs accurately and for the least communication cost is an important goal. To achieve this, MO position prediction is important. Some studies have proposed real-time monitoring systems that share route information between MOs and a server to predict MO position; however, these systems target public transportation, such as buses, for which the route is always fixed. To the best of the authors' knowledge, a real-time monitoring method for sharing route information between passenger cars and a central server does not exist. This paper proposes such a method using the sharing of frequently used routes (FUR) data between each MO and the server. The server and the MO predict near-future position of MO on the basis of the FUR information using a common algorithm. When the position of the MO deviates from the prediction, it sends a message to the server to maintain position synchronization. This paper evaluates the proposed method using real trajectories of cars and shows that the proposed method outperforms the conventional method, that is, dead-reckoning on a road network.

1 Introduction

Efficient transportation systems for human travel and logistics are required for both environmental and resource-saving reasons. In addition, the diversification of information services required for comfortable and efficient car driving has increased. In particular, the popularity of in-car navigation systems is increasing, as well as pedestrian navigation systems using mobile phones. Using these devices, it is possible to track moving objects and deliver to them various types of location-based services (LBS) .

Car position monitoring is an essential technology for management of taxis, home delivery vehicles, patrol cars, and ambulances . Such moving objects (MOs) can get their positions easily by using GPS (Global Positioning System). If MOs send their position to a server every T_u seconds, the server can monitor the positions of the fleet in real time. However, this method requires very frequent reports to maintain high accuracy of the MO positions. On the other hand, when T_u is large and an MO moves a large distance during T_u seconds, the server cannot know the real trajectory of that MO during this interval. However, if the trajectory information is known, then the state of traffic congestion on the road where the MO is moving can be estimated. Therefore, efficient real-time monitoring of MO position and trajectory is required.

To reduce the overhead of position updating, several shared-prediction-based approaches have been proposed [9]. These approaches share a prediction of MO's near-future position between each MO and the server. The MO sends a message to the server when the MO's real position deviates from the predicted position. When the server receives the message, it modifies the prediction based on the parameters in the message and then continues monitoring.

Over the previous two decades, several methods have been reported for the prediction of MO movement; however, most of these have targeted MOs freely moving in Euclidian space. The most common applications are tracking for cellular phone networks and LBSs, which are not constrained to movement on a road network.

Scheduled vehicles, for example buses, are an important application target of real-time monitoring. In this case, the MO moves strictly along the predetermined route, and therefore the arrival time to bus stops can be easily estimated from the schedule and deviations from the route do not need to be considered. Late and early deviations from the schedule are the only cases that require a location update to be sent to the server.

Dead-reckoning on a road network is the second method for MO position prediction. This method predicts MO near-future position based on the assumption that the MO is moving along a road with a constant speed. When the MO reaches the end of the road or a T-junction, the MO sends the information about the newly selected road.

The third method uses prediction from the trajectory history of the individual MO. For example, a delivery car that goes around to franchise stores can be assumed to follow a similar route every day. By accumulating the routes day by day, the car's position can be predicted using this frequently used routes (FUR) information. The proposed method is based on this observation. This idea is straightforward, and Liu et al. [12] have proposed such a trajectory prediction method based on FUR information. However, their work is limited to future trajectory prediction to serve LBS, and they do not apply this method to prediction for the real-time monitoring of MOs.

To the best of the authors' knowledge, this is the first work to adopt FUR information to real-time monitoring of MO locations. We also propose an adaptive Level of Detail (LoD) setting for monitoring. We show that the proposed method outperforms dead-reckoning on a road network by showing the results from experiments using real movement paths.

The rest of this paper is organized as follows. Section 2 describes the outline of the proposed method. Section 3 shows the MO tracking algorithms by explaining the data structure, messages exchanged between MOs and the server, and the monitoring algorithms on the MO side and the server side. Section 4 describes the experiments and results. Section 5 reviews the related work. Finally, Section 6 summarizes our present work and its future direction.

2 Moving Object Monitoring with FUR

2.1 System Configuration

This section describes the fundamental ideas of the proposed real-time monitoring system. Figure 1 shows the configuration of the system. A MO has a terminal equipped with a GPS receiver, a wireless communication device, a road map, and a computer. This configuration consists of almost the same components as the in-car navigation systems common in Japan, except for the wireless communication device.

The MO acquires its position every 1 second using the GPS, and this position is matched to the map to determine the position on the road network. The MO also predicts its own position. The predicted position is compared with the position matched on the map, and the difference between them is calculated. If the difference exceeds a predetermined threshold value (θ_D), the MO sends a message to the server. If the difference is less than θ_D, the MO does not send any message to the server and continues prediction and map matching.

The server monitors several MOs simultaneously. The monitoring of an MO starts when the server receives the start-of-trip message from the MO, after which the server starts a thread for monitoring the MO. In the thread, the location of the MO is predicted by the same algorithm and the parameters of the MO. Therefore, the prediction is synchronized between the MO and the server. When the server receives a message (usually a position correction) from an MO, the server corrects the parameters for the MO, and then continues the prediction.

In parallel with this monitoring, the server also sends MO locations to the LBS application programs. The sent locations have uncertainties which depend on the threshold θ_D. When a small θ_D is specified, the communication cost is likely to be high. Usually, a large θ_D is specified, for example 1 km. When the application program requests a more precise location, the server sends a

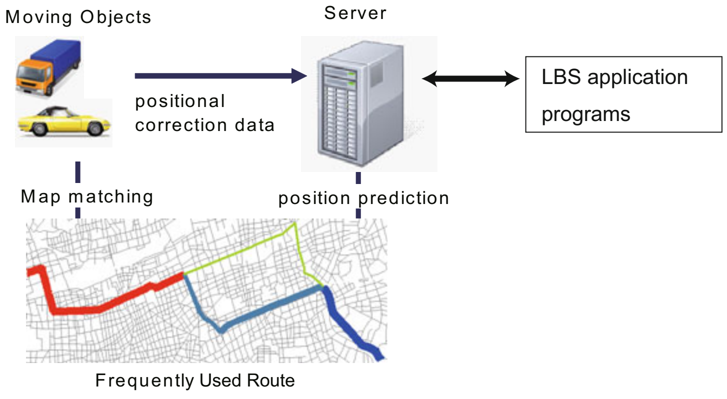


Fig. 1. General concept of the proposed system

message to MOs to decrease θ_D. This operation is called the "LoD (Level of Detail) control". LoD control is necessary to control some location requests that need more precision and also occasionally when communication capacity becomes limited. Usually, high LoD is requested only for a limited number of MOs, for example, for MOs that are inside a special area of the town or neighborhood of a designated position.

For location prediction on a road network, the road segments along which each MO is passing must be determined. This is determined by two methods, one based on FUR and the other based on dead-reckoning on the road network.

2.2 Frequently Used Routes

When one drives a car daily, a trip starts from one location, then ends at another location. For example, one can start the trip from home and go to a shopping center, stay there for some time, and then go to the office. During this trip, one may stop at a gas station for refueling or stop at a convenience store to buy a magazine. Hereafter, we call the start location of a trip or any place at which the MO stays for long periods of time a "Base Point (BP)". In the above example, home, the shopping center, and the office are BPs. Furthermore, places at which one stops for relatively short periods of time are called "Points of Interest (POIs)". In the same example, the gas station and the convenience store are POIs. To summarize, a trip starts at a BP aiming for another BP, with some POIs being visited during the trip.

It can be assumed that there are several target BPs for a trip starting from an origin BP. For example, when a trip starts from home, it can be assumed that the office, the usual shopping centers, and favorite restaurants may be target BPs. In daily life, the number of possible target BPs can be assumed to be limited. In a daily drive, the drivers are likely to take their favorite routes between two BPs, and so the routes can be assumed to be similar. The proposed method is based on this assumption.

As another example, a delivery car going around franchise chain stores would consider the office, the warehouse, and the garage as BPs, and the stores as POIs. On a given day, it can be assumed that the driver takes a similar route connecting the stores in order because the driver knows the most efficient or comfortable roads for driving; however, deviations due to traffic jams, road repair, or a sense of adventure may occur. Hereafter, we call a set of historical routes starting from a BP a FUR.

Figure 2 shows an example of a FUR. The FUR consists of a series of BP to BP paths. Each FUR has a start BP (A in this example) and several destination BPs (B, C, and D). The thickness of roads shows the frequency at which the given route is used. Thus, this figure shows that the car often goes to B from A and only occasionally to C or D.

2.3 FUR Description

A FUR is initially acquired using one of the methods described below. When many historical trajectories of an MO are available, the FUR can be extracted

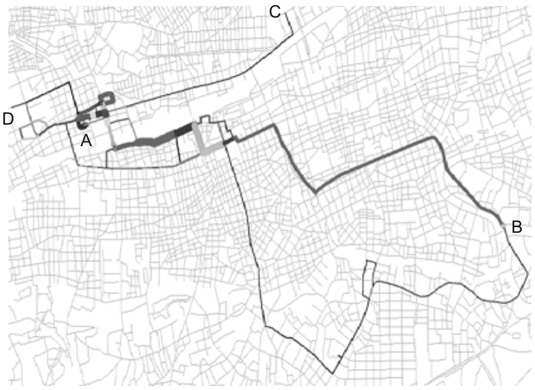


Fig. 2. Example of frequently used routes

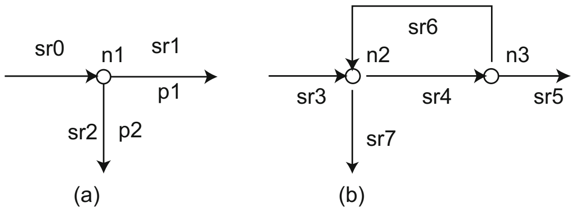


Fig. 3. Construction of a FUR

from this information. When an MO has a recommended route, for example a shortest route or a route that is easy to drive, this can be set as the initial FUR. The most probable situation is incremental acquisition starting from an empty FUR.

Independent of how the routes in the FUR are acquired, an MO can have many different FURs, each with a different starting BP. Once the starting BP is specified, the FUR is determined. Usually, a FUR is not only a tree rooted at the starting BP, but also a network that contains closed loops. This network is a subgraph of the background road network, which consists of a graph whose nodes represent intersections and links represent road segments.

Because the length of each road segment connecting intersections is not long, a route can contain an enormous number of road segments. Therefore, a simple expression is used to describe the formation of the route network. A FUR has branching paths and merging paths. The intersections at which these paths branch or merge are called *control nodes*. The network denoting a FUR consists of BPs and control nodes. A path on the network connecting the control points with each other or between a control point and a BP is called a *subroute*. A subroute consists of several road segments.

Figure 3(a) shows an example of the branches in a FUR network. In this figure, sr0, sr1, and sr2 indicate the subroute links, and p1 and p2 indicate

Table 1. Subroute structure

SR_ID	subroute ID
v	average velocity
SN	start node ID of subroute
EN	end node ID of subroute
f	choosing frequency of SR_ID
n	number of links
Vtx[0]	vertex ID[0]
...	...
Vtx[n–1]	vertex ID[n–1]
Dist[0]	distance from SN to Vtx[0]
...	...
Dist[n–1]	distance from SN to Vtx[n–1]

the probabilities that the subroute will be taken if the MO arrives at node n1 from sr0. At node n1, subroute sr0 is followed by sr1 with probability p1, and turns right to sr2 with probability p2. Figure 3(b) shows a more complicated example. At node n2, the subroute sr3 branches into sr4 and sr7, and the subroute sr4 branches again at node n3 into sr5 and sr6. The subroute reaching node n2 from sr6 always continues to sr7. As this example shows, a subroute should be expressed by a directed graph that includes the probabilities of the next selected subroute for a given trip. Karimi and Liu [10] proposed a method using probability matrices to express the trajectory choice at each intersection. They considered all combinations of the coming links and going links at each intersection. On the other hand, the proposed method can reduce the number of combinations and still obtain an expression of the trajectory choice. On the FUR network, the intersections that need an expression of the trajectory choice are restricted to the FUR network nodes, that is, the control nodes.

The subroutes in the network are expressed in the format shown in Table 1. In this table, SR_ID is the ID assigned to the subroute. SN and EN are the start node ID and the end node ID of the subroute, and f is the frequency with which the subroute is selected. Each subroute is assigned the average velocity (v) of the MO. Vtx[i] is a vertex ID forming the subroute. Dist[i] expresses the distance along the road in meters from the start of the subroute (SN) to Vtx[i]. This value is used for rapid calculation of the distance between the MO's current and expected positions on a FUR.

3 Moving Object Tracking Algorithms

3.1 Moving Object Tracking

As mentioned earlier, the MO and the server share FURs and the algorithm predicting the MO's position. Each MO continuously acquires its current position using GPS measurements. Thus, an attempt is made to match the position of an MO with the current predicted position on the route. When late arrival, early arrival, or deviation from the predicted route is detected, the MO reports it to the server. When the server receives this message from the MO, the server can capture the position of the MO within a predetermined allowance.

Two prediction modes are used in the algorithm. One predicts the position based on a FUR (*On Route (OR) mode*), and the other one is dead-reckoning on the road network (*Dead-Reckoning (DR) mode*). Tracking of MOs primarily begins in the OR mode. Then, when an MO diverges from the FUR, the prediction mode is shifted to the DR mode. The outline of the prediction algorithm is as follows:

(1) Match the object's position with the most frequently used subroute. Find the distance between the current and predicted positions on the route. If this exceeds the threshold value due to late or early arrival, the MO sends its current position to the server for position synchronization.
(2) When the MO diverges to another subroute on a FUR of lower trip frequency, the MO reports to the server the newly selected subroute for synchronization.
(3) When the MO diverges to a road segment that is not on a FUR, both the server and the MO predict the route and the position using dead-reckoning on the road network.

Usually, dead-reckoning predicts the future position under the assumption that the MO continues in the same direction at the same speed. However, under the conditions dealt with in this paper, the locus of MOs is restricted to the road network. Thus, dead-reckoning is restricted to the road network (dead-reckoning on the road network: DRRN). Namely, the MO moves in the same direction along the road until it reaches a dead-end or a T-junction. There, the MO sends information about its change in direction to the server, and then continues with dead-reckoning. Ding and Güting [5] and Čivilis et al. [3] also use a similar prediction method. The following summarizes the method used for dead reckoning.

– An MO's motion is restricted to the road network.
– When an MO encounters an intersection, it selects the road segment going straight ahead.
– When an MO encounters a T-junction, DRRN is suspended until the road link on which the selection of MO can be determined.

It is often the case that a road does not have a unique name. This is especially common in Japan, but rarer in most European and North American countries. Thus, DRRN is defined as above. If each road has a unique name, DRRN can be defined as an MO continuing on the same named road at a constant speed.

As described above, the server and the MO share the FUR, the prediction algorithm, and the parameters. When the distance between the MO's actual and the predicted positions exceeds a threshold value or the MO deviates from the predicted route, the MO sends the current parameters and maintains the synchronization with the prediction.

Table 2. op-codes

code	meaning	parameters
0	start of trip	carID and BPID
1	send position (OR-mode)	carID and position
2	another subroute is selected (OR-mode)	carID, position, and subrouteID
3	enter to dead-reckoning mode	carID,new RoadID, direction, and position
4	send position (DR-mode)	carID and position
5	another road segment is selected (DR-mode)	carID, new RoadID, direction, and position
6	recover to OR-mode	carID, position, and subrouteID
7	end of trip	carID and position

3.2 Moving Object Side Algorithm

Algorithm 1 shows the procedure executed on the MO side for following the above-described method. In this algorithm, the *send* function sends a message to the server. The first parameter of *send* is op-code, the meaning of which is shown in Table 2.

Each car (MO) is assigned an individual ID to identify the car uniquely in the system. *getPosition* in the first line returns the location (p) of the MO captured by GPS, and *getBPID* searches for the BP that is the nearest neighbor of p and returns its ID. $getFUR$ in the third line reads the FUR whose starting BP matches *BP#*. Then, the most frequent subroute (rt) starting from *BP#* is determined from the FUR. Then, the MO sends op-code 0 with the current position to the server to signal the beginning of the trip (Line 4). *mode* in line 5 shows the tracking mode that takes either the value OR (on route) or DR (dead-reckoning). At the beginning of the tracking, the mode is set to OR. As described later, when the server receives this message, the server also starts the monitoring of the MO. Lines 6 to 41 are repeated until the end of the trip.

The position of the MO is updated every 1 second (Line 7). The position is checked to determine whether it is still on the predetermined subroute. $OnRoute(p, rt)$ in line 9 does this check and returns a Boolean value, where the value **true** shows the MO's current position p is located on rt, and **false** shows p has deviated from the subroute rt. When the result is **true**, the distance between p and the predicted position of rt at the current time is calculated by the function $distNow(p, rt)$. Then, if the distance d is greater than a predetermined threshold value (θ_D), the MO informs the server that it is late or early with respect to the expected time of arrival.

Lines 15 to 21 correspond to the situation of the MO deviating from the predicted subroute, usually at an intersection. When p diverges from the currently predicted route, a new predicted route is determined by the function $getSubroute(p)$ (line 15). $getSubroute(p)$ first checks whether p is adjacent to the end node (EN) of the subroute. If it is, other subroutes connected to EN are retrieved, and the subroute (sr) to which p can be best matched p is determined.

When a matching subroute (or any subroute connected at the intersection) does not exist, *getSubroute* returns NULL.

If the search for a branch to which p can be matched on a FUR succeeds, the MO informs the server about the route change (line 17), and the MO's position is matched to this newly selected subroute(rt). On the other hand, when rt is NULL (i.e., the search fails), the tracking mode shifts to DR mode. *getDRRoute* returns a tuple consisting of rt, *newRoad#*, and *dir*, where rt is the path for dead-reckoning composed of the chain of road segments which has the best match with the position p, *newRoad#* is the uniquely assigned road segment ID on which dead-reckoning is started, and *dir* is the direction of movement on the road segment. To inform the server of this mode change, op-code 3 with car#, newRoad#, and position p is sent (line 21).

Lines stating from line 25 of the algorithm correspond to DR mode. In this mode, the MO's position is predicted based on the assumption that the MO continues straight ahead on the road. As with OR mode, whether the MO is still on the predicted route is checked (line 25). When p is on the route (rt), the difference between the MO's current position and the predicted position is calculated. When the difference exceeds the specified value θ_D, the MO sends the real position to the server (line 28). When the server receives this information, it corrects the MO's current position. When the MO diverges from the current route inferred by dead-reckoning, the part of the route from the start of rt to the current intersection is added to the FUR (line 31). This function also returns a Boolean value. **true** indicates that the MO has returned to a known FUR, in which case, the mode is switched to OR mode. On the other hand, when the return value is **false**, DR mode continues. Then, the newly rt is searched (line 36). Next, the MO sends the newly selected road ID, the direction of motion(*dir*), and the current position to the server (line 37). When the MO reaches a T-junction or cannot find a road segment going straight, the MO also sends this format of signal to the server.

When the trip ends, the MO sends OP-code 7, which terminates the monitoring (line 42).

For simplifying the description of the algorithm, the frequency update for selected subroute is omitted. However, every time a new subroute is selected on the FUR, the frequency attached to the subroute is incremented.

3.3 Server Side Algorithm

Monitoring the system on the server side consists of three kinds of modules. One is the *communication module*, which monitors the data sent from each MO. When the server receives an op-code 0 message, this module starts a thread that tracks a new MO, which is a *tracking module*. The communication module distributes the messages from each MO to the corresponding tracking module. The third module is a *location observer* to provide the MOs' position to several types of LBS applications. This module receives the individual MO route and the latest positions, and responds to LBS applications based on their requests, for example, range query or k-NN query. This module also directs the communication module to alter LoD according to the LBS application's requests.

Algorithm 1. Moving Objects

```
1: $p \leftarrow getPosition()$
2: $BP\# \leftarrow getBPID(p)$
3: $rt \leftarrow getFUR(BP\#)$
4: send(0,car#, BP#)
5: $mode \leftarrow$ OR {On Route}
6: repeat
7:    $p \leftarrow getPosition()$
8:    if $mode =$ OR then
9:       if OnRoute(p,rt) then
10:         $d \leftarrow distNow(p, rt)$
11:         if $d > \theta_D$ then
12:            send(1,car#,p)
13:         end if
14:      else
15:         $\{rt, rt\#\} \leftarrow getSubRoute(p)$
16:         if $rt$ is not NULL then
17:            send(2,car#,rt#) {rt# is the selected subrouteID}
18:         else
19:            $mode \leftarrow DR$ {Dead-Reckoning}
20:            $\{rt, newRoad\#, dir\} \leftarrow getDRRoute(p)$
21:            send(3,car#,newRoad#, dir, p)
22:         end if
23:      end if
24:   else
25:      if $OnRoute(p, rt)$ then
26:         $d \leftarrow distNow(p, rt)$
27:         if $d > \theta_D$ then
28:            send(4,car#,p)
29:         end if
30:      else
31:         if $addNewLinkToFUR(rt)$ then
32:            $\{rt, rt\#\} \leftarrow getSubRoute(p)$
33:            $mode \leftarrow OR$
34:            send(6,p,rt#)
35:         else
36:            $\{rt, newRoad\#, dir\} \leftarrow getDRRoute(p)$
37:            send(5,car#,newRoad#, dir, p)
38:         end if
39:      end if
40:   end if
41: until end of trip
42: send(7,car#)
```

Algorithm 2 shows the pseudocode carried out on the tracking module. This module is started with two parameters, car# and BP#. This module retrieves the database, gets FUR for car# starting from BP#, sets the current car position at the BP# position, and selects the initial route that has the highest frequency on the FUR. This tracking module watches the data received from the MO and alters the MO's state according to the received data.

Algorithm 2. Tracking Module

```
Require: car# and BP#
 1: Retrieve the FUR of the car# starting from BP#, then set the result to $tr$.
 2: $cp \leftarrow$ BP#'s position
 3: $mode \leftarrow$ OR {On Route}
 4: repeat
 5:   $\{code, args\} \leftarrow$ receive()
 6:   if $code = 1$ then
 7:     $cp \leftarrow$ reported position by $args$.
 8:   else if $code = 2$ then
 9:     Set $tr$ to the new subroute specified by the SubrouteID in $args$.
10:     $cp \leftarrow$ reported position by $args$.
11:   else if $code = 3$ then
12:     $mode \leftarrow$ DR {Dead-Reckoning}
13:     $tr \leftarrow getDRRoute(args)$
14:     $cp \leftarrow$ reported position by $args$.
15:   else if $code = 4$ then
16:     $cp \leftarrow$ reported position by $args$.
17:   else if $code = 5$ then
18:     $addNewLinkToFUR(rt)$
19:     $tr \leftarrow getDRRoute(args)$
20:     $cp \leftarrow$ reported position by $args$.
21:   else if $code = 6$ then
22:     $mode \leftarrow OR$
23:     $cp \leftarrow$ reported position by $args$.
24:     Set $tr$ to the new subroute specified by SubrouteID in $args$.
25:   end if
26:   send car#, $tr$, $currentTime$, $cp$ to the location observer.
27: until code = 7
```

The individual tracking module in charge of each MO sends a tuple that consists of *tr*, *ct*, and *cp* to the location observer every time when the tracking module receives a message from the MO. *tr* is the route along which the MO will progress, *ct* is the current time, and *cp* is the current position when the update is received. *tr* and *cp* are altered based on the messages from the MO.

The location observer integrates the message sent from the tracking modules and provides the MO locations to LBS applications, depending on their request. This module estimates the current position of MOs periodically (e.g., every 1 minute) based on *tr*, *ct*, and *cp* of each MO.

4 Experimental Results

4.1 Environments

For the experiments, a digitized road map from a 1/25,000 scale base map that covers Saitama City, Japan was used. The trajectory of the MO was captured by a GPS logger (Global Sat, BT-335) placed in the car. The GPS logger recorded longitude, latitude, and time of measurement every 1 second.

Many trajectories were collected using the GPS logger for 1 year. These trajectories are mainly the commuting route from the home of one of the authors to the office (about 10 km). Most of the trips were made from home to the office between 08:00 and 09:30 and from the office back home between 21:00 and 22:30. Figure 2 shows an example of the FUR obtained from the accumulated log by map matching.

4.2 Position Monitoring

The communication cost of the proposed method was compared with that using dead-reckoning on road network (DRRN) for the entire trip to evaluate the efficiency of the proposed method. In the method described in Section 3, MOs send a small amount of data for all eight types of information. It was assumed that one communication requires only one packet, and therefore, the efficiency was evaluated using the number of packets sent.

First, the FUR information described in Section 2.3 with about 100 trajectories was obtained. Next, the MO position monitoring using the other 70 trajectories that were not used to create the FURs were tested. Figure 4 compares the proposed method and DRRN. In the proposed method, the speed obtained from the historical data can be used in OR mode. However, DRRN lacks this information. Thus, the MO's speed was varied from 10 to 30 km/h in increments of 10 km/h.

The horizontal axis in Figure 4 is the permissible positional error (θ_D), and the vertical axis is the average number of communication packets needed to keep tracking under a θ_D value from 100 m to 1 km. As shown in this figure, though the best result was obtained when the speed was set 10 or 20 km/h, the packet number with dead-reckoning is not so sensitive to the MO's speed,

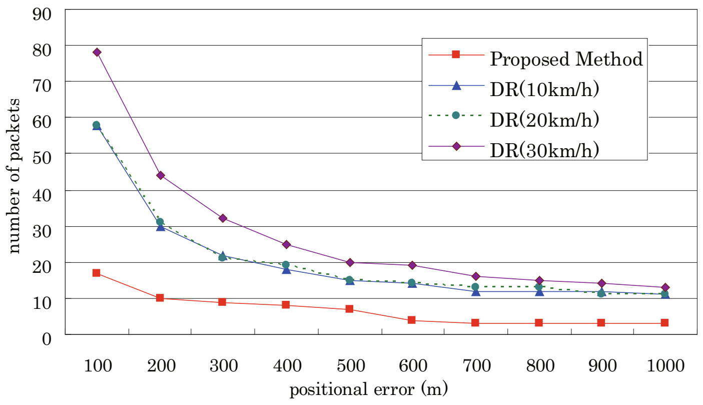


Fig. 4. Comparison of communication cost

especially when the θ_D value is large. This is because most packets in DRRN are only used to report road changes. The proposed method outperforms the conventional method DRRN by factors from 2 to 4. As described in subsection 2.1, the positional update threshold value θ_D usually can be set with a large value, and then the precise location is requested for a limited number of MOs to satisfy the needs of LBS. This control can be performed by a suitable setting of LoD. Whenever this assumption is true, the update cost can be quite low.

5 Related Work

Location tracking of moving objects (MO) has been studied actively for such applications as wildlife animal monitoring, child-care systems, intelligent transport systems, logistics, and fleet management. These studies are roughly categorized into two groups: one targeting MOs that can move freely in the Euclidean space, and the other one targeting MOs for which the trajectories are restricted to a road network.

Although frequent reports improve the accuracy of monitoring MO position, it increases location update cost. To reduce this cost, several policies, including time-based [1], distance-based [14], dead-reckoning [19], and safe range adjustment [22] algorithms, have been proposed. The tracking method described in this paper belongs to the category of MOs restricted to a road network. Therefore, this will be the focus of the rest of this section.

The most of the efficient methods proposed in the literature are based on the prediction of movement of MO on a road network. Wolfson et al. [18],[20] studied this problem under the assumption that objects move along a pre-specified route. They proposed location update policies based on several types of dead-reckoning. In their study, deviations from the pre-specified route are not considered. Tiesyte et al. [16],[17] proposed a monitoring method targeting buses that strictly follow a predetermined route. However, their algorithm also does not allow any divergence from the specified route. Thus, these studies have different objectives than our study.

Ding and Güting [5] studied MO management on a road network. They assumed that objects could move freely on the road network. In their work, they proposed three location update policies: ID-triggered location update, distance-threshold-triggered location update, and speed-threshold-triggered location update. Čivilis et al. [3],[4] proposed a tracking method for MOs based on a client-server architecture. A client (MO) is equipped with a GPS receiver and a communication device. A client predicts its own position, and when the difference between the current and predicted positions exceeds a threshold value, the client sends an update to the server. In their work, three simple prediction policies were proposed. However, these studies do not use knowledge about objects' patterns of motion.

To create an MO monitoring method, a technique for matching a location acquired by a device such as GPS with a position on a road network is essential. This technique is called *map matching*, and has been actively studied in [2], [7], [15], [21]. In our study, we applied the map-matching technique to acquire initial FUR, and to determine deviation from the FUR and the dead-reckoning route.

Spatio-temporal access methods to manage the moment-to-moment changes in an object's position have also been studied actively. Frentzos [6] proposed a spatio-temporal data structure for MOs on a road network. Ding et al. [5] proposed a suitable method for managing object positions on a road network. Usually, a road network is divided into several short segments that connect intersection to intersection. However, to manage the object positions on such short segments is not efficient. Thus, they proposed a data structure named Moving Objects on Dynamic Transportation Networks in which road segments are connected to form a route. The proposed FUR method follows this suggestion.

In addition to acquiring the MO positions, a spatio-temporal query method is also essential for real-time monitoring applications. Ku et al. [11] proposed a k-NN search for MOs. Mouratidis et al. [13] also proposed a k-NN MO search method. Hsueh et al. [8] proposed an MO management method using a location information table. These techniques are necessary to serve LBS based on MO monitoring.

6 Conclusions

This paper proposes a method for the real-time monitoring of MOs on a road network using FURs. Because conventional real-time monitoring does not use knowledge about routes, scalability remains low and the accuracy of tracking is also low. On the other hand, using FURs extracted from historical trajectories decreases the communication cost and achieves highly accurate monitoring.

In this paper, it was assumed that the MOs possess a road map and sufficient computation power. However, the equipment that is required is expensive. Therefore, research into developing appropriate economical terminals, for example, equipped only with a GPS receiver and a communication device but without the road map data inside, is required.

Acknowledgment

This study was supported in part by Grant-in-Aid for Scientific Research (C) (21500093), and by Transdisciplinary Research Integration Center.

References

1. Bar-Noy, A., Kessler, I., Sidi, M.: Mobile users: to update or not to update? Wireless Networks 1(2), 175–185 (1995)
2. Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On map-matching vehicle tracking data. In: Proc. 31st VLDB Conference, pp. 853–864 (2005)
3. Čivilis, A., Jensen, C.S., Nenortaitė, J., Pakalnis, S.: Efficient tracking of moving objects with precision guarantees. Technical Report TR-5, Department of Computer Science, Aalborg University (2004)
4. Čivilis, A., Jensen, C.S., Pakalnis, S.: Techniques for efficient road-network-based tracking of moving objects. IEEE Trans. on Knowledge and Data Engineering 17(5), 698–712 (2005)

5. Ding, Z., Güting, R.H.: Managing moving objects on dynamic transportation networks. In: Proc. 16th SSDBM, pp. 287–296 (2004)
6. Frentzos, E.: Indexing objects moving on fixed networks. In: Hadzilacos, T., Manolopoulos, Y., Roddick, J., Theodoridis, Y. (eds.) SSTD 2003. LNCS, vol. 2750, pp. 289–305. Springer, Heidelberg (2003)
7. Greenfeld, J.S.: Matching GPS observations to locations on a digital map. In: Proc. 81th Annual Meeting of the Transportation Reseach Board (2002)
8. Hsueh, Y.L., Zimmermann, R., Wang, H., Ku, W.S.: Partition-based lazy updates for continuous queries over moving objects. In: Proc. ACM GIS 2007 (2007)
9. Jensen, C.S., Pakalnis, S.: TRAX – real-world tracking of moving objects. In: Proceeding of 33rd VLDB, pp. 1362–1365 (2007)
10. Karimi, H.A., Liu, X.: A predictive location model for location-based services. In: Proc. ACM GIS 2003, pp. 126–133 (2003)
11. Ku, W.S., Zimmermann, R., Wang, H., Wan, C.N.: Adaptive nearest neighbor queries in travel time network. In: Proc. ACM GIS 2005, pp. 210–219 (2005)
12. Liu, X., Karimi, H.A.: Location awareness through trajectory prediction. Computers, Environment and Urban Systems 30, 741–756 (2006)
13. Mouratidis, K., Yiu, M.L., Papadias, D., Mamoulis, N.: Continuous nearest neighbor monitoring in road networks. In: Proc. 32th VLDB, pp. 43–54 (2006)
14. Pitoura, E., Samaras, G.: Locating objects in mobile computing. IEEE Trans. on Knowledge and Data Engineering 13(4), 571–592 (2001)
15. Quddus, M.A., Ochieng, W.Y., Zhao, L., Noland, R.B.: A general map matching algorithm for transport telematics applications. GPS Solutions 7, 157–167 (2003)
16. Tiesyte, D., Jensen, C.S.: Efficient cost-based tracking of scheduled vehicle journeys. In: The 9th MDM, pp. 9–16 (2008)
17. Tiesyte, D., Jensen, C.S.: Similarity-based prediction of travel times for vehicles traveling on known routes. In: Proc. ACM GIS 2008 (2008)
18. Wolfson, O., Chamberlain, S., Dao, S., Jiang, L., Mendez, G.: Cost and imprecision in modeling the position of moving objects. In: Proc. 14th ICDE, pp. 588–596 (1998)
19. Wolfson, O., Sistla, A.P., Chamberlain, S., Yesha, Y.: Updating and queying databases that track mobile units. Distributed and Parallel Databases 7(3), 257–287 (1999)
20. Wolfson, O., Xu, B., Chamberlain, S., Jiang, L.: Moving objects databases: Issues and solutions. In: Proc. 10th SSDBM, pp. 111–122 (1998)
21. Yin, H., Wolfson, O.: A weight-based map matching method in moving objects databases. In: Proc. 16th SSDBM, pp. 437–438 (2004)
22. Zhou, J., Leong, H.V., Lu, Q., Lee, K.C.: Generic adaptive moving object tracking algorithms. In: International Conference on Parallel Processing, pp. 93–100 (2006)

*w*Neighbors: A Method for Finding k Nearest Neighbors in Weighted Regions[⋆]

Chuanwen Li, Yu Gu, Ge Yu, and Fangfang Li

Northeastern University, China
{lichuanwen,guyu,yuge,lifangfang}@ise.neu.edu.cn

Abstract. As the fundamental application of Location Based Service (LBS), k nearest neighbors query has received dramatic attention. In this paper, for the first time, we study how to monitor the weighted k nearest neighbors(WkNN) in a novel weighted space to reflect more complex scenario. Different from traditional kNN approaches, the distances are measured according to a weighted Euclidean metric. The length of a path is defined to be the sum of its weighted subpaths, where a weighted subpath is relative to the weights of its passing regions. By dividing the plane into a set of Combination Regions, a data structure "Weighted Indexing Map"(WIM) is presented. The WIM keeps an index of the weighted length information. Based on WIM, we propose an efficient algorithm, called wNeighbors, for answering the WkNN query. The experimental results show that our WIM based WkNN processing algorithm are effective and efficient.

Keywords: Nearest neighbor query, Weighted Region, kNN, LBS, Weighted Indexing Map.

1 Introduction

Location-based services(LBS) provide accurate position information in a variety of contexts, such as health care, tourism, traffic monitoring, industrial production, etc. There has been an increasing development for LBS currently, involving larger spatial data sets and more efficient query algorithms[19]. Nearest neighbor query and its variant k nearest neighbor(kNN) query are of the fundamental issues in the LBS research area. As an effective method to determine the most important objects of interests, the kNN query is designed to find the top k closest objects to a specified query point, given a set of objects and a distance metric.

Most consideration of the kNN query is focused on the techniques[17,14,2]. Also, the problem of determining the top k nearest neighbors from the source in a plane in the presence of disjoint simple polygonal obstacles is studied[12]. In obstructed scenarios, one can not travel straightly if an obstacle lies on the line to its destination. However, all these space model can not effectively reflect

⋆ This research was partially supported by the National Natural Science Foundation of China under Grant No.60773220, No.61003058 and No.60933001, and the 863 High Technology Foundation of China under Grant No. 2009AA01Z131.

realistic monitoring scenarios, where regions of different features are divided by the road networks.

Consider a hiker(that can be seen as a point) whose objective is to use a given terrain map to find optimal route choices through varied terrain from a source to some types of destinations(inns, snack bars, etc.). The terrain map might look like the diagram in Fig. 1. The meaning of "optimal" here may be considered as the minimal traveling time. When traveling through different terrains, the hiker may take different speeds with respect to the terrain types(e.g., snow, rocks, forests, grass). We segments the terrain into regions according to the covered terrain types and model the regions as polygonal patches. Note that there may be roads between two patches(different or of the same type), on which one can travel faster than inside the surrounding patches. Obviously, based on available methods, effective spatial query cannot be conducted directly. In this paper, we

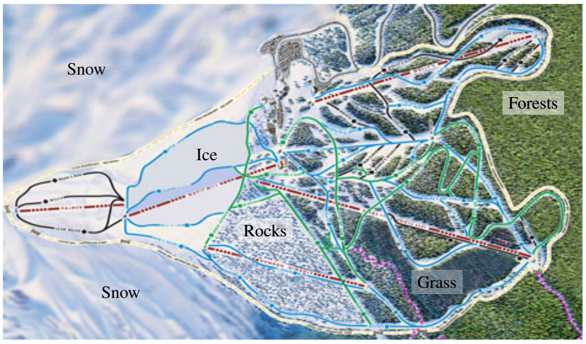


Fig. 1. A map of varied terrain

formalize the novel space model, namely *Weighted Region*, where the plane is subdivided into polygonal regions with different weights by road network. The weight α associated to each region specifies the "cost per unit distance" of traveling in that region. In fact, although weighted region problem has long been studied in GIS community[15], to the best of our knowledge, common spatial queries such as kNN queries still remain uninvestigated at large. Therefore, our main target is to crack the nut of finding the top k nearest neighbors in the weighted regions, we refer as the Weighted k Nearest Neighbors(WkNN) problem. Also, the problem of resolving kNN in obstructed plane can be easily seen as WkNN problem by setting the weight of regions to 1 or $+\infty$ depending on whether the space is free or obstacle, respectively. This type of query can be applied to many scenarios, such as algorithmic motion-planning for robotics, military exercises, navigation monitoring, location-based games, virtual world simulation, intelligent robot control, etc.

In order to effectively deal with the kNN problem in weighted regions, we propose a *Weighted Indexing Map*(WIM) data structure to index the weighted distances in *combination regions*. The combination region is a combination of faces that share the same indexing data. A WkNN searching algorithm, called wNeighbors, is designed based on WIM. In wNeighbors, we adopt the Dijkstra's[3] algorithm to find an upper bound of the top k neighbors. The upper bound can help pruning most irrelevant points before constructing the candidate set. For a query point, the destination points are classified to be *determinate* and *indeterminate*. The determinate points can be checked by WIM directly and the weighted distances of the indeterminate points is calculated by the Mitchell's method[13].

Specifically, the main contribution of this work includes:

- The concept of *combination region* is proposed, which ensures the possibility of indexing the weighted length.
- The Weighted Indexing Map(WIM) data structure is proposed, which can tremendously reduce the real time calculation of the weighted distance.
- Based on the WIM, an efficient searching algorithm, called wNeighbors, is designed to process WkNN queries.
- Extensive experiments are conducted to verify that the proposed algorithms achieve satisfactory performance.

The remainder of this paper is organized as follows: The related work is described in Section 2. In Section 3, we introduce the preliminary of weighted spaces. Section 4 describes the Weighted Indexing Map(WIM) data structure and the wNeighbors method . The experimental studies are presented in Section 5. Finally, the work is concluded in Section 6.

2 Related Work

Common kNN search algorithms mostly consider the Euclidean space. One of the common features is the branch-and-bound strategy on the tree structure, such as the DF-kNN [17] or BF-kNN [9]. The kNN search in road networks are also a consideration of the researching community[10,2]. Another important kNN problem is that of determining the top-k nearest neighbors in the presence of disjoint simple polygonal obstacles. Zhang et al. proposed the *obstructed NN*(ONN) query[18] that can find top-k points in a data set to a specified fixed query point q in an obstructed area. Each query request is considered as a totally new query, which can result in considerable duplicate calculation while the query point moves. Gao et al. considered the continuous visible nearest neighbor queries in obstructed areas. The approach in [4,7] handles the obstructed MkNN in the special scenario when the query point q moves on a given line segment. To deal with the obstacles, a novel concept of *control points* is proposed. The reverse k-nearest neighbor query with consideration of obstacles is proposed in [6] and [5], which employs *half-plane property* and *visibility check* to prune the search space and thus reduces the preprocessing time. In [16], the authors propose an architecture that integrates network and Euclidean information that can be successfully applied to the most popular spatial queries.

Although the kNN problem is not considered, the weighted region problem has been intensively studied in GIS community. The most common approach of finding shortest path between two specified point in weighted region is to lay down a grid on top of the surface[15]. This produces a grid graph of pixels each of which is a small square with sides equal in length of the fineness of the grid. Then costs are assigned to each arcs which connect adjacent pixels. The Dijkstra's algorithm[3] is then used to compute minimum cost paths in the grid graph. The problem of this approach is to require extremely fine grids to capture the content of a simple map, and it creates a digitization bias because of the metrication error imposed by confining movements to limited number of orientations.

3 Preliminary

3.1 Problem Definition

We consider a straight-line planar polygonal subdivision $\mathscr{P}$, specified by a set of faces, edges, and vertices, with each edge occurring between two faces. The information of $\mathscr{P}$ can be stored in data structured like *quad-tree* which allows simple operations. We are given a set of points D specifying particular destinations. Given a special point q indicating the query position, the problem of k nearest neighbors is to find q's top k nearest neighbors from D. The concept of length

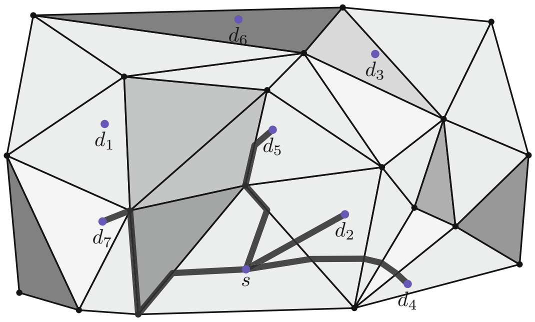


Fig. 2. Shortest paths in weighted regions

in this paper is defined according to a *weighted Euclidean metric*. Each face f has an associated weight $\alpha_f \in [1, +\infty]$, indicating the cost per unit distance of traveling through f. Each edge e, similarly, has a weight $\alpha_e \in [1, +\infty]$ associated with it. The weighted length $d(x, y)$ between two point x and y is simply the product of α_f and the Euclidean distance $|x, y|$, when they are both in face f. Similarly, $d(x, y) = \alpha_e|xy|$ when x and y are both in edge e. The weighted length of a path through the subdivision is then the sum of the weighted lengths of its subpaths through each face and along each edge.

We assume, without loss of generality, that the roads in $\mathscr{P}$ are all along the edges. The weight of an edge is less than that of its surrounding faces if the edge represents a part of a road. Otherwise, we have $\alpha_e = \min\{\alpha_f, \alpha_{f'}\}$ when the edge e is not a road, where faces f and f' are shared by e. This indicates that when one travels on the boundary between f and f', it is as if one is traveling just inside the cheaper of the two faces. Note that $\alpha_e > \min\{\alpha_f, \alpha_{f'}\}$ does not make any sense, since one can travel just along the edge in the cheaper face instead of actually traveling on it.

With our assumption that all faces are triangles(which can be easily achieved by triangulation) and each face has a uniform weight, we are able to define the kNN problem in weighted regions:

Definition 1. *Given a finite triangulation in the plane, assignments of weights α_e and α_f to each edge and each face, a query point s, a destination point set D and a specified value* k. *The Weighted* k *Nearest Neighbor(W*k*NN) query is defined to find a result set R of the top* k *nearest neighbors of s in D, that satisfies*

$$\{d(x,q) < d(x',q) | x \in R, x' \in D - R, |R| = k\}, \tag{1}$$

where $d(\cdot)$ stands for the weighted distance *between two points.*

See Fig. 2 for an example. The weights of each faces is indicated by the gray level of each faces, meaning that it is more expensive to travel through a darker face than through a lighter one. We find the top 4 NNs($\{d_2, d_4, d_5, d_7\}$) for the query point s. The thick lines indicate the shortest paths from s to these points. Note that the shortest paths are not necessarily be a straight line(as in normal Euclidean planes) or line segments around obstacles(as in obstructed planes). The shortest path in weighted regions can be a straight line(as line sd_2), a path that intersects an edge before going along edges and outgoing to the destination(as line sd_5, sd_7), a series of "bending" segments(as line sd_4) or a mixture of these types.

In the rest of this paper, we'll discuss efficient indexing and processing methods for finding the top k nearest neighbors of a query point.

3.2 Basic Concepts

Lemma 1. Snell's Law of Refraction[15]
The nearest weighted path passing through a boundary e between regions f and f' with indices of refraction α_f and $\alpha_{f'}$ obeys the relationship that $\alpha_f \sin\theta = \alpha_{f'} \sin\theta'$, where θ and θ' are the angles of incidence and refraction(respectively).

We define the angle between the incoming ray and the normal to the region boundary be the *angle of incidence*, θ; and the angle between the outgoing ray and the normal be the *angle of refraction*, θ'(see Fig. 3). According to the Snell's Law, the shortest path between point d_1 and d_3 is line segments $d_1 l_2 d_3$.

The angle, $\theta_c(f, f')$, at which $\frac{\alpha_f}{\alpha_{f'}} \sin\theta_c(f, f') = 1$, is called the *critical angle* with regard to faces f and f'. The critical angle can be used whenever we are considering nearest path going from one region f to a less expensive region f', or otherwise. An example is given in Fig. 3, providing that $\alpha_f \geq \alpha_{f'}$, the shortest path from d_1 to d_3 can "bend" at l_2 with the injection angle θ and the reflection angle θ'.

The shortest path that travels through f to strike an edge at the critical angle may travel along ab rather than enter into the interior of f' immediately. Before leaving edge ab(at the critical angle) into f' to the destination point, the shortest path travels along the edge ab for a positive distance. See Fig. 3 for an example, the shortest path from d_1 to d_4 travels along the line segment l_3l_5 before it leaves the edge ab.

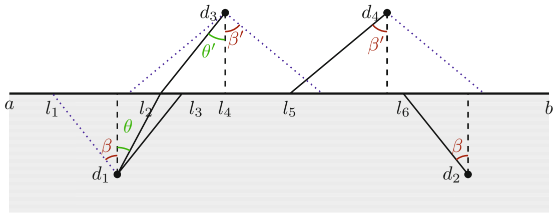


Fig. 3. Examples for Snell's law

Another situation occurs when the source point and the destination point are both near the edge(road), when the path is incident on edge ab at the critical angle θ_c and then travels along ab for some distance. Then the path exits edge ab and goes back into face f before leaving the edge at the critical angle. This kind of path is called *critical reflected* by edge ab. In Fig. 3, the shortest path from d_1 along segments d_1l_3, l_3l_6, l_6d_2 to d_2 is of this kind.

Based on the discussion above, the following lemma can be achieved(which is detailedly discussed in [13]):

Lemma 2. *Assuming that $\alpha_e < min\{\alpha_f, \alpha_{f'}\}$, the shortest path that crosses edge e will do so in one of two ways: either intersect edge e at one point of crossing and obey Snell's Law at that point, or it will hit edge e at the incoming critical angle $\theta_c = \sin^{-1}(\alpha_e, \alpha_f)$, travel along the edge for some distance, and then exit the edge(to the other side) at an outgoing critical angle $\theta_c = \sin^{-1}(\alpha_e, \alpha_{f'})$.*

4 The wNeighbors

In this section, we describe a novel kNN processing scheme, called wNeighbors, which is used in the weighted regions. The design of wNeighbors is motivated by the following observations. First, some of the shortest paths between two points are mostly along the roads, especially when the faces that the two points belong are not adjacent. Second, in particular situations the shortest path to some destination points always travels through one of the vertices of the face that the destination point belongs. Third, although the weighted distance is different from the Euclidean distance, there are relations between these two metrics. The basic idea behind our methodology is to reduce real time calculation as more as possible by indexing essential data in advance.

4.1 Characterization of Shortest Paths

Before introducing the main wNeighbors method, we first infer some essential properties of the shortest paths(weighted). Now consider this situation, when a point p_1 is on one edge e_1 of a face f, whose weight is α_f, and another point p_2 is on another edge e_2 of f, what is the shortest path between p_1 and p_2 like? Lemma 3 gives the answer to this question.

Lemma 3. *Given a triangle face f with its weight be m, two points p_1 and p_2 on different edges(e_1 and e_2 respectively) of f, the intersection angle of e_1 and e_2 be β and the intersection point of e_1 and e_2 be v. We have that*

1. *When $\cos\beta < 1-2/m^2$, the shortest path from p_1 to the point p_2 must travel along the edges of f(either along e_1 to v and then along edge e_2, or along the other edge of f);*
2. *Otherwise, the shortest path between p_1 and p_2 may be the direct line between the two points(not necessary but possible).*

Proof. Let $d(p_1, v)$ be x and $d(p_2, v)$ be y, then the weighted distance of p_1vp_2 and p_1p_2 are $x+y$ and $m \cdot \sqrt{(x \cdot \cos\beta - y)^2 + x^2 \cdot \sin^2\beta}$. Solve the equation

$$x + y - m \cdot \sqrt{(x \cdot \cos\beta - y)^2 + x^2 \cdot \sin^2\beta} < 0, \tag{2}$$

we get that when $\cos\beta < 1 - 2/m^2$, equation 2 has no real number solution, which means that the weighted distance p_1vp_2 is always less than the direct line p_1p_2. □

This lemma tells us that faces which have angles satisfying the condition $\cos\beta < 1 - 2/m^2$ can block the shortest paths that try to cross the adjacent edges of these angles. We call an angle β which satisfies $\cos\beta < 1-2/m^2$ the *block angle*, and a face(recall that all faces are triangles under our assumption) is a *block face* if all three angles are block angles. We call the angle which is not a *block angle* the *combination angle*, meaning that a shortest path can cross the two edges of this angle from one side to another as if the angle connects the two sides.

Now we can make a few observations that will be of use later.

Lemma 4. *A shortest path can never cross a* block face *if the destination point and the source point are not in the face.*

Proof. The proof is trivial and we omit it for lack of space. □

In the wNeighbors method, the whole plane is divided into several *Combination Regions*. The *Combination Region*, *CR* for short, is defined as follows:

Definition 2. *A* Combination Region*(CR), denoted by $\mathscr{R}$, is an area that consists of (1) one block face, or (2) faces that are connected by* combination angle*s.*

Fig. 4 shows the *CR* division of the example in Fig. 2, in which *CR*s are divided by the thick black lines. We can see that some *CR*s consist of several faces, like the one that contains d_6, the one that contains d_5 and the one that contains d_2 and d_4. These faces are connected by combination angles(illustrated by black sectors). The thin dotted lines indicate the original boundaries of the faces.

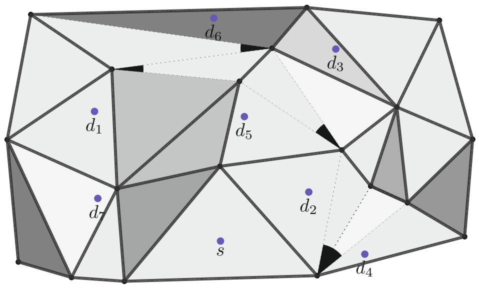


Fig. 4. Connection Regions

Analogous to Lemma 4, we get the following observation.

Lemma 5. *A shortest path can never cross $\mathscr{R}$, if the destination and source points are not in $\mathscr{R}$ or neighbor CRs of $\mathscr{R}$.*

Proof. Assume the claim is false. Then the shortest path between two points s and d crosses $\mathscr{R}$ at two crossing points c_1 and c_2. From the definition of combination angle, we can deduce that one combination region has at least 3 faces, for a combination angle can connect the face at it's left side, it's right side and it's belonging face. Then there are two possibilities for the points c_1 and c_2: (1) they belong to the same face; (2) they belong to different faces. We discuss these two cases:

1. When c_1 and c_2 belong to the same face f, assume the two edges of f are e_1 and e_2(obviously, they can not be on the same edge). We know that the intersection angle of e_1 and e_2 are not combination angle, for that the edges of a combination angle can not be the surrounding edge of a CR(see the definition of the combination region). Therefore, any path that cross e_1 and e_2 cannot be optimal.

2. When c_1 and c_2 belong to different faces, the shortest path must cross one or more edge(s). Assume c_1 and c_2 belong to edges e_1 and e_2, the edges crossed by the shortest path are $\{e_1, e'_1, \ldots, e'_n, e_2\}$. The intersection angle between e_1, e'_1 or e'_n,e_2 can not be a combination angle, for the same reason as the proof above. Consequently, the path cannot be optimal. □

Now we have the following important characterization of shortest paths based on the two lemma above.

Theorem 1. *Given two points s(in $\mathscr{R}_s$) and d(in $\mathscr{R}_d$). If $\mathscr{R}_s$ and $\mathscr{R}_d$ are not the same or adjacent to each other, the shortest path between them must pass at least one vertex v_1 of $\mathscr{R}_s$ and at least one vertex v_2 of $\mathscr{R}_d$. Furthermore, the shortest path between v_1 and v_2 are all along the edges.*

Proof. Assume the claim is false. The shortest path, p, passes the boundary edge of $\mathscr{R}_s$ at point c_1 at edge e_1, which means that in $\mathscr{R}_s$ the incoming part of p is the optimal path between c_1 and s. We know from the definition of critical region that the edges of a critical angle cannot be the boundary edge of a critical region. Consequently, the two angles which has an edge being e_1 cannot be critical angle. Therefore, the path from c_1 cannot be optimal(see Lemma 5), which conflicts with the assumption. $\square$

4.2 Data Structure

In wNeighbors, all the information is stored in the *Weighted Indexing Map* (WIM), a data structure that stores several kinds of indexing data. In WIM, we first keep a list, *crList*, of combination region(CR)s. A CR, *cr*, in *crList* has the following information associated with it: its ID, *crId*; its containing faces, *fList*; its containing destination points, *dList* and its neighbor CRs, *ncrList*. We then write $cr = \{crId, fList, dList, ncrList\}$. We also keep an altered R* tree[1], *pList*, to store the position information of the vertices and destination points with the relation information of their neighbors. A point, p, in *pList* has the following information associated with it: its ID, *pId*; its type, t, indicating it's a vertex or a destination point; its neighbor list, *nList*, which contains the direct neighbor vertices(for a vertex) or the vertices of the face that it belongs(for a destination point); the related destination points(for a vertex), *dList*(along with the shortest distance to these points); the *CR* that it belongs, *cr*. We then write $p = \{pId, t, nList, dList, cr\}$.

The WIM is constructed by a three-step algorithm. In the first step, the plane is split into a set of CRs. Then, for each destination point, the weighted shortest distance to the vertices of its *CR* are calculated. Finally, in the third step, all data points(vertices and destination points) are indexed in the WIM.

In the CR splitting step, we examine each angle of faces to check whether the condition in Lemma 3 is met. The neighbor faces which share edges with a critical angle are connected to create a CR. The splitting process is done after all angles are tested and all CRs are created. The second and third step is trivial and we omit the description for lack of space. We only discuss a lemma that can help calculate the weighted length between a destination point and the vertices of the face that it belongs:

Lemma 6. *Given a point p which belongs to face f, a vertex v of f and two surrounding line segments l_1 and l_2 which intersect at v. The shortest(weighted) path from p to v is (1) the direct line segment between p and v, when all the two intersection angles between line pv and v-intersected borders are greater than $\pi/2 - \theta$, where θ is the critical angle of f, or (2) otherwise, the path along the critical intersection path to the nearer line.*

Proof. Without loss of generality, assume that $\alpha < \beta$. We wish to get the smaller length(weighted) between $l_1 = d(e_1, v_1) + m \cdot d(p_1, e_1)$ and $l_2 = d(e_2, v_1) + m \cdot d(p_1, e_2)$, where $d(x, y)$ means the Euclidean distance between points x and y.

Assume that $d(p_1, v_1) = c$. Then, $l_1 - l_2 = c \cdot ((\sin\alpha - \sin\beta) \cdot \cos\theta / \sin\theta + (\cos\alpha - \cos\beta)) = 2c \cdot \sin(\alpha - \beta)/2(\cos(\alpha + \beta)/2 \cdot \cos\theta / \sin\theta - \sin(\alpha + \beta)/2)$. Note that $\alpha, \beta < \pi/2 - \theta$, then $(\alpha + \beta)/2 < \pi/2 - \theta$. Consequently, we have $\cos(\alpha + \beta)/2 / \sin(\alpha + \beta)/2 > \sin\theta / \cos\theta$, $\sin(\alpha - \beta)/2 < 0$. Therefore, we can conclude that $l_1 - l_2 < 0$, which means, even in weighted regions, the path along the nearer road is shorter. □

4.3 W*k*NN Search Algorithm

By adopting the WIM data structure, an immediate consequence of Theorem 1 is achieved, which forms the basis of our searching algorithm:

Theorem 2. *The shortest path between a query point p_1, which is in $\mathscr{R}_1$, and a destination point p_2, which is in $\mathscr{R}_2$, can be calculated as follows:*

1. *When $\mathscr{R}_1$ and $\mathscr{R}_2$ are disconnected(they do not share any edge), the weighted length of the shortest path between p_1 and p_2 can be calculated by Dijkstra's algorithm[3] using the position information of WIM.*
2. *When $\mathscr{R}_1$ and $\mathscr{R}_2$ are adjacent(they share one or more edges), the weighted length of the shortest path between p_1 and p_2 can be calculated by Mitchell's algorithm[13].*

We call p_2 a determinate *point in the first case and* indeterminate *otherwise.*

Proof. For the first case, when $\mathscr{R}_1$ and $\mathscr{R}_2$ are disconnected, we can deduce from Theorem 1 that the shortest path between p_1 and p_2 are from p_1 to a certain vertex of $\mathscr{R}_1$ and then go along edges till it hits $\mathscr{R}$ at its certain vertex, thereafter go to p_2 inside $\mathscr{R}$ by an optimal path. We know from the definition of WIM that the information of weighted length between vertices and destination points are all stored as a graph, then Dijkstra's algorithm[3] can be adopted here to find the optimal path.

For the second case, the shortest path can not be calculated just from the information in WIM. Then we adopt Mitchell's algorithm[13] to handle this situation. Mitchell's algorithm[13] can find the shortest path between two points in arbitrary faces. □

The main procedure of our wNeighbors method is illustrated in Algorithm 1. Before explaining the main concept of the algorithm, let us discuss several important routines. Routine `Max`(resp., `Min`) returns the upper(resp., lower) bound of the length from query point to the specified node. The upper bound between two points is their distance inferred from WIM, and the lower bound is defined to be the product of their Euclidean distance and the lowest weight of regions along the direct line between them. Note that for determinate nodes the upper bounds and lower bounds are both set as the real distances to the query point. The routine `MaxUpperbound`(R) is a successively comparing procedure, which returns the maximum upper bound of each node in R. Routine `LocalNearest`(s, D) returns the nearest indeterminate destination point. Note that from the indexing data in

Algorithm 1. Weighted k Nearest Neighbor Search(wNeighbors)

```
Input  : query point $s$, destination set $D$, WIM index $WIM$
Output: result set $R$
1  $C \leftarrow D$, $R \leftarrow \Phi$
2  $R \leftarrow$ find top $k$ nearest destination points in $WIM$ using Dijkstra's algorithm[3]
3  $D \leftarrow D - R$;
4  $upperBound \leftarrow$MaxUpperbound($R$)
5  while $n \leftarrow$LocalNearest($s$,$D$) and Min($n$) $<$ $upperBound$ do
6      $D \xleftarrow{\text{add}} n$
7      if Max($n$) $<$ $upperBound$ then
8          $upperBound \leftarrow$MaxUpperbound($R$)
9          foreach $d \in D$ do
10             Remove $d$ if its lower bound is more than the $upperBound$
11 foreach $r \in R$ do
12     $l_r \leftarrow$ Calculate $d(s,r)$ by Mitchell's algorithm[13]
13     if $l_r < upperBound$ then
14         $upperBound \leftarrow l_r$
15         foreach $d \in D$ do
16             Remove $d$ if its lower bound is more than the $upperBound$
17     if The first k points in $R$ are all determinate then
18         Remove other points expect the top $k$ from $R$
19         Break the loop
20 return $R$
```

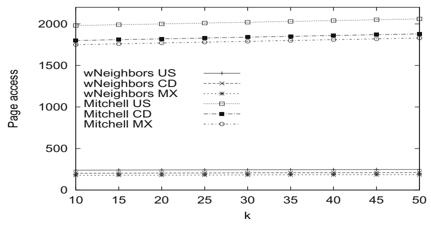
WIM, we can easily get the nearest neighbor(by Euclidean metric) of the query point. We now explain the searching algorithm. The essence of the algorithm is to utilize the *Weighted Indexing Map*(WIM) to directly calculate the distances of determinate points and to refrain as many indeterminate points as possible from the calculation of Mitchell's algorithm. Searching in wNeighbors begins by scanning the WIM, using the Dijkstra's algorithm[3], for the candidate result set R of top k nearest neighbors. Note that some neighbor destination points in R may be indeterminate, for that the destination point and the query point p may be in the same CR. Consequently, the weighed distance from p to these indeterminate points are not real shortest distance to them. However, from these distances we can deduce an upper bound, denoted by *upperBound*, of the top k nearest neighbors, which can help pruning the points whose lower bounds are greater than it. Then the search algorithm only need to check the points that have not been pruned(whose lower bounds are less than *upperBound*). Recall that the position information in WIM is spatially indexed(using the altered R* tree[1]), we can easily found these indeterminate points by the order of their Euclidean distances to the query point. The searching process stops when all the points in the candidate set are checked.

5 Experiments

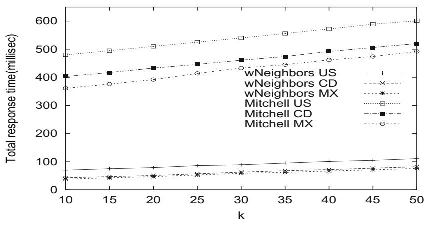
In this section, we present a detailed performance analysis of our solution framework. We first describe our experimental settings, and then discuss the effectiveness of our approach based on the experimental results.

5.1 Experimental Setup

Our evaluation is based on both real and synthetic experiments and is implemented in C++. All experiments are run on a 1.86 GHz Intel Core 2 6300 CPU and 4 GB RAM.

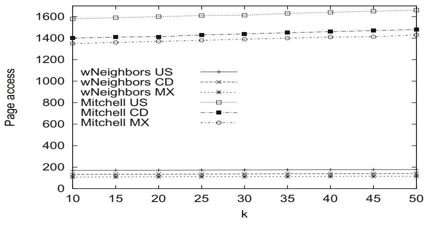


(a) Page access

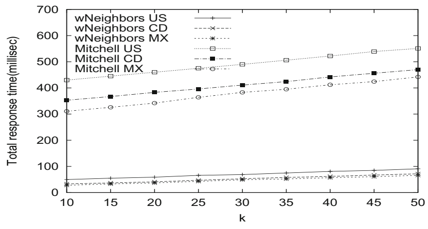


(b) Total response time

Fig. 5. Comparative study, Uniform



(a) Page access



(b) Total response time

Fig. 6. Comparative study, Clustered

The search space is fixed at 2000000×2000000 square shaped ranges and adopts R*-tree[1] as the spatial index. We deploy three real datasets to present the regions and three real datasets to present the destination points[1]. The regions are divided by line segment data of railroads and roads in *US*, *Canada(CD)* and *Mexico(MX)*. The destination points are the point data of popular places and cultural landmarks in *US*, *Canada* and *Mexico*. The line segment datasets contain information of 355,312, 121,416 and 92,392 line segments respectively, and the popular places and cultural landmarks datasets contain 21,223, 7,093

[1] The datasets are available in the R-tree Portal website (`http://www.rtreeportal.org`).

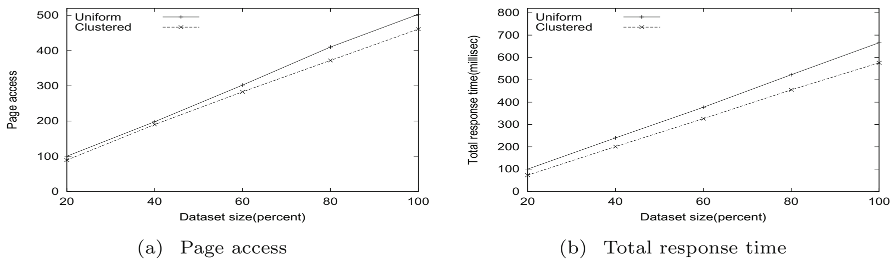
and 5,380 points respectively We normalize the real datasets to fit our search space and triangulate these regions to fit our algorithm. Besides the real regions, the weight of each regions are synthetically generated. The two generated weight datasets are uniformly distributed and clustered distributed. In the Uniform generated datasets, the weight values are uniformly placed along each dimension. For the clustered weight value, the default number of clusters is 400 and the weights are set proportional to the distance to its nearest cluster center. We first compare the *w*Neighbors method and the state-of-art GIS methods and find that the *w*Neighbors method is much better. Then we focus our study on the behavior of *w*Neighbors with various parameters and under different workloads. In our evaluation, we use the number of page access and the total response time as the performance metric. To account for the imprecision that may occur in the synthetic datasets, we run every experiment 10 times and take the average value.

5.2 Comparative Study of *w*Neighbors and Other Approaches

We begin by comparing *w*Neighbors with original Mitchell's method[13]. Since this is the first work, to the best of our knowledge, of the W*k*NN problem, we can only compare our approach to the "*k*NN version" of Mitchell's method, which means the distance between points are calculated by the Mitchell's method and the *k*NN search algorithm adopts the widely used *k*NN search method[11]. As we can see in Fig.5(Uniform) and Fig.6(Clustered), *w*Neighbors has significantly fewer page accesses and shorter total response time than Mitchell's method. The page access and total response time in clustered scenarios are both less than that in the uniform scenarios. This is because that the shortest paths in the clustered area have more chance to be long the roads, which makes the Dijkstra's method available.

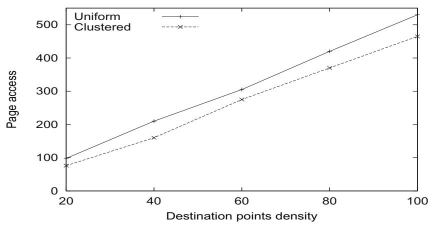
5.3 *w*Neighbors under Different Parameters and Workloads

We further study the performance of *w*Neighbors in this subsection by evaluating the effects of different parameters and different workloads. In the first experiment, we examine the effect of the dataset size. The results of queries

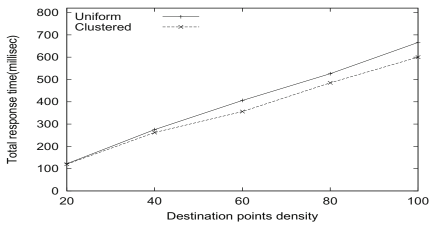


(a) Page access (b) Total response time

Fig. 7. Effects of dataset size



(a) Page access



(b) Total response time

Fig. 8. Effects of destination points density

using 20-100 percents of the real dataset(both destination points and line segments) are shown in Fig. 7. We can see that as the size of dataset increases, both the number of page access and total response time increase. This is expected as more data needs to be examined. The second experiment studies the effect of the density of destination points on the performance on wNeighbors by using 20-100 percents of the real dataset. The results are shown in Fig.8. As expected, as the destination increases, wNeighbors incurs larger number of page access and total response time.

6 Conclusion

In this paper, we consider the weighted kNN search problem. By proposing the concept of combination region, we design an effective indexing data structure, called Weighted Indexing Map(WIM), which can tremendously reduce the calculation cost of the online kNN processing. The WkNN processing algorithm, called wNeighbors, is proposed based on the WIM data structure. Experiments show that the wNeighbors algorithm can handle the WkNN problem effectively and efficiently.

References

1. Beckmann, N., Kriegel, H., Schneider, R., Seeger, B.: The r*-tree: an efficient and robust access method for points and rectangles. ACM SIGMOD Record 19(2), 322–331 (1990)
2. Chen, Z., Shen, H.T., Zhou, X., Yu, J.X.: Monitoring path nearest neighbor in road networks. In: SigMod (2009)
3. Dijkstra, E.: A note on two problems in connexion with graphs. Numerische Mathematik 1(1), 269–271 (1959)
4. Gao, Y., Zheng, B.: Continuous obstructed nearest neighbor queries in spatial databases. In: SIGMOD, pp. 577–590 (2009)
5. Gao, Y., Zheng, B., Chen, G., Lee, W.-C., Lee, K.C., Li, Q.: Visible reverse k-nearest neighbor query processing in spatial databases. TKDE 21(9), 1314–1327 (2009)
6. Gao, Y., Zheng, B., Chen, G., Lee, W.-C., Lee, K.C.K., Li, Q.: Visible reverse k-nearest neighbor queries. In: ICDE, pp. 1203–1206 (2009)

7. Gao, Y., Zheng, B., Lee, W.-C., Chen, G.: Continuous visible nearest neighbor queries. In: EDBT, pp. 144–155 (2009)
8. Guibas, L., Stolfi, J.: Primitives for the manipulation of general subdivisions and the computation of Voronoi. ACM Transactions on Graphics (TOG) 4(2), 123 (1985)
9. Hjaltason, G., Samet, H.: Ranking in spatial databases. In: Advances in Spatial Databases, pp. 83–95. Springer, Heidelberg (1995)
10. Hu, H., Lee, D., Lee, V.: Distance indexing on road networks. In: Proceedings of the 32nd International Conference on Very Large Data Bases, p. 905, VLDB Endowment (2006)
11. Jagadish, H., Ooi, B., Tan, K., Yu, C., Zhang, R.: iDistance: An adaptive B-tree based indexing method for nearest neighbor search. ACM Transactions on Database Systems (TODS 2005) 30(2), 364–397 (2005)
12. Li, C., Gu, Y., Li, F., Chen, M.: Moving k-nearest neighbor query over obstructed regions. In: APWeb, Pusan, Korea (2010)
13. Mitchell, J., Papadimitriou, C.: The weighted region problem: Finding shortest paths through a weighted planar subdivision. Journal of the ACM (JACM) 38(1), 18–73 (1991)
14. Nutanong, S., Zhang, R., Tanin, E., Kulik, L.: The v*diagram: A query dependent approach to moving knn queries. In: VLDB, pp. 1095–1106 (2008)
15. Papadakis, N., Perakis, A.: Deterministic minimal time vessel routing. Operations Research 38(3), 426–438 (1990)
16. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. In: Proceedings of the 29th International Conference on Very Large Data Bases, vol. 29, pp. 802–813, VLDB Endowment (2003)
17. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In: SIGMOD, pp. 71–79 (1995)
18. Zhang, J., Papadias, D., Mouratidis, K., Zhu, M.: Spatial queries in the presence of obstacles. In: Hwang, J., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 366–384. Springer, Heidelberg (2004)
19. Zhang, J., Zhu, M., Papadias, D., Tao, Y., Lee, D.L.: Location-based spatial queries. In: SIGMOD, pp. 443–454 (2003)

Aggregate Farthest-Neighbor Queries over Spatial Data

Yuan Gao, Lidan Shou, Ke Chen, and Gang Chen

College of Computer Science, Zhejiang University, China

Abstract. In this paper, we study a new type of spatial query, namely *aggregate k farthest neighbor* (AkFN) search. Given a data point set P, a query point set Q, an AkFN query returns k points in P with the largest aggregate distances to all points in Q. For instance, it is reasonable to build a new hotel where the aggregate distances to all existing hotels are maximized to reduce competition. Our investigation of AkFN queries focuses on three aggregate functions, namely SUM, MAX and MIN. Assuming that the data set is indexed by R-tree, we propose two algorithms, namely *minimum bounding* (MB) and *best first* (BF), for efficiently solving AkFN queries with all three aggregate functions. The BF algorithm is incremental and IO optimal. Extensive experiments on both synthetic and real data sets confirm the efficiency and effectiveness of our proposed algorithms.

1 Introduction

The *aggregate nearest neighbor* (ANN) search is an important variant of the nearest neighbor query. It takes multiple query points into account, and has been well studied in recent years [1] [2]. Let P be a set of data points in multidimensional space. Given a query point set Q, an ANN query retrieves the point p in P, which has the smallest aggregate distances to all points in Q. Specifically, $\forall p\prime \in P - \{p\}$, the aggregate distance $\text{AGG}_{q \in Q}\ \|p, q\| \leq \text{AGG}_{q \in Q}\ \|p\prime, q\|$[1], where AGG denotes an aggregate distance function. Taking aggregate function of SUM for an example, ANN finds a meeting point that minimizes the sum of distances to all users (query points). Figure 1 shows a simple example, with data set $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$ and $Q = \{q_1, q_2\}$, the $ANN(Q)$ with aggregate function SUM is p_6, as $\|p_6, q_1\| + \|p_6, q_2\| \leq \|p_i, q_1\| + \|p_i, q_2\|, \forall p_i \in P - \{p_6\}$. This is obvious because p_6 is on line segment $\overline{q_1 q_2}$, thus the minimum sum of distances to q_1 and q_2 is $\|q_1, q_2\|$ All the other points have sum of distances greater than $\|q_1, q_2\|$ by the rule of triangle inequality.

While *nearest neighbor* (NN) search [3] [4] is one of the most important query type in spatial database and has been studied extensively, *farthest neighbor* (FN) search has not received adequate attention. However, FN is useful in real life. For example, a user wants to buy a facility that has serviceable range, like transceiver, telescope etc. He(she) can utilize the distance between he(she) and his(her) FN to decide which kind of facility to buy. Moreover, as a counterpart of the NN, the FN indicates a point that is least interesting to the user. Alternatively, FN is the one that can reduce the undesirable influence from a query point mostly. So FN is useful for finding some quiet place away from a noisy factory. As depicted in Figure 1, the FN points of q_1 and q_2 are p_2 and p_3, respectively.

[1] Without loss of generality, $\|p, q\|$ computes the Euclidean distance between points p and q.

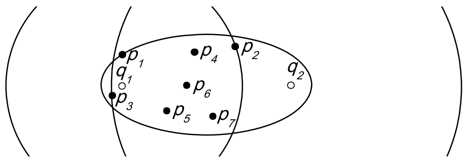


Fig. 1. Example of AkFN queries

To date, the existing work on computing FN only considers single query point and all-pairs farthest neighbor problem [5] [6] [7]. In this paper, we propose a novel type of FN query, called *aggregate k farthest neighbor* (AkFN), which involves multiple query points. Given a data set P and a query set Q, an AkFN query retrieves a set of k points A in P with largest aggregate distances to all points in Q. i.e., $\forall p \in A$, $\forall p\prime \in P - A$, $\text{AGG}_{q\in Q} \; \|p, q\| \geq \text{AGG}_{q\in Q} \; \|p\prime, q\|$. AkFN is useful in real-life applications such as decision support. A possible scenario of an AkFN is building a new hotel in a city. Taking the existing hotels into account, the location of the new hotel is chosen so as to maximize the aggregate distances to all the other hotels for reducing competition most. Another scenario of an AkFN is to find a location for a school keeping away from noisy factories, amusement places, etc. A third scenario is a multi-point monitoring where n places(Q) need to monitor m places(P). The result of MIN-AFN can be used to buy telescopes that make sure all places in P are monitored by at least one place in Q, while MAX-AFN is used to monitor all places in P by all places in Q. As depicted in Figure 1, the $A1FN(Q)$ (or $AFN(Q)$) with aggregate function SUM is p_1, all other points are in the ellipse that has the same total distance to q_1 and q_2 as p_1. It is important to note that p_1 is neither the FN of q_1 nor the FN of q_2, so the SUM-AkFN of a query set may not be in the FN set of any single query point.

To the best of our knowledge, AkFN has not been studied in the literature. The main contributions of this paper are summarized as follows:

– We formally define a new type of query called AkFN, and investigate its characteristics for three aggregate functions, namely SUM, MAX and MIN.
– We propose two algorithms, namely the *minimum bounding* (MB) algorithm and the *best first* (BF) algorithm, for solving AkFN efficiently. The BF algorithm is incremental and proved to be optimal in terms of IO cost.
– Extensive experiments on both synthetic and real data sets are performed to demonstrate the efficiency and effectiveness of our proposed algorithms.

For the following discussion, we consider Euclidean distance and 2D point data sets indexed by R-tree [8], but the proposed techniques are applicable to other spatial indexes such as quadtree [9]. The extension to higher dimensions will be discussed in Section 4.3.

The rest of the article is organized as follows. Section 2 reviews related work on NN queries on R-tree and ANN searches. Section 3 provides preliminaries on AkFN query and some distance metrics used in this paper. Section 4 presents the two algorithms for processing AkFN queries efficiently. Results of our experimental study are reported in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

In this section, we briefly review previous work related to AkFN queries. We shall look at techniques for answering NN queries on R-tree and ANN queries.

2.1 Algorithms for kNN Search on R-tree

The *k nearest neighbor* (kNN) search finds the k points nearest to a given query point. The algorithms for kNN query usually prune nodes based on some metrics in a branch-and-bound manner when traversing the R-tree. The commonly used metrics include MINDIST, MAXDIST and MINMAXDIST.

Existing algorithms for kNN query follow either *depth-first* (DF) or *best-first* (BF) paradigm. DF-kNN [3] algorithm recursively visits R-tree nodes and maintains a candidate set containing the k nearest neighbors found so far. The k^{th} nearest neighbor's distance to the query point is then used as the pruning distance to discard R-tree nodes. The candidate set becomes kNN when all nodes and points have been either visited or pruned. The DF-kNN algorithm is suboptimal as it accesses more nodes than necessary.

The BF-kNN [4] algorithm keeps the nodes visited so far in ascending order of MINDIST in a heap. It then recursively retrieves entries from the head of the heap. If the retrived entry is a node, the child entries of the node are inserted into the list for later exploration. Otherwise, it must be a data point and is inserted into the answer set. The algorithm terminates when the kNN is found. Compared to DF-kNN, BF-kNN is optimal in terms of IO cost. In addition, BF-kNN is an incremental algorithm.

2.2 Aggregate Nearest Neighbor Queries

Papadias et al. introduced *group nearest neighbor* (GNN) query [10] as a novel form of NN search that involves multiple query points. Later, the authors extended GNN to ANN [1] with aggregate functions SUM, MAX and MIN. In the latter work, they proposed three algorithms, named *multiple query method* (MQM), *single point method* (SPM) and *minimum bounding method* (MBM) respectively, for processing ANN queries.

Variants of ANN and GNN queries like ANN queries in network databases [2], the *group visible nearest neighbor* (GVNN) query [11], the *aggregate visible k nearest neighbor* (AVkNN) query [12] and the *probabilistic group nearest neighbor* (PGNN) query [13] have also been studied.

3 Preliminaries

In this section, we present the definitions of AkFN query and some distance metrics. Given a set of data points $P = \{p_1, p_2, ..., p_n\}$, a set of query points $Q = \{q_1, q_2, ..., q_m\}$, the aggregate distance is defined in Definition 1, based on which we formulate the AkFN query in Definition 2.

Definition 1 *(Aggregate Distance (*AGGDIST*)).* Given a data point p, the aggregate distance between p and Q is

$$\text{AGGDIST}(p, Q) = \text{AGG}_{i=1}^{m} \|p, q_i\|,$$

where AGG is an aggregate function such as SUM, MAX and MIN. All three aggregate functions correspond to three distance functions, namely ASUMDIST, AMAXDIST and AMINDIST, respectively. In the sequel, AGG in any definition can be replaced by ASUM, AMAX and AMIN to represent the respective aggregate function.

Definition 2 *(Aggregate k Farthest Neighbor (AkFN) query).* The aggregate k farthest neighbor of Q obtains an answer set A such that: (i) A contains k data points from P (given that the number of points in P is greater than or equal to k); (ii) for any given p in A and $p\prime$ not in A, $p\prime \in P - A$, $\text{AGGDIST}(p, Q)$ is greater than or equal to $\text{AGGDIST}(p\prime, Q)$.

There are three kinds of AkFN, namely, SUM-AkFN, MAX-AkFN and MIN-AkFN based on the selection of the aggregate function.

As widely presented with R-tree, MAXDIST is the maximum distance between a point and a *minimum bounding rectangle* (MBR). In this paper, we will use MAXDIST to measure the maximum distance of two MBRs too. It is formally defined as follows.

Definition 3 *(Maximum Distance (*MAXDIST*)).* (i) Given a point q and an MBR M, the maximum distance between q and M is

$$\text{MAXDIST}(q, M) = \text{MAX}_{p \in M} \|q, p\|,$$

where $\in$ means in the range of an MBR. (ii) Given two MBRs M_1 and M_2, the maximum distance between M_1 and M_2 is

$$\text{MAXDIST}(M_1, M_2) = \text{MAX}_{p \in M_1, q \in M_2} \|p, q\|.$$

The MAXDIST of a point p and an MBR M has a property, which was also used in [4], as described in the following lemma. The formal proof of it is given in Section 4.2.

Lemma 1. The MAXDIST of p and M is the distance between p and the farthest corner of M. Namely, the farthest neighbor of p in M is one of the corners.

It immediately follows Lemma 1 that:

Lemma 2. The MAXDIST of M_1 and M_2 is the maximum of distances between one corner of M_1 and another of M_2, i.e.,

$$\text{MAXDIST}(M_1, M_2) = \text{MAX}_{p \in \text{COR}(M_1), q \in \text{COR}(M_2)} \|p, q\|,$$

where $\text{COR}(M)$ returns the set of corner points of M.

Proof: Suppose the maximum distance between two MBRs M_1 and M_2 is $\|p, q\|$ where $p \in M_1$, $q \in M_2$ and at least one of the points are not at the corners. First we fix the point p, according to Lemma 1, we can find a point $q\prime$ that is one of the corners of M_2 such that $\|p, q\prime\| \geq \|p, q\|$. Then, we fix the point $q\prime$ and can find a point $p\prime$ that is a corner of M_1 such that $\|p\prime, q\prime\| \geq \|p, q\prime\|$. Hence, $\|p\prime, q\prime\| \geq \|p, q\|$. ■

Since the locations of points in an MBR cannot be predicted, it is difficult to know the maximum AGGDIST of a point in the MBR to Q. This maximum AGGDIST is called MAXAGGDIST and is defined as following.

Definition 4 *(Maximum* AGGDIST *(*MAXAGGDIST*)).* Given an MBR M, the maximum AGGDIST of M and Q is

$$\text{MAXAGGDIST}(Q, M) = \text{MAX}_{p \in M} \text{AGGDIST}(p, Q).$$

The point in M that attains MAXAGGDIST is denoted as p^*, namely,

$$p^* = \textsc{ArgMax}_{p \in M} \textsc{AggDist}(p, Q).$$

In particular, $\textsc{MaxAggDist}(Q, p)$ equals to $\textsc{AggDist}(p, Q)$ when an MBR reduces to a point.

Note that MAXAGGDIST is the optimal metric for node access ordering, which, if combined with the *best-first* algorithm, would lead to an IO optimal method. We will investigate this method in Section 4.2.

4 AkFN Query Processing

In this section, we present efficient algorithms for processing AkFN queries, assuming that the data set P is indexed by R-tree and the query set Q fits in memory. We describe the *minimum bounding* (MB) algorithm in Section 4.1 and the *best first* (BF) algorithm in Section 4.2.

4.1 The Minimum Bounding Algorithm

The MB algorithm is motivated by the *minimum bounding method* (MBM) algorithm in [1], which has been shown to have good performance. The main idea of MB is to use the MBR of Q ($MBR(Q)$) and the MBR of some data points $P\prime$ ($MBR(P\prime)$) to compute an upper bound of the maximum aggregate distance between Q and any data point in $MBR(P\prime)$. If this bound is not qualified, i.e., too small, then $MBR(P\prime)$ can safely be pruned.

In order to demonstrate the pruning strategies, we first need to define the aggregate MAXDIST, whose definition is given as follows.

Definition 5 *(Aggregate* MAXDIST *(*AGGMAXDIST*)).* (i) Given an MBR M, the aggregate MAXDIST of Q and M is

$$\textsc{AggMaxDist}(Q, M) = \textsc{Agg}_{i=1}^{m} \textsc{MaxDist}(q_i, M).$$

(ii) Given an MBR M, the aggregate MAXDIST of $MBR(Q)$ and M is

$$\textsc{AggMaxDist}(MBR(Q), M) = \textsc{Agg}_{i=1}^{m} \textsc{MaxDist}(MBR(Q), M),$$

i.e., aggregate of m $\textsc{MaxDist}(MBR(Q), M)$.

Since AGGMAXDIST is easy to compute, Lemma 3 can be used to prune R-tree nodes efficiently.

Lemma 3. Given an MBR M, the following inequality holds,

$$\begin{aligned}&\textsc{AggMaxDist}(MBR(Q), M)\\ &\geq \textsc{AggMaxDist}(Q, M)\\ &\geq \textsc{MaxAggDist}(Q, M).\end{aligned}$$

Proof: For SUM, the following relations can be derived.

$$\begin{aligned}&\textsc{AsumMaxDist}(MBR(Q), M)\\&= \textsc{Sum}_{i=1}^{m}\textsc{MaxDist}(MBR(Q), M)\\&\geq \textsc{Sum}_{i=1}^{m}\textsc{MaxDist}(q_i, M) = \textsc{AsumMaxDist}(Q, M)\\&\geq \textsc{Sum}_{i=1}^{m}\|q_i, p^*\| = \textsc{MaxAsumDist}(Q, M)\end{aligned}$$

The cases for MAX and MIN can be proved similarly but are not presented here to save space. ■

Algorithm 1. MB-AkFN($RTree$, Q, k)

```
1 Initialize an empty set $A$ of answers
2 Call DF-MB-AkFN($RTree.root$, $Q$, $k$, $A$)
3 **return** $A$
```

By the definition of MAXAGGDIST, AkFN query needs to visit a R-tree node (MBR) iff its MAXAGGDIST is larger than the k^{th} result *k_best_dist* found so far. So, if one of the AGGMAXDISTs is smaller than or equal to *k_best_dist*, there is no way for MAXAGGDIST to be larger than *k_best_dist* and the corresponding R-tree node can be pruned. Due to the computation complexity of AGGMAXDIST(Q, M), it is used only if M can not be pruned by AGGMAXDIST$(MBR(Q), M)$.

As a special case of Lemma 3, when an MBR becomes a point p, we have AGGMAXDIST$(p, MBR(Q)) \geq$ AGGDIST(p, Q), where p is also regarded as a set containing a single point. This is used to prune points before checking AGGDIST.

The MB-AkFN algorithm (Algorithm 1) is based on DF-kNN algorithm and applies Lemma 3 to prune nodes. The body of the algorithm just calls another recursive function DF-MB-AkFN (Algorithm 2), which does the actual work. In DF-MB-AkFN, Lines 1 to 7 deal with leaf nodes. For each data point in a leaf node, AGGMAXDIST is calculated for pruning (Line 3) before checking the actual AGGDIST (Line 4). If the AGGDIST of any point is larger than the k^{th} largest distance in A, $A.k\_best\_dist$[2], it is inserted into A (Line 5). Lines 6 and 7 make sure $|A|$ is no more than k. In terms of intermediate nodes (Lines 8 to 18), all the child nodes of E that pass the two-phase filtering in sequence is added to $EntryList$ (Lines 10 to 13). In Line 14, $EntryList$ is sorted in descending order of AGGMAXDIST, i.e., from largest to smallest, to allow early end of iteration through $EntryList$ (Line 18), saving unnecessary nodes processing. DF-MB-AkFN is recursively called on the child nodes that has AGGMAXDIST larger than $A.k\_best\_dist$ (Line 17).

It is worthwhile to mention that MB can also be used with the best first paradigm. Nodes can be ordered by descending order of AGGMAXDIST(Q, M) in a priority queue, as it is a tighter upper bound of MAXAGGDIST(Q, M).

4.2 The Best First Algorithm

As mentioned in Section 3, MAXAGGDIST can be used in a best first algorithm to optimize IO cost. The BF-AkFN algorithm (Algorithm 3) starts by initializing a priority

[2] In case $|A| < k$, $A.k\_best\_dist < 0$.

Algorithm 2. DF-MB-AkFN(E, Q, k, A)

```
1  if $E$ is a leaf node then
2      for each data point $p$ in $E$ do
3          if AGGMAXDIST$(p, MBR(Q)) > A.k\_best\_dist$ then
4              if AGGDIST$(p, Q) > A.k\_best\_dist$ then
5                  Insert $(p, \text{AGGDIST}(p, Q))$ into $A$
6                  if $|A| > k$ then
7                      Remove the nearest entry from $A$
8  else //$E$ is an intermediate node
9      Initialize an empty list $EntryList$
10     for each entry $e$ in $E$ do
11         if AGGMAXDIST$(MBR(Q), e.MBR) > A.k\_best\_dist$ then
12             if AGGMAXDIST$(Q, e.MBR) > A.k\_best\_dist$ then
13                 Insert $e$ into $EntryList$
14     Sort $EntryList$ in descending order of AGGMAXDIST$(Q, M)$
15     for each entry $e$ in $EntryList$ do
16         if AGGMAXDIST$(Q, e.MBR) > A.k\_best\_dist$ then
17             DF-MB-AkFN($e$, $Q$, $k$, $A$)
18         else break
```

queue PQ with the root node of R-tree as the first entry and an empty answer set A (Lines 1 and 2). The while loop (Lines 3 to 10) continues until PQ is empty or A has k entries. Each time an entry E is retrieved from PQ, it is inserted into A (Line 5) if it is a point; otherwise, for each child entry of E, MAXAGGDIST is calculated and then inserted into PQ (Lines 6 to 10).

Unfortunately, computing MAXAGGDIST is not trivial. In the sequel, we will first show that MAXASUMDIST and MAXAMAXDIST can be calculated efficiently by only considering the corners of an MBR. Then we develop another solution for MAXAMINDIST.

Let us consider the region formed by all the points that have AGGDIST less than or equal to a distance. Such region is known for ASUMDIST when $|Q| \leq 2$ – it is enclosed by a circle when $|Q| = 1$ and an ellipse when $|Q| = 2$. As a generalized form, the *aggregate region* (AGGREGION) with arbitrary $|Q|$ is defined as follows.

Definition 6 *(Aggregate Region (*AGGREGION*)).* Given a distance d, the aggregate region AGGREGION(Q, d) is the set of points p such that AGGDIST(p, Q) is less than or equal to d, i.e.,

$$\text{AGGREGION}(Q, d) = \{p : \text{AGGDIST}(p, Q) \leq d\}.$$

In [12] Nutanong et al. showed that ASUMREGION and AMAXREGION are convex. Formally, we have the following lemma.

Lemma 4 *(Lemma 1 in [12]).* The ASUMREGION and AMAXREGION are convex. Particularly, for any two points x and y in the ASUMREGION (AMAXREGION), all points on segment $\overline{xy}$ are in the region too.

Algorithm 3. BF-AkFN($RTree$, Q, k)

```
Initialize a priority queue $PQ$ with $RTree.root$
Initialize an empty set $A$ of answers
while $PQ$ is not empty and $|A| < k$ do
    $E \leftarrow PQ$.pophead()
    if $E$ is a point then Insert $E$ into $A$
    else //$E$ is a node
        for each entry $e$ in $E$ do
            $D \leftarrow$ MaxAggDist$(Q, e.MBR)$
            $NewEntry \leftarrow (e, D)$
            Insert $NewEntry$ into $PQ$
return $A$
```

With Lemma 4, the following claim becomes immediate.

Lemma 5. The maximum aggregate (Sum and Max) distance between a query set and a convex polygon must be the maximum of AggDists of the convex polygon's vertices.

Proof: We prove it by contradiction. Suppose the point p with maximal AggDist is not a vertex of convex polygon. As shown in Figure 2, then p is inside the convex polygon $\overline{p_1p_2p_3p_4p_5p_1}$. Let d = AsumDist(p, Q) (AmaxDist(p, Q)), all the vertices must be strictly in AsumRegion(Q, d) (AmaxRegion(Q, d)). If we draw a line through a vertex and p, it must intersect with an edge. The line extends $\overline{p_3p}$ intersects with $\overline{p_1p_5}$ at a in Figure 2. By Lemma 4, we know a is in AsumRegion(Q, d) (AmaxRegion(Q, d)) and so is p. This leads to $d < d$. ■

Since an MBR is a rectangle, which is a convex polygon, thus we have:

Corollary 1. p^* is in the corner set of an MBR for Sum and Max.

Note that Lemma 1 is just a special case of Corollary 1 where $|Q| = 1$.

For Sum and Max, the MaxAggDist can be calculated by Algorithm 4.

Algorithm 4. MaxAggDist(Q, M) for Sum and Max

```
$Ans \leftarrow 0$
for $c$ in $Cor(M)$ do
    $Ans$ = Max$(Ans$, AggDist$(c, Q))$
return $Ans$
```

For the Min aggregate function, AminRegions do not have the property of convexity. Therefore it is not enough to only take the corners of an MBR into account for MaxAminDist. By definition of MaxAminDist, infinite number of possible points in an MBR need to be considered. Note that even if there are only a few points from P in an MBR, the precise locations are not known in advance knowing the MBR. Hence, it is also impractical to compute MaxAminDist based on the definition because of the infinite number of possible AminDists. Intuitively, the target of MIN-A1FN is to

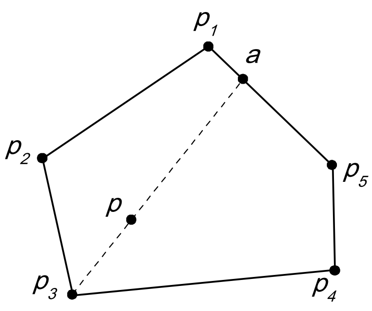


Fig. 2. Proof of Lemma 5

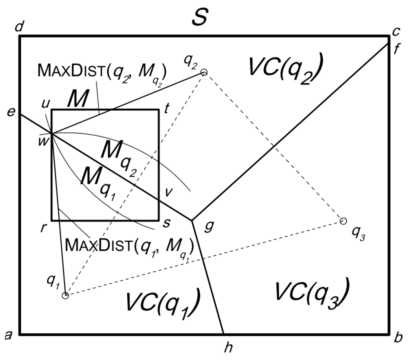


Fig. 3. Example of voronoi diagram

find a point in P, centered at which a circle can spread mostly and does not enclose any point in Q. For a fixed point p, it is the nearest neighbor of p in Q that determines the radius. Thus we need a powerful structure for computing NNs, namely the *Voronoi Diagram* (VD) [14], to help find the target point.

The voronoi diagram is an important technique mainly designed for evaluating nearest neighbor queries efficiently. It partitions the plane to disjoin *voronoi cells* (VCs), such that all the points in the same VC have the same NN. With VD, the task of NN search is reduced to finding which VC the query point is in. The definitions of VD and VC can be formally defined as:

Definition 7 *(voronoi diagram (VD) and voronoi cell (VC)).* For a set of points $Q = \{q_1, q_2, ..., q_m\}$ in space S, the voronoi cell of q_i $VC(q_i)$ is a region in S given by

$$VC(q_i) = \{p : \|p, q_i\| \leq \|p, q_j\|\} for j \neq i,$$

i.e., any point in $VC(q_i)$ has q_i as its NN. The voronoi diagram of Q, $VD(Q)$, is then the collection of all VCs,

$$VD(Q) = \{VC(q_1), VC(q_2), ..., VC(q_m)\}.$$

Figure 3 illustrates a sample voronoi diagram for $Q = \{q_1, q_2, q_3\}$, where S is box $\overline{abcda}$, and polygon $\overline{ahgea}$ is the VC for q_1.

Since VCs are formed by intersecting half planes, they are convex polygons. By definition of VD, The entire space is divided into VCs of Q. Thus, for an MBR M, it must be either fully enclosed by a VC or intersecting with some VCs. For simplicity, the region M intersects with $VC(q_i)$ is denoted as M_{q_i}. M_{q_i} is either empty if M does not intersect with $VC(q_i)$ or a convex polygon for sure. Furthermore, the union of all M_{q_i} is M.

As the example in Figure 3 shows, M is a R-tree node represented by $\overline{rstur}$, which intersects with $VC(q_1)$ and $VC(q_2)$. Thus, M is partitioned into two parts: M_{q_1} and M_{q_2}, corresponding to polygons $\overline{rsvwr}$ and $\overline{wvtuw}$, respectively. By the definition of VC, any points in M_{q_1} will take q_1 as their NNs and any points in M_{q_2} will take q_2 as their NNs, w.r.t Q. This is obvious because $M_{q_1} \subseteq VC(q_1)$ and $M_{q_2} \subseteq VC(q_2)$.

Since our goal is to maximize the distance, it only matters what is the maximum distance between each q_i and M_{q_i}. We denote this maximum distance as MAXDIST (q_i, M_{q_i}) as a generalized MAXDIST in Definition 3. MAXDIST(q_i, M_{q_i}) is easy to compute by Lemma 5 since M_{q_i} is a convex polygon. Particularly, MAXDIST(q_i, M_{q_i}) is 0 if M_{q_i} is empty. Then, MAXAMINDIST is immediately obtained by taking the maximum of all MAXDIST(q_i, M_{q_i}). The algorithm for computing MAXAMINDIST is given as Algorithm 5.

Algorithm 5. MAXAMINDIST(Q, M)

```
1 $Ans \leftarrow 0$
2 Compute $VD(Q)$
3 for $i = 1, ..., m$ do
4     $M_{q_i} \leftarrow M \bigcap VC(q_i)$
5     $Ans = \text{MAX}(Ans, \text{MAXDIST}(q_i, M_{q_i}))$
6 return $Ans$
```

Return to the example in Figure 3, we have MAXDIST$(q_1, M_{q_1}) = \|q_1, w\|$ and MAXDIST$(q_2, M_{q_2}) = \|q_2, w\|$. Since w is on the perpendicular bisector of $\overline{q_1 q_2}$, $\|q_1, w\|$ is actually equal to $\|q_2, w\|$. As for q_3, MAXDIST(q_3, M_{q_3}) is 0 since M does not intersect $VC(q_3)$ at all. Hence, by Lemma 6, MAXAMINDIST(Q, M) is $\|q_1, w\|$ (or $\|q_2, w\|$).

As an optimization for Algorithm 3, we can store the point that gets the MAXAGGDIST in PQ too. With this modification, it first checks whether a child entry contains this optimal point before calculating MAXAGGDIST (Line 8). As a property of BF algorithm, if the optimal point is enclosed by a child node e, e must be the next entry to visit and no distance computation is required. This optimization is especially useful for MIN as MAXAMINDIST is harder to obtain than MAXASUMDIST and MAXAMAXDIST.

Finally, we prove that BF is IO optimal.

Theorem 1. The IO cost of BF algorithm is optimal.

Proof: BF algorithm orders R-tree nodes by MAXAGGDIST, so all the nodes processed by BF is necessary. Thus, the algorithm visits the minimum number of nodes and is IO optimal. ■

4.3 Discussions

The BF algorithm is incremental while the MB algorithm is not. The most desirable advantage of an incremental algorithm is that it can report results progressively. So the AkFN results of BF are produced in descending order by nature. However, MB needs to start from scratch when k changes.

Next, we discuss extensions of our algorithms to higher dimensions. First, MB and BF for both SUM-AkFN and MAX-AkFN can be directly used in higher dimensions since they only use vertices of MBRs for distance computations. Second, the idea of BF for MIN-AkFN can be utilized in high dimensions. However, computing VD and intersections of high dimensional convex polygons is infeasible because of high computation complexity and programming complexity.

5 Experimental Results

In this section, we report the results of our experimental study. The algorithms are implemented in C++ and the experiments are conducted on a Windows XP PC with an Intel 2.26GHz CPU and 4GB memory.

Data sets. Our experiments are based on both synthetic and real data sets. The real data sets consists of nodes from three road networks, namely Califonia (CA), San Francisco (SF) and North America (NA). These data sets are available online[3]. Similar to [15], we normalize each data set into space S = (0,0) $\times$ (1000000, 1000000) and merge them into one larger real data set. The real data set contains 371817 number of points. The synthetic data set is randomly generated based on uniform distribution. The coordinate of each point is generated uniformly along each dimension as we assume that a point's coordinates on different dimensions are mutually independent. All data sets are indexed by disk-based R*-tree [16] with the page size fixed at 4KB. Each R*-tree has the buffer capacity of 5% of its index size.

Table 1. Parameter ranges and default values

Parameter	Range
k	10, **40**, 70, 100
m	20, 40, **60**, 80, 100
n ($\times$ 1000)	10, 50, 100, 500, **1000**, 1500, 2000, 3000, 4000, 5000

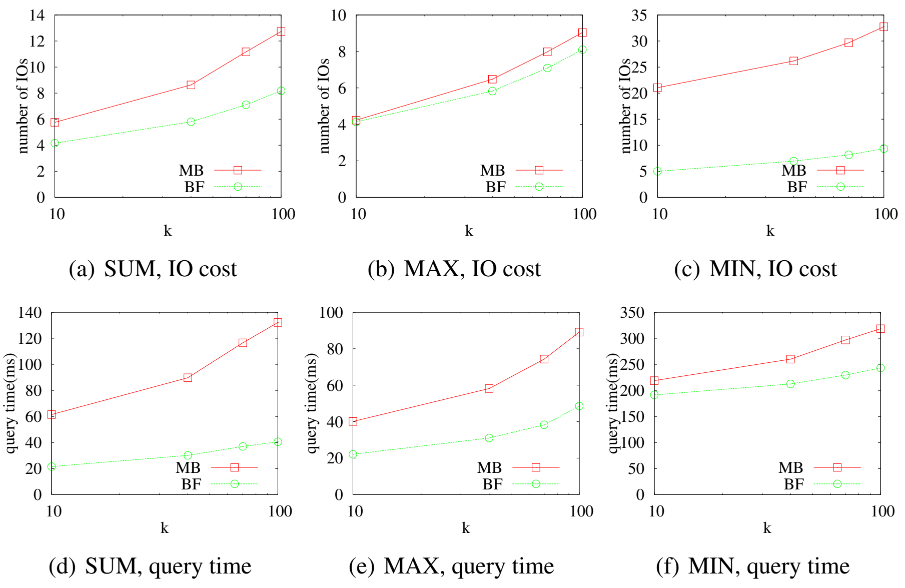
Performance measurement. We use two performance metrics in our study, namely, *IO cost* and *query time*. The query time is the total execution time of an algorithm which includes the IO time and the CPU time. The efficiency and effectiveness of our proposed algorithms are evaluated under various parameters which are summarized in Table 1. The underlined numbers are the default setting for each parameter. In each set of experiment, we only vary one parameter with the other ones fixed at default values to measure its impact on the performance. By default, each experiment runs 100 queries and the average performance is reported. For each query, the m points are uniformly generated in the whole space. Due to space limitation, we only present the performance on the real data sets in 5.1 and 5.2 since the algorithms perform similarly on the synthetic data sets.

5.1 Effect of k

Figure 4 shows the performance of the algorithms as a function of k on the real data sets.

SUM-AkFN. Figure 4(a) shows the IO cost and figure 4(d) shows the query time of SUM-AkFN. As expected, both IO cost and query time of the two algorithms increase as k increases, because a higher value of k implies a larger search space and more distance calculations. Moreover, as k increases, the number of entries in the answer list of

[3] www.cs.fsu.edu/~lifeifei/SpatialDataset.htm



(a) SUM, IO cost (b) MAX, IO cost (c) MIN, IO cost

(d) SUM, query time (e) MAX, query time (f) MIN, query time

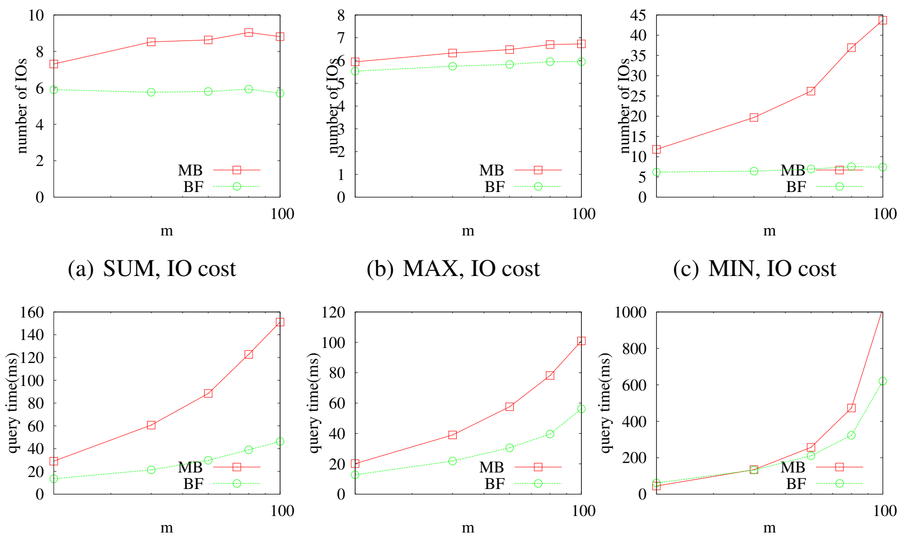
Fig. 4. The effect of k on real data set

MB and the priority queue of BF increase. These changes contribute to more expensive maintenance cost. It is observed that BF outperforms MB under all values of k.

MAX-AkFN. Similar to SUM-AkFN, BF is still better in all cases. However, both IO cost and thus query time of MB are lower than SUM-AkFN, narrowing the gap between MB and BF. We will explain this very shortly.

MIN-AkFN. Compared to SUM-AkFN and MAX-AkFN, the IO cost of BF does not vary significantly while that of MB is much steeper. Though the gap in IO cost increases, the gap in query time is not as obvious. This is because BF needs to compute VD and polygon intersection, which is more complex than MB. Nevertheless, BF is still better in most cases.

Discussions. BF outperforms MB under all circumstance. The IO cost of BF is almost invariant for the three aggregate functions. However, the query time for MIN-AkFN is much longer than the other two due to computation complexity. Both IO cost and query time of MB increase in the order of MAX-AkFN, SUM-AkFN and MIN-AkFN, since it is IO dominated. This also reflects the tightness of AGGMAXDIST. Actually, AMAXMAXDIST equals to MAXAMAXDIST because they both obtain the maximum distance between a corner point of an MBR and a query point. This explains the closeness of the two curves in Figure 4(b). Nevertheless, MB for MAX-AkFN is still not IO optimal due to its algorithm scheme. ASUMMAXDIST is better than AMINMAXDIST since ASUMREGION is convex while AMINREGION is not and AGGMAXDIST uses the corner points of MBRs.



(a) SUM, IO cost (b) MAX, IO cost (c) MIN, IO cost

(d) SUM, query time (e) MAX, query time (f) MIN, query time

Fig. 5. The effect of m on real data set

5.2 Effect of m

Figure 5 illustrates the cost of the algorithms under different m values on the real data sets.

SUM-AkFN. In terms of IO cost, both algorithms remain stable or increase very slowly as m increases, and BF costs less. The query time of both algorithms increase by m. However MB grows more rapidly than BF. This is because MB involves more distance computations than BF.

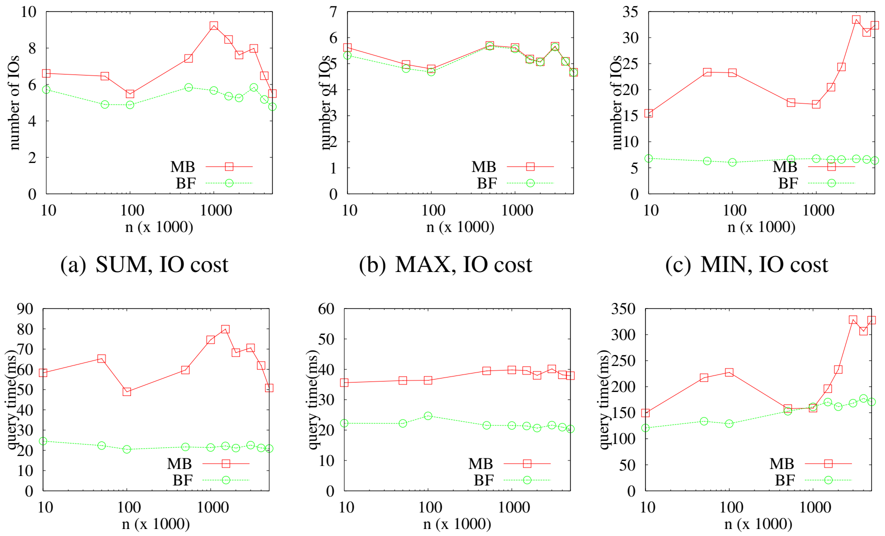
MAX-AkFN. As explained above, MB has very similar IO cost as BF. The growth rates of the query times of the two algorithms are similar to the case in SUM-AkFN.

MIN-AkFN. Unlike SUM-AkFN and MAX-AkFN, the IO cost of MB in MIN-AkFN increases rapidly as m increases. The reason is that a larger value of m makes the VD more complex and AMINMAXDIST bounds less tightly, which leads to more unnecessary nodes access. On the other hand, the performance of BF is stable and costs only slightly more than that of SUM-AkFN and MAX-AkFN. Although the query time of BF is less than MB in most cases, the increase in CPU time may overweight the gap of IO time, which makes BF less efficient than MB (e.g. $m = 20$ in Figure 5(f)).

Discussions. In this set of experiments, BF is still the better choice in common cases.

5.3 Effect of n

We only evaluate this set of experiments on synthetic data set. The performance of the algorithms under various values of n is depicted in Figure 6.



(a) SUM, IO cost (b) MAX, IO cost (c) MIN, IO cost

(d) SUM, query time (e) MAX, query time (f) MIN, query time

Fig. 6. The effect of n on synthetic data set

SUM-AkFN. As the figure shows, BF performs better under both metrics. The value of n does not appear to have considerable impact on the performance. For MB, although the search space grows as n increases, the locality of query results also increases due to larger density of points in the fixed space, which improves the performance. So the performance of MB is unpredictable in the experiments. In contrast, BF is very stable for data sets of different sizes.

MAX-AkFN. It is similar to SUM-AkFN except that MB is significantly better in terms of IO cost and query time.

MIN-AkFN. BF is better than MB in terms of IO cost and only performs worse when $n = 1$ million in terms of query time.

Discussions. The performance of BF remains stable for all three aggregate functions while that of MB sees apparent fluctuation. In summary, BF still outperforms MB in this set of experiments.

6 Conclusions

In this article, we investigated the *aggregate k-farthest-neighbor* (AkFN) problem defined for three aggregate functions: SUM, MAX and MIN. Furthermore, we presented two R-tree based AkFN algorithms, MB and BF. MB follows the *depth-first* scheme and uses AGGMAXDIST for branch ordering. BF is a *best-first* algorithm and thus IO optimal. It maintains entries in a priority-queue by MAXAGGDIST. The experimental results show that BF outperforms MB in terms of IO cost and query time. For future work, we are interested in AkFN query for objects instead of points. We would also consider AkFN in road networks.

Acknowledgment

This work is supported in part by the National Science Foundation of China (NSFC Grant No. 60803003, 60970124).

References

1. Papadias, D., Tao, Y., Mouratidis, K., Hui, C.K.: Aggregate nearest neighbor queries in spatial databases. ACM Trans. Database Syst. 30(2), 529–576 (2005)
2. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate nearest neighbor queries in road networks. IEEE Trans. Knowl. Data Eng. 17(6), 820–833 (2005)
3. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In: SIGMOD Conference, pp. 71–79 (1995)
4. Hjaltason, G.R., Samet, H.: Distance browsing in spatial databases. ACM Trans. Database Syst. 24(2), 265–318 (1999)
5. Aggarwal, A., Kravets, D.: A linear time algorithm for finding all farthest neighbors in a convex polygon. Inf. Process. Lett. 31(1), 17–20 (1989)
6. Suri, S.: Computing geodesic furthest neighbors in simple polygons. J. Comput. Syst. Sci. 39(2), 220–235 (1989)
7. Cheong, O., Shin, C.-S., Vigneron, A.: Computing farthest neighbors on a convex polytope. Theor. Comput. Sci. 296(1), 47–58 (2003)
8. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD Conference, pp. 47–57 (1984)
9. Turkiyyah, G.: Foundations of multidimensional and metric data structures, p. 1024. Morgan Kaufmann, San Francisco (2006); ISBN 978-0-12-369446-1; Computer-Aided Design, vol. 40(4), pp. 518–519 (2008)
10. Papadias, D., Shen, Q., Tao, Y., Mouratidis, K.: Group nearest neighbor queries. In: ICDE, pp. 301–312 (2004)
11. Xu, H., Li, Z., Lu, Y., Deng, K., Zhou, X.: Group visible nearest neighbor queries in spatial databases. In: Chen, L., Tang, C., Yang, J., Gao, Y. (eds.) WAIM 2010. LNCS, vol. 6184, pp. 333–344. Springer, Heidelberg (2010)
12. Nutanong, S., Tanin, E., Zhang, R.: Incremental evaluation of visible nearest neighbor queries. IEEE Trans. Knowl. Data Eng. 22(5), 665–681 (2010)
13. Lian, X., Chen, L.: Probabilistic group nearest neighbor queries in uncertain databases. IEEE Trans. Knowl. Data Eng. 20(6), 809–824 (2008)
14. Aurenhammer, F.: Voronoi diagrams - a survey of a fundamental geometric data structure. ACM Comput. Surv. 23(3), 345–405 (1991)
15. Yao, B., Li, F., Kumar, P.: Reverse furthest neighbors in spatial databases. In: ICDE, pp. 664–675 (2009)
16. 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)

Querying Business Process Models Based on Semantics

Tao Jin[1,2], Jianmin Wang[2], and Lijie Wen[2]

[1] Department of Computer Science and Technology, Tsinghua University, China
[2] School of Software, Tsinghua University, China
{jint05,wenlj00}@mails.thu.edu.cn, jimwang@tsinghua.edu.cn

Abstract. In recent years, the technology of business process management is being more widely used, so that there are more and more business process models (graphs). How to manage such a large number of business process models is challenging, among which the business process model query is a basic function. For example, based on business process model query, the model designer can find the related models and evolve them instead of starting from scratch. It will save a lot of time and is less error-prone. To this end, we propose a language (*BQL*) for users to express their requirements based on semantics. For efficiency, we adopt an efficient method to compute the semantic features of business process models and use indexes to support the query processing. To make our approach more applicable, we consider the semantic similarity between labels. Our approach proposed in this paper is implemented in our system BeehiveZ. Analysis and experiments show that our approach works well.

Keywords: business process model, graph, query, behavior, semantics.

1 Introduction

With the help of business process management technology, enterprises can build or update their process aware information systems [1] quickly. They can adjust their business processes according to the changes from market or policy adjustment from government and so on, and improve their service in time.

The wide use of business process management technology results in a large number of business process models in various industries with different formats. For example, there are more than 3000 models in China Haier, and there are more than 600 EPC models in SAP reference models. China CNR Corporation Limited is a newly regrouped company which has more than 20 subsidiary companies. Before the group company was established, most of these subsidiary companies independently deployed their information systems with a total of more than 200,000 process models. These models describe how business runs in an enterprise or in an industry, who is or are responsible for a specific task, and what data is required or produced. They are invaluable assets for enterprises or industries. How to manage them is challenging, among which the business process model query is a basic function. Based on business process model query, the users can retrieve the related models according to their requirements. For example, before modeling a process, the designer can query the repository and obtain the related models, and then evolve them instead of starting from scratch, it would save a lot of time and is less error-prone.

Business process models can be regarded as a kind of graph. The behavior characteristic is the essential characteristic of business process models. When we query the repository, we always want to obtain some models satisfying the given behavior feature requirements. For example, there would be requirements as follows (or combination of them).

- R_1: find the models in which the task "A" would be executed;
- R_2: find the models in which after the execution of "A", "B" would be executed, i.e. the execution of "A" precedes the execution of "B";
- R_3: find the models in which task "A" can be executed parallelly with "B";
- R_4: find the models in which task "A" cannot be executed in the same instance with "B".

In this paper, the problem to be solved can be described as follows. Given a model repository, retrieve all the models satisfying the user's requirements based on semantics (behavior). Although business process models are always stored as XML format, XQuery cannot be used here because it is not convenient to express the semantics of business process models. Although business process models can be regarded as a kind of graph, the query and indexing research work on graph such as [2,3,4,5,6] cannot be used here because these works only focus on graph structure instead of on semantics.

There are many different notations used to describe the business processes, such as *BPMN*, *XPDL*, *BPEL*, *EPC*, *YAWL*, *PNML* and so on. Because Petri net has good formal foundation and is easy to understand, many research works have been done on the transformation from other notations to Petri nets. A good overview was given in [7]. To deal with business process models in a uniform way, we assume that all the models in the repository are represented as Petri nets or mapped from other formats into Petri nets. In addition, we only focus on control-flow perspective of business process models in this paper, namely, we only focus on what tasks are executed and the order they are executed. We summarize the contributions of our work as follows.

1. To allow the users to describe the behavior features of the models that they want to obtain, we propose a novel, user-friendly text language *BQL* (Behavior Query Language);
2. To compute the behavior features of a business process model efficiently, we adopt unfolding technology[8];
3. To achieve an efficient and scalable performance, we use inverted indexes to support the query processing in the back-end, and all the indexes can be updated incrementally;
4. To tackle the problem of applying different terminologies when modeling processes, we consider the semantic similarity between labels in the query processing;
5. The system is implemented in BeehiveZ.

The rest of this paper is organized as follows. Section 2 introduces the definitions used through this paper, such as Petri net and its semantics. Section 3 describes the architecture of our system and the query language *BQL*. Section 4 describes the index construction and how query is processed. Section 5 shows the system implemented and experiments on both a synthetic dataset and a real dataset. Section 6 compares our work with other works. Section 7 concludes our work and points out the future work.

2 Preliminaries

Since Petri net has a good formalization foundation and is easy to understand, it was introduced into business process management area for modeling, verification and analysis[9].

Definition 1 (Petri net). *A petri net is a triple $N = (P, T, F)$, with P and T as finite disjoint sets of places and transitions $(P \cap T = \emptyset)$, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation)*

We write $X = (P \cup T)$ for all nodes of a Petri net. For a node $x \in X$, $\bullet x = \{y \in X|(y, x) \in F\}$, $x\bullet = \{y \in X|(x, y) \in F\}$. A node $x \in X$ is an input (output) node of a node $y \in X$, iff $x \in \bullet y (x \in y\bullet)$. Let F^* and F^+ denote irreflexive and reflexive transitive closure of F.

Definition 2 (Petri net semantics). *Let $N = (P, T, F)$ be a Petri net.*

- *$M : P \to \mathbb{N}$ is a marking of N, $\mathbb{N}$ is the set of natural numbers. $\mathbb{M}$ denotes all markings of N. $M(p)$ denotes the number of tokens in place p. $[p]$ denotes the marking when place p contains just one token and all the other places contain no tokens.*
- *For any transition $t \in T$ and any marking $M \in \mathbb{M}$, t is enabled in M, denoted by $(N, M)[t\rangle$, iff $\forall p \in \bullet t : M(p) \geq 1$.*
- *Marking M' is reached from M by firing of t, denoted by $(N, M)[t\rangle(N, M')$, meaning that $M' = M - \bullet t + t\bullet$, i.e. one token is taken from each input place of t and one token is added to each output place of t.*
- *A firing sequence of length $n \in \mathbb{N}$ is a function $\sigma : \{0, \ldots, n-1\} \to T$. For $\sigma = \{(0, t_x), \ldots, (n-1, t_y)\}$, we also write $\sigma = t_0, \ldots, t_{n-1}$.*
- *For any two markings $M, M' \in \mathbb{M}$, M' is reachable from M in N, denoted by $M' \in [N, M\rangle$, iff there exists a firing sequence σ leading from M to M'.*
- *A net system is a pair (N, M_0), where N is a net and M_0 is the initial marking of N.*

Definition 3 (Reachability graph). *The reachability graph of a Petri net $N = (P, T, F)$ is a directed graph $RG = (\mathbb{M}, E)$, where $\mathbb{M}$ is the set of all markings of N, and $E \subseteq \mathbb{M} \times \mathbb{M}$. For every $e = (M, M') \in E$, $M, M' \in \mathbb{M}$, it is attached with a transition $t \in T$ such that $(N, M)[t\rangle(N, M')$.*

Although people always use reachability graph to compute the behavior features of business process models, there would be a problem of state explosion, and the efficiency is limited. So we use complete prefix unfolding technique [8] instead.

Definition 4 (Occurrence net). *A Petri net $N = (P, T, F)$ is an occurrence net iff $\forall x, y \in P \cup T : (x, y) \in F^+ \Rightarrow (y, x) \notin F^+$ and $\forall p \in P : |\bullet p| \leq 1$.*

Definition 5 (Ordering relations). *Let $N = (P, T, F)$ be an occurrence net and let $x, y \in X$ be two nodes of N.*

- *x and y are in causal relation, if the net contains a path with at least one arc leading from x to y (x precedes y, denoted as $x- > y$).*
- *x and y are in conflict relation, if the net contains two paths leading to x and y respectively which start at the same place and immediately diverge (x excludes y, denoted as $x\#\#y$).*
- *x and y are in concurrency relation, if x and y are neither in causal relation nor in conflict relation (x can be executed parallelly with y, denoted as $x == y$).*

A Petri net system can be "unfolded" to an occurrence net. The corresponding complete prefix unfolding is more compact than the occurrence net but contains all the information about markings contained in the occurrence net. The complete prefix unfolding is obtained by truncating the occurrence net in points where the information about reachable markings starts to be redundant.

Definition 6 (Complete prefix unfolding)

- *A local configuration $\lceil t \rceil$ is the set of transitions that precede t (t is included).*
- *The final marking of a local configuration $Mark(\lceil t \rceil)$ is the set of places that are marked after all the transitions in $\lceil t \rceil$ fire.*
- *An adequate order $\prec$ is a strict well-founded partial order on local configurations so that $\lceil t \rceil \subset \lceil t' \rceil$ implies $\lceil t \rceil \prec \lceil t' \rceil$.*
- *A transition t is a cutoff transition if there exists another transition t' such that $Mark(\lceil t \rceil) = Mark(\lceil t' \rceil)$ and $\lceil t' \rceil \prec \lceil t \rceil$.*
- *A complete prefix unfolding is the greatest backward closed subnet of an occurrence net containing no transitions after cutoff transitions.*

All the above definitions come from [10] except Definition 3 and Definition 5. For example, Fig. 1 shows a Petri net and its reachability graph and complete prefix unfolding. In Fig. 1(c), E and F are cutoff transitions.

3 System Architecture and Query Language

Fig. 2 shows the system architecture with the following main components:

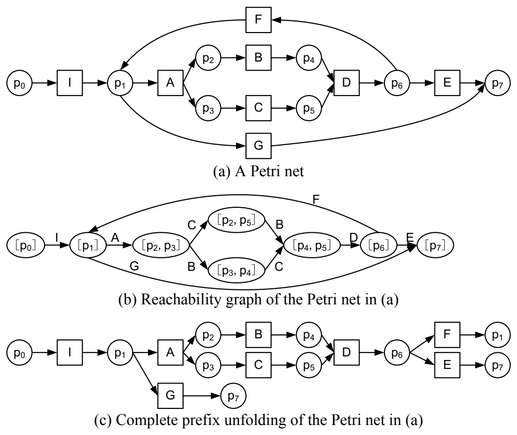

F
I
A
B
C
D
E
G
(a) A Petri net
(b) Reachability graph of the Petri net in (a)
(c) Complete prefix unfolding of the Petri net in (a)

Fig. 1. A Petri net example and its reachability graph and complete prefix unfolding

- `Translator`: translates business process models from variant notations to Petri nets;
- `Repository`: stores the business process model definitions, including the original format and the corresponding Petri nets;
- `Task Relation Indexes`: indexes on the behavior features extracted from the models in the repository;
- `Text Query Editor`: receives the users' text query statements;
- `Semantic Expander`: expands the users' statements using the semantic similar labels in the repository;
- `Query Processor`: processes the query statements and uses the indexes to retrieve the models satisfying the requirements;
- `Model Viewer`: displays the results returned by the query processor.

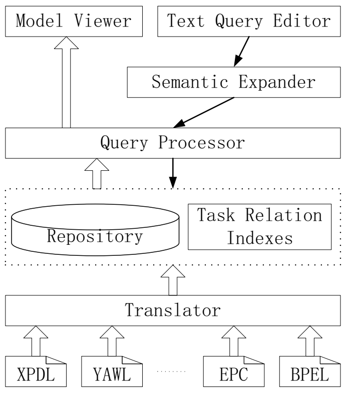


Fig. 2. System architecture

Based on the relations defined in Definition 5, we can describe the behavior features of the models we want to retrieve. For convenience, we define the text language *BQL* here. The BNF of our language can be found as follows. The definition and usage of whitespace is omitted.

- $\langle Expression \rangle$ ::= $\langle AndExpression \rangle$ ($\langle Or \rangle$ $\langle AndExpression \rangle$)*
- $\langle AndExpression \rangle$ ::= ($\langle RelationExpression \rangle$ | $\langle ExistExpression \rangle$ | "(" $\langle Expression \rangle$ ")") ($\langle And \rangle$ ($\langle NotExpression \rangle$ | $\langle RelationExpression \rangle$ | $\langle ExistExpression \rangle$ | "(" $\langle Expression \rangle$ ")"))*
- $\langle NotExpression \rangle$ ::= $\langle Not \rangle$ ($\langle RelationExpression \rangle$ | $\langle ExistExpression \rangle$ | "(" $\langle Expression \rangle$ ")")

- $\langle ExistExpression \rangle$ ::= $\langle Exist \rangle$ ($\langle Activity \rangle$ | "("($\langle Activity \rangle$)+")")
- $\langle RelationExpression \rangle$ ::= $\langle Activity \rangle$ $\langle Relation \rangle$ $\langle Activity \rangle$
- $\langle Exist \rangle$ ::= "exist"
- $\langle Activity \rangle$ ::= " (~["])+ "
- $\langle Relation \rangle$ ::= $\langle ParallelWith \rangle$ | $\langle Exclude \rangle$ | $\langle Precede \rangle$
- $\langle ParallelWith \rangle$ ::= "parallel with" | "=="
- $\langle Exclude \rangle$ ::= "exclude" | "##"
- $\langle Precede \rangle$ ::= "precede" | "– >"
- $\langle And \rangle$ ::= "and" | "&&"
- $\langle Or \rangle$ ::= "or" | "||"
- $\langle Not \rangle$ ::= "not" | "!"

Using *BQL*, we can describe the behavior features of the models we want to retrieve, including the relations between tasks and existence of tasks. The AND-OR-INVERT function is provided. Based on the text language, we can define other graph languages.

For example, *"A"– >"B"* && *"F"– >"D"* && *"B"=="C"* means that we want to retrieve the models where the execution of task "A" precedes task "B", task "F" precedes task "D", and task "B" can be executed parallelly with task "C". It is easy to see that the Petri net in Fig. 1(a) satisfies the requirement.

4 Index Construction and Query Processing

To enhance the efficiency of query, we integrate indexes on the behavior features of the models. Firstly, we extract the behavior features from the models, as described in Section 4.1. Secondly, we build the indexes based on the behavior features and show how query is processed, as described in Section 4.2. At last, how to deal with the label similarity can be found in Section 4.3.

4.1 Behavior Features Extraction

As can be seen in the BNF of *BQL*, there are four categories of behavior features, namely, existences, causal relations, conflict relations and concurrency relations. All these behavior features can be extracted based on the reachability graphs or the complete prefix unfoldings. But there would be a problem of state explosion during the construction of reachability graphs, especially for the Petri nets with too many transitions in parallel. On the contrary, the computation of complete prefix unfoldings is more efficient. The performance comparison can be found in Section 5.1. You can get the algorithm of reachability graph construction from [11], and get both the algorithm of complete prefix unfolding construction and the complexity analysis from [8].

The method based on reachability graph. Based on a reachability graph, we can check whether a task can be executed (existence feature) by traversing all the transitions in the reachability graph. The reason why we do not traverse the original Petri net is that it would be very hard for us to determine whether

Algorithm 1. Compute ordering relations of a reachability graph

```
input  : A Petri net $W = (P_W, T_W, F_W)$, its reachability graph $RG = (\mathbb{M}, E)$,
         and the mapping function $l_W : E \rightarrow T_W$
output: the ordering relation matrix between transitions in $W$

1  foreach $M \in \mathbb{M}$ do
2      foreach $e_i \in$ in-edges of $M$ do
3          foreach $e_o \in$ out-edges of $M$ do
4              $follow[l_w(e_i)][l_w(e_o)] = reach[l_w(e_i)][l_w(e_o)] = true$;
5  for $k$=1 to $|T_W|$ do
6      for $i$=1 to $|T_W|$ do
7          for $j$=1 to $|T_W|$ do
8              if $reach[T_i][T_k]$ && $reach[T_k][T_j]$ then
9                  $reach[T_i][T_j] = true$;
10 for $i$=1 to $|T_W|$ do
11     for $j$=1 to $|T_W|$ do
12         if $follow[T_i][T_j]$ && $follow[T_j][T_i]$ then
13             if $T_i, T_j$ in length one loop or length two loop then
14                 set $(T_i -> T_j), (T_j -> T_i)$ in $ORel$;
15             else
16                 set $(T_i == T_j), (T_j == T_i)$ in $ORel$;
17         else if $!follow[T_i][T_j]$ && $follow[T_j][T_i]$ then
18             set $(T_j -> T_i)$ in $ORel$;
19         else if $follow[T_i][T_j]$ && $!follow[T_j][T_i]$ then
20             set $(T_i -> T_j)$ in $ORel$;
21         else if $!follow[T_i][T_j]$ && $!follow[T_j][T_i]$ then
22             if $reach[T_i][T_j]$ then
23                 set $(T_i -> T_j)$ in $ORel$;
24             if $reach[T_j][T_i]$ then
25                 set $(T_j -> T_i)$ in $ORel$;
26             if $!reach[T_i][T_j]$ && $!reach[T_j][T_i]$ then
27                 set $(T_i \# \# T_j), (T_j \# \# T_i)$ in $ORel$;
28 return $ORel$;
```

a transition can be executed in a Petri net directly. But it is easy for us to determine whether a transition can be executed based on the corresponding reachability graph or the corresponding complete prefix unfolding. Petri nets focus on graph structure, but reachability graphs and complete prefix unfoldings focus on the semantics (behavior). Algorithm 1 can be used to compute the features of causal relations, conflict relations and concurrency relations. This algorithm has a low polynomial time to the size of the reachability graph.

The method based on complete prefix unfolding. Based on the construction of complete prefix unfolding, we can check whether a task can be executed (existence feature) by traversing all the transitions in the complete prefix unfolding. Algorithm 2 (adapted from [10]) computes the ordering relations based

Algorithm 2. Compute ordering relations of a complete prefix unfolding

```
input  : A Petri net $W = (P_W, T_W, F_W)$, its complete prefix unfolding
         $U = (P, T, F)$, and the mapping function $l_W : T \rightarrow T_W$
output: the ordering relation matrix between transitions in $U$
1  foreach $t_i, t_j \in T$ do
2      set $(t_i == t_j)$ in $ORel$;
3  foreach $t_i \in T$ following a preorder traversal of the $U$ do
4      foreach $t_j \in T$ such that $t_j \in \bullet(\bullet t_i)$ do
5          set $(t_j -> t_i)$ in $ORel$;
6          foreach $t_k \in T$ such that $(t_k -> t_j) \in ORel$ do
7              set $(t_k -> t_i)$ in $ORel$;
8          foreach $t_k \in T$ such that $(t_k \#\# t_j) \in ORel$ do
9              set $(t_k \#\# t_i), (t_i \#\# t_k)$ in $ORel$;
10     foreach $t_j \in T$ such that $t_i \neq t_j \wedge \bullet t_i \cap \bullet t_j \neq \emptyset$ do
11         set $(t_i \#\# t_j), (t_j \#\# t_i)$ in $ORel$;
12         foreach $t_k \in T$ such that $(t_j -> t_k) \in ORel$ do
13             set $(t_k \#\# t_i), (t_i \#\# t_k)$ in $ORel$;
   // update ordering relations from cutoff transitions and backwards
14 foreach $t_i, t_j, t_k \in T$ such that $t_i$ is a cutoff transition in $U$, $t_j$ is not a cutoff
   transition, $Mark(\lceil t_i \rceil) = Mark(\lceil t_j \rceil)$, and $l_W(t_i) \bullet \cap l_W(t_j) \bullet \cap \bullet l_W(t_k) \neq \emptyset$ do
15     foreach $t_m, t_n \in T$ such that $(t_k -> t_m) \in ORel \vee t_k = t_m$ and $t_n \in \lceil t_i \rceil$ do
16         set $(t_n -> t_m)$ in $ORel$;
17 return $ORel$;
```

on a complete prefix unfolding. This algorithm has a low polynomial time to the size of the complete prefix unfolding. The relations of causal and conflict can be extracted from the ordering relations directly, but because of cutoff transitions, some concurrency relations are missing. For example, when we compute the relation between B and C in Fig. 1(c), we get $B -> C$ and $C -> B$, because F is a cutoff transition. But in fact, B can be executed parallelly with C. To solve this problem, after we finish Algorithm 2, we find all the pairs (x, y) with $x -> y \wedge y -> x$ and check whether x can be executed parallelly with y by using Algorithm 3, and then we can obtain all the relations of concurrency. In Algorithm 3, we try to find the nearest common ancestor of the given two transitions x and y. If the nearest common ancestor node is type of transition, x and y can be executed parallelly. This algorithm has a linear time to the size of complete prefix unfolding. If we consider the *foata level* information (refer to [8] for details) produced during the construction of complete prefix unfolding, this check can be faster.

Algorithm 3. Check whether two tasks can be executed parallelly

```
input  : A complete prefix unfolding $U = (P, T, F)$ and two transitions $x$ and $y$
         with $x- > y \wedge y- > x$
output: true or false
// mark all the ancestor nodes of $x$
1  add $x$ into queue;
2  while queue is not empty do
3      remove one node $n$ from queue and mark $n$ with $x$;
4      foreach $pre \in \bullet n$ not visited from $x$ do
5          add $pre$ into queue;
// mark all the ancestor nodes of $y$, if some node is marked by $x$
   before and this node is type of transition, return true
6  add $y$ into queue;
7  while queue is not empty do
8      remove one node $n$ from queue;
9      if $n$ is marked from $x$ then
10         if $n$ is type of transition then
11             return true;
12         else
13             return false;
14     else
15         mark $n$ with $y$;
16         foreach $pre \in \bullet n$ not visited from $y$ do
17             add $pre$ into queue;
18 return false;
```

4.2 Index Building and Query Processing

Based on the behavior features extracted in Section 4.1, we can build the inverted indexes. In total, there are four inverted indexes, indexing on “existence”, the relations of “causal”, “concurrency”, and “conflict” respectively. Each inverted index stores a mapping from a kind of behavior features to the models where these features exists.

When a new model is added to the repository, all the behavior features are extracted first and then inserted into the corresponding inverted indexes. It is easy to see that our indexes can be updated incrementally. When a model is deleted from the repository, we first extract all the behavior features from that model, and then delete the mappings between these features and that model.

Given a query statement, after it is parsed we can get some basic queries (with only one item such as $A- > B$), and then we can query on the inverted indexes. For every basic query, we can get a set of models containing the corresponding feature. Finally, we will apply the given AND-OR-INVERT function on these sets to obtain the result set.

4.3 Dealing with Label Similarity

In real life, different labels may be used for the same task. To make our approach more applicable, we introduce the similarity between labels. Two tasks with the label similarity greater than or equal to a specific threshold are regarded as the same task. In our approach, the introduction of similarity between labels should satisfy the following requirements:

1. It should be up to the users to decide whether to enable the similarity between labels during the query time;
2. The similarity threshold should be able to be configured by the users during the query time;
3. The efficiency of query with label similarity should be as high as possible.

Accordingly, (i) we leave the construction of feature indexes unchanged, so when the users enable or disable the label similarity or change the threshold, the feature indexes will not be reconstructed; (ii) we extend the query expression using the similar labels when the query is processed, so that the users can change the threshold during the query time; (iii) we construct a label index to speed up the retrieval of similar labels, and the label index is also not affected by the similarity threshold and can also be updated incrementally.

Label Similarity. Let $W(l)$ be the number of words can be extracted from the label string l. Let $SCW(l_1, l_2)$ be the number of words in the label l_1 whose synonymous can be found in the words of the label l_2. The similarity between two labels can be calculated using Equation 1, which is similar to Dices Coefficient[12] except that the synonymous are considered now.

$$labelSim(l_1, l_2) = \frac{2 \times SCW(l_1, l_2)}{W(l_1) + W(l_2)}. \tag{1}$$

Note that before this equation is used, all the words must be extracted from the labels, and converted into lower case. Stop words must be removed and the remaining words are replaced with their stem. Of course, we can replace the Equation 1 with other term based similarity measures.

Query Expansion. When the similarity between labels is considered, every basic query should be expanded with its similar ones. For example, for a basic query like $A- > B$, if A' is similar to A and B' is similar to B, the original basic query will be expanded with $A'- > B$, $A'- > B'$ and $A- > B'$ so that there would be four basic queries now.

Label Index. To speed up the retrieval of similar labels according to the specific threshold, we construct an index on the labels. It is an inverted index, which stores the mapping from words to labels where these words appear. When this index is built, we extract all the words from the labels, lowercase these words, remove the stop words from these words and replace every remaining word with its stem. When we want to retrieve the similar labels according to the given label and the specific similarity threshold, firstly, we tokenize the given label in the same way as the index is built; secondly, we expand the obtained words with their synonyms; thirdly, we retrieve on the label index using these words to get the candidate labels where at least one of these words appears; finally, we calculate the similarity between the given label and every candidate label and return similar labels with the similarity greater than or equal to the threshold.

5 System Implementation and Experiments

To evaluate our approach, we implemented it in a system named BeehiveZ[1]. BeehiveZ was developed in Java, and all the models were stored as `Text` in MySQL RDBMS. We used the ProM[13] to display the Petri nets. All the inverted indexes proposed in this paper were managed by Lucene. To retrieve synonyms quickly, we mapped WordNet synonyms into memory, which consumed approximately 10MB. Whether to enable or disable the use of similar labels and the similarity threshold both can be configured in BeehiveZ. During our experiments, we used a computer with Intel(R) Core(TM)2 Duo CPU E8400 @3.00GHz and 3GB memory. This computer ran Windows XP Professional SP3 and JDK6. The heap memory for JVM was configured as 1GB.

5.1 Behavior Features Extraction Performance Comparison

We extract behavior features from business process models based on the complete prefix unfoldings in this paper. Compared to the extraction based on the reachability graphs, the performance is improved greatly, especially when there are many tasks in parallel as shown in Fig. 3. In this situation, the construction of reachability graph has an exponential time to the number of tasks, while the construction of complete prefix unfolding has a linear time to the number of tasks. The dominating factor for behavior features extraction complexity is the construction of reachability graph or complete prefix unfolding, so the method based on complete prefix unfolding is more efficient. The performance comparison can be found in Table 1. In this table, "n" means how many tasks are in parallel, "CP" means the extraction based on complete prefix unfoldings, and "RG" means the extraction based on reachability graphs. "-" means it is not computable because of the error of out-of-memory. We can see that the method based on complete prefix unfolding works better.

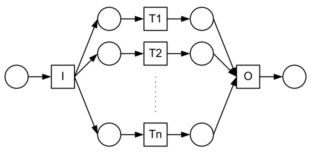


Fig. 3. A Petri net example with many tasks in parallel

5.2 Experiments on Synthetic Dataset

In this section, we conduct experiments on a synthetic dataset to show the performance of query processing. All the models (Petri nets) in our experiments

[1] BeehiveZ can be downloaded from `http://sourceforge.net/projects/beehivez/`

Table 1. Performance comparison on behavior features extraction

n	3	7	11	15	19	38	76
CP	3ms	3ms	3ms	3ms	3ms	13ms	44ms
RG	3ms	231ms	2919850ms	-	-	-	-

were generated automatically by using the rules in [11]. Because all the labels were strings generated automatically, we disabled the use of label similarity. There were more than 300,000 models in the repository. The number of models with different number of transitions followed the normal distribution. The number of transitions in one model ranged from 1 to 50, the number of places in one model ranged from 2 to 31, and the number of arcs in one model ranged from 2 to 102[2]. In total, there were 7,802,438 transitions in the repository, and the number of transitions with different labels was 242,234. During the experiments, we recorded the index construction time, query time, and index storage size. The analysis can be found as follows.

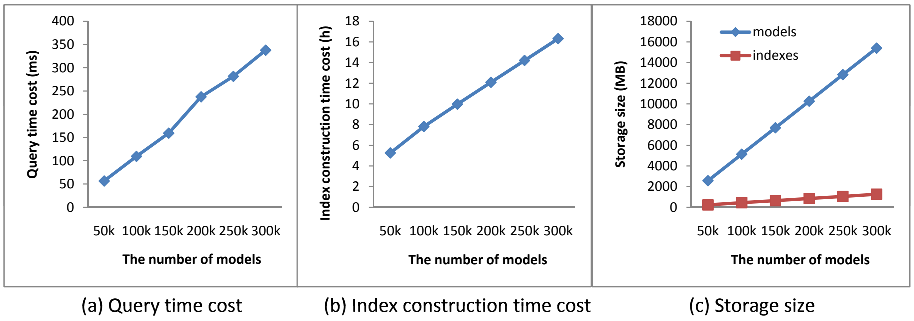


Fig. 4. Performance of our approach

Since the performances of different queries are similar, we use the query "06"$-\!>$ "*w4h*" && "*OZ*" == "*gnd*" && "*YK*"##"*a*" && *exist*("*G*" "4*YV*") as an example here. With more and more models added into the repository, the change of retrieval efficiency can be found in Fig. 4(a). We can see that with more models added into the repository, it is more and more time-consuming, but the query time is still acceptable. The index construction time can be found in Fig. 4(b), which shows the accumulated time for index construction from scratch. Since our index can be constructed incrementally, when a new model is added into the repository, the index can be updated immediately, and the time for index updating is very short. The storage size of indexes (including the behavior

[2] According to 7PMG proposed in [14], models should be decomposed if they have more than 50 elements. That's why we generated models with the maximum number of transitions as 50, the number of places and arcs in a model is not configurable.

feature indexes and the label index) can be found in Fig. 4(c). The storage size of indexes is less than 10% of the storage size of models. Now, we can draw a conclusion that our approach works well.

5.3 Experiments on SAP Reference Models Dataset

The SAP reference models are represented as EPCs, hence they were transformed into Petri nets using ProM [13]. This resulted in 591 Petri nets (13 SAP reference models could not be mapped to Petri nets using ProM). The characteristics of these 591 Petri nets can be found in Table 2. There are at most 1,494 differently labeled transitions out of 4,439 transitions in total. On this dataset experiments were conducted with different similarity thresholds. First, all the models were added to the repository and the indexes were built, then we used queries on the repository with different similarity thresholds. Since the performances of different queries are similar, we use an query "*Measure Processing*"$->$ "*Analysis*" as an example here.

Table 2. The characteristic of SAP models

	Number of transitions	Number of places	Number of arcs
Min	1	2	2
Max	53	83	138
Average	7.5	12.7	19.7
StDev	7.3	11.5	19.6

The size of result sets increases when the label similarity is enabled, and the size will be smaller when the similarity threshold is higher. The query time changes the same way. These changes can be seen in Table 3.

Table 3. Changes with the use of different label similarity thresholds

	disabled	0.1	0.3	0.5	0.7	0.9
result set size	2	13	9	3	2	2
query time (ms)	0	259	169	25	6	3

6 Related Work

The importance of query languages for business processes has been recognized by BPMI (the Business Process Management Initiative) who started a BPQL (Business Process Query Language) initiative in 2002. However, no draft standard was published since. Our query language *BQL* can be regarded as an example.

There are already some works on business process model query. In [15] the authors use indexing techniques to search for matched process models. However, models here are represented as annotated finite state automata, whereas we use generic Petri nets. BP-QL was proposed in [16]. It is based on an abstraction of the emerging BPEL standard, but it only can retrieve the models containing the specific nodes and paths, and ignores the run-time semantics. The VisTrails system [17] allows users to query workflow by example and to refine workflows by analogies. A workflow search engine, WISE, was proposed in [18], which returns the most specific workflow hierarchies containing matching keywords. A framework was proposed in [19], which is based on a visual query language for business process models named BPMN-Q, and makes use of the robust indexing infrastructure available by RDBMS. But all the above works only focus on graph structure, not on semantics. For example, they would return some result models in which some given tasks will not be executed at all (i.e. dead tasks), and they cannot be used to query the models in which some given tasks can be executed parallelly or some given tasks cannot be executed in the same instance as described by R_3 or R_4 in Section 1. According to our knowledge, we are the first ones to query business process models based on semantics (behavior).

7 Conclusion and Future Work

We propose a text query language in this paper. It can be used to describe the behavior features of the models we want to obtain. To extract the behavior features from models efficiently, we adopt the technology of complete prefix unfolding. For efficiency, we use indexes to support the query processing. To make our approach more applicable, we introduce the semantic similarity between labels in our approach. A system has been implemented and experiments show that our approach works well.

In this paper, we only focus on the control-flow perspective of business process models. In the future, we will extend our work to include the data and resource information as well.

Acknowledgments. We thank Arthur ter Hofstede, Chaokun Wang and Liang Song for their suggestions on this paper. The work is supported by the National Basic Research Program (973 Plan) of China (No. 2009CB320700), two NSF Projects of China (No. 61003099 and No. 61073005).

References

1. Dumas, M., Van Der Aalst, W., Ter Hofstede, A.: Process-Aware Information Systems. Wiley Interscience, Hoboken (2005)
2. Shasha, D., Wang, J.T.-L., Giugno, R.: Algorithmics and Applications of Tree and Graph Searching. In: PODS, pp. 39–52 (2002)
3. Yan, X., Yu, P.S., Han, J.: Graph Indexing: A Frequent Structure-Based Approach. In: SIGMOD Conference, pp. 335–346 (2004)
4. Cheng, J., Ke, Y., Ng, W., Lu, A.: Fg-Index: Towards Verification-Free Query Processing on Graph Databases. In: SIGMOD Conference, pp. 857–872 (2007)

5. Zhao, P., Yu, J.X., Yu, P.S.: Graph Indexing: Tree + Delta >= Graph. In: VLDB, pp. 938–949 (2007)
6. Williams, D.W., Huan, J., Wang, W.: Graph Database Indexing Using Structured Graph Decomposition. In: ICDE, pp. 976–985 (2007)
7. Lohmann, N., Verbeek, E., Dijkman, R.M.: Petri Net Transformations for Business Processes - A Survey. T. Petri Nets and Other Models of Concurrency 2, 46–63 (2009)
8. Esparza, J., Römer, S., Vogler, W.: An Improvement of McMillan's Unfolding Algorithm. Formal Methods in System Design 20(3), 285–310 (2002)
9. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems, and Computers 8(1), 21–66 (1998)
10. Polyvyanyy, A., García-Bañuelos, L., Dumas, M.: Structuring Acyclic Process Models. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 276–293. Springer, Heidelberg (2010)
11. Murata, T.: Petri Nets: Properties, Analysis and Applications, vol. 77, pp. 541–580 (1989)
12. Dice, L.R.: Measures of The Amount of Ecologic Association between Species. Ecology, 297–302 (1945)
13. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The proM framework: A new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)
14. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven Process Modeling Guidelines (7PMG). Information & Software Technology 52(2), 127–136 (2010)
15. Mahleko, B., Wombacher, A.: Indexing Business Processes Based on Annotated Finite State Automata. In: ICWS, pp. 303–311 (2006)
16. Beeri, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying Business Processes. In: VLDB, pp. 343–354 (2006)
17. Scheidegger, C.E., Vo, H.T., Koop, D., Freire, J., Silva, C.T.: Querying and Re-Using Workflows with VisTrails. In: SIGMOD Conference, pp. 1251–1254 (2008)
18. Shao, Q., Sun, P., Chen, Y.: WISE: A Workflow Information Search Engine. In: ICDE, pp. 1491–1494 (2009)
19. Sakr, S., Awad, A.: A Framework for Querying Graph-Based Business Process Models. In: WWW, pp. 1297–1300 (2010)

Discovering Implicit Categorical Semantics for Schema Matching

Guohui Ding[1,2] and Guoren Wang[1,2]

[1] Key Laboratory of Medical Image Computing (NEU), Ministry of Education
[2] College of Information Science & Engineering, Northeastern University, China
dgh_acheng@sina.com, wanggr@mail.neu.edu.cn

Abstract. Attribute-level schema matching is a critical step in numerous database applications, such as DataSpaces, Ontology Merging and Schema Integration. There exist many researches on this topic, however, they ignore the implicit categorical information which is crucial to find high-quality matches between schema attributes. In this paper, we discover the categorical semantics implicit in source instances, and associate them with the matches in order to improve overall quality of schema matching. Our method works in three phases. The first phase is a pre-detecting step that detects the possible categories of source instances by using clustering techniques. In the second phase, we employ *information entropy* to find the attributes whose instances imply the categorical semantics. In the third phase, we introduce a new concept *c-mapping* to represent the associations between the matches and the categorical semantics. Then, we employ an adaptive *scoring function* to evaluate the *c-mappings* to achieve the task of associating the matches with the semantics. Moreover, we show how to translate the matches with semantics into schema mapping expressions, and use the *chase* procedure to transform source data into target schemas. An experimental study shows that our approach is effective and has good performance.

1 Introduction

Schema matching is an essential building block in schema mapping. The basic issue of schema matching is to find attribute correspondences, namely matches. Significant attention has been paid to this topic in the literature, and a rich body of techniques have been proposed [1,4,8,11]. The task of finding matches is difficult, because one person sometimes cannot understand the meaning of attributes designed by another person accurately. As a result, schema matching continues to be a challenge, and be a valuable research problem in practice.

Recently, Bohannon et al. [6] put the *context* into schema matching for the refinement of matches. The *context* actually refers to an attribute whose values are discrete (usually binary, e.g., the attribute "gender" in table "student"). These values are able to classify the data instances into different classes (e.g., schoolboy and schoolgirl). They use the *context* to enable matches to work well for different instances (see [6] for more details). However, the *context* cannot

S.Computer

id	type	sn	user	price	date
0	ThinkPad R60	L3-T6255	Tom	8600	2001-07
1	Lenovo dx7400	CNG821143M	Prof. Li	3200	2001-07
2	ThinkPad R61i	L3-T6738	Prof. Li	9900	2007-10
3	Lenovo dc7800	CNG9020ZFY	Tom	4200	2007-10
4	Lenovo dx2700	CNG7510RK1	Jone	3600	2004-11

(a) Source Table, Instances of Computer

T.Laptop

id	model	sn	principal	price
10	ThinkPad X200	LX-T367B	Prof. Wang	9600
11	ThinkPad T410	LT-T4289	Lili	11500
⋮	⋮			

(b) Target Table, Instances of Laptop

T.Desktop

id	model	sn	principal	price	screen
20	Lenovo dx7408	CNG85201ZN	Jack	3700	LCD
21	Lenovo dx2708	CNG7230QV9	Lili	4000	CRT
⋮	⋮				

(c) Target Table, Instances of Desktop

Fig. 1. Motivating Example: Source and Target Instances

always appear in practice scenarios. Instead, it is common that the categorical information is implicit in some attribute whose values are not discrete. Consider the transformation of source data in Figure 1. $\mathbb{S}$ represents a source schema with only one table $\mathbb{S}.Computer$ which describes the information about the computers of a lab. While the information in target schema $\mathbb{T}$ is similar to that of $\mathbb{S}$, the data instances are structured in two separate tables, $\mathbb{T}.Laptop$ and $\mathbb{T}.Desktop$, in another lab. The matches m_{1-8} shown in Figure 2 are the output of a traditional matching tool. If these matches are directly used to transform source instances in $\mathbb{S}$ into $\mathbb{T}$, it will violate the semantics of $\mathbb{T}$. This is because only partial source instances conform to the semantics of $\mathbb{T}.Laptop$. A reasonable solution might be associating the matches with constraints to restrict them to be available for corresponding source instances. Unfortunately, in $\mathbb{S}$, there is no *context* which can be used to generate the constraints. As a consequence, the approach proposed in [6] falls short in our scenario of interest. In practice, this scenario is very common because of the legacy schemas with early design. For example, $\mathbb{S}$ might be created a long time ago, then, they only have the desktops. Thus, they did not design the attribute describing the type of the computer in $\mathbb{S}.Computer$. Here, someone may insist on the approach [6] by inserting an attribute that describes the type of the computer into $\mathbb{S}.Computer$. However, this method is not feasible for two reasons: first, for most practical scenarios, we have no permission to modify source schemas; second, if the table size is large, it will be a difficult task to populate the attribute inserted with the type data.

No attribute with discrete values explicitly indicating the categories of source instances notwithstanding, we discover by the observation that the categorical semantics is implicit in the instances of some attribute, e.g., the attributes "*Computer.type*" (we call it categorical attributes). In this paper, we refer to this kind of implicit categorical semantics as the constraints or the filtering conditions, in short FC, and associate them with matches to improve overall quality of schema matching. Our approach works in three phases. The first phase is a pre-detecting step that detects the possible categories of source instances by using clustering techniques. In particular, we exploit the *minimum spanning tree* (MST) clustering algorithm to partition objects of an attribute into different clusters each of which represents a possible category. The task of our second phase is to find

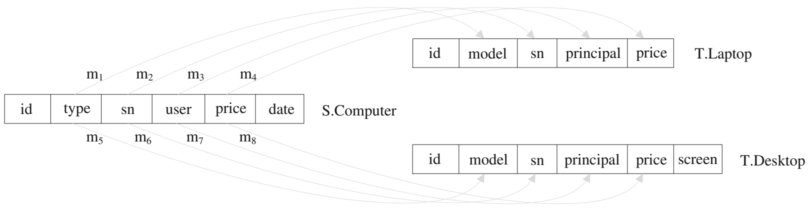


Fig. 2. Traditional Schema Matches

the categorical attributes whose instances imply the categorical semantics, like "*Computer.type*". Intuitively, we can achieve this task by checking the clustering result of the first phase. However, this intuitive method is not feasible, because there exist some interference attributes, like "*Computer.user*", whose instances can also be well clustered. To this end, we make use of *information entropy* to find the categorical attributes. In the third phase, we achieve the task of associating the matches with the corresponding FCs. Specially, we introduce a new concept *c-mapping* to represent the associations between the matches and the FCs. Then, an adaptive *scoring function* [10] is used to evaluate these *c-mappings*. Finally, we will find the optimal *c-mapping* based on the function, which corresponds to a set of optimal associations. As an example of our approach, consider two object clusters in the attribute "*Computer.type*", c_1={"T. R60", "T. R61i"} and c_2={"L. dx7400", "L. dc7800", "L. dx2700"}. Matches m_{1-4} might be associated with the FC "*type* in c_1", while m_{5-8} with "*type* in c_2". The view can be used to reflect an FC for several matches, for example, the view "select *type, sn, price, user* from *computer* where *type in* c_1" for m_{1-4}. The contributions of this paper are summarized as follows:

1. We refine the match results of standard schema matching tools by discovering the categorical semantics implicit in source instances.
2. We make use of the *minimum spanning tree* (MST) clustering algorithm to detect the possible categories of the source instances.
3. The *information entropy* is used to find the categorical attributes. Then, we achieve the association between the matches and the corresponding FCs.
4. We perform an extensive experimental study on real-world data sets. The experimental results show that the proposed algorithm has good performance.

The rest of this paper is organized as follows. Section 2 describes the traditional schema matching. The details of the three phases of our approach are discussed in Section 3. The experimental results are given in Section 4. A brief related work is reviewed in Section 5. Finally, we conclude in Section 6.

2 Preliminaries

In this section, we first introduce the basic terminology used in our approach. Then, we present an overview of the traditional schema matching techniques, including the definition of the match, the classification of the techniques, etc.

2.1 Data Model

We mainly focus on relational schema in our approach. Source schema is represented by the symbol $\mathbb{S}$, while the target schema by the symbol $\mathbb{T}$. A schema is a collection of tables which are denoted as the capital letter with a subscript sometimes, $R, R_i, R_{\mathbb{S}}, R_{\mathbb{T}}$. The attributes of R are represented by a series of low case letters $a, b, ...$, also by an attribute set, $A(R)$. We also refer to the attribute a as $R.a$ to show the ownership between the table R and its attribute a. The tuples of the source relations are also called the source instances. The set $R = \{t_1, ..., t_i, ...t_n\}$ represents the relation R and its tuples. The values of an attribute $R.a$ are denoted as the set $V(R.a)$ and called the instances of $R.a$ as well. Finally, the notations with the letter c are used to represent the clusters.

2.2 Traditional Schema Matching

Here, we briefly describe the traditional schema matching techniques. The aim of schema matching is to discover the attribute correspondences, also called matches, between a source schema and a target schema. A traditional match is of the from $(R_{\mathbb{S}}.a, R_{\mathbb{T}}.b)$, where a, b are two attributes. An accepted match indicates that the two attributes are similar semantically. To achieve this task, schema matching tools make use of a variety of matching algorithms, also referred to as *matchers*, to compute the similarity (range 0-1) of semantics between a pair of attributes (candidate match). These matching algorithms depend a lot on the kinds of information they use and how they interpret it, and typically, they are classified as schema-based or instance-based matchers. The available information for schema-level matcher includes the usual properties of schema elements, such as the attribute name, data type, schema structure and constraints etc., while the instance-level matcher exploits the statistical information derived from the contents of the schema elements, such as frequent patterns of words.

3 Finding Matches with Filtering Conditions

In this section, we detail the three phases of our techniques. Moreover, we show how to transform source data into the target schemas. Before the discussion of the phases, we present the formal description of the match with an FC.

Definition 1. *Let $(R_{\mathbb{S}}.a, R_{\mathbb{T}}.b)$ be a traditional match, where $R_{\mathbb{S}} \in \mathbb{S}$ and $R_{\mathbb{T}} \in \mathbb{T}$. The match with a filtering condition is defined as a triple $m = (R_{\mathbb{S}}.a, R_{\mathbb{T}}.b, fc)$, where fc is the filtering condition.*

This definition is uncompleted because of fc (its definition is provided later). Actually, fc corresponds to a view of the form "select * from $R_{\mathbb{S}}$ where fc", which represents a class of source instances. Thus, the match m is restricted to be just available for this category of instances in the view, not all instances in $R_{\mathbb{S}}$. Consequently, the matching results are refined and overall quality of schema matching is improved. Now, what we need to face is the generation of the FC.

3.1 Detecting Possible Categories of Source Instances

In this subsection, we show the details of how to detect the possible categories of source instances. This is the pre-detecting phase of our approach, where the preparations for the generation of the FCs are performed.

A Naive Approach. Here, we first illustrate the naive method to make our approach more approachable. To detect the possible categories, we need to group the instances of the source relation into different clusters. Each cluster represents a possible category. Intuitively, the simplest method is to consider each tuple of a table R to be a point in the space each dimension of which is an attribute of the table R. Then, the values of each attribute are normalized, and the clustering techniques based on Euclidean distance can be used to partition these points into different clusters. However, we argue that not all attributes of table R have the categorical information, e.g., "*Computer.date*" and "*Computer.user*". If many attributes of this type exist in table R, we think, the quality of the clustering result may be too bad. Consequently, the naive approach may be not feasible.

An Approach at Attribute-Level. It may not be feasible to detect the possible categories at the tuple-level. Our motivating example shows that the categorical semantics are just implicit in the instances of some attribute or several attributes, not all attributes. Hence, we consider this problem at the attribute-level. The key idea of our approach is that a category of source instances may share the similar values on some attribute, in reverse, if we can group these similar values of this attribute together, then we may find a category of source instances. In particular, for an source attribute $R_{\mathbb{S}}.a$, we consider each value of $R_{\mathbb{S}}.a$ to be a data object. If the data type of $R_{\mathbb{S}}.a$ is String, we will make use of Q-grams (3-grams) to measure the distance between objects of $R_{\mathbb{S}}.a$, while the data type of $R_{\mathbb{S}}.a$ is Numeral, the Euclidean distance is used. Then, we normalize the distance between two objects into the value in the range 0 to 1.

Definition 2. *Let $R_{\mathbb{S}}.a$ be a source attribute, and $|V(R_{\mathbb{S}}.a)| = n$. Let $o, \acute{o}$ be two objects and $o, \acute{o} \in V(R_{\mathbb{S}}.a)$. Let $|\boldsymbol{o} - \boldsymbol{\acute{o}}|$ be the distance between o and $\acute{o}$, which may be the Q-grams or Euclidean distance. We normalize $|\boldsymbol{o} - \boldsymbol{\acute{o}}|$ as follows:*

$$\bar{d}(o, \acute{o}) = \frac{|\boldsymbol{o} - \boldsymbol{\acute{o}}|}{\max(|\boldsymbol{o_i} - \boldsymbol{o_j}|_{i,j \leq n})} \tag{1}$$

Based on this distance normalized, we employ the MST clustering technique, which is an agglomerative hierarchical clustering algorithm based on the *minimum distance*, to group the data objects of an attribute into different clusters. Each cluster might represent a possible category of source instances. The details are shown in Algorithm 1, which is an iterative process. The **for** loop (lines 1-10) iteratively outputs the clustering result (C) in an attribute and inserts it into the set $\mathbb{C}_{\mathbb{D}}$ (line 10). If any two objects in $V(R.a)$ cannot be grouped into a cluster, we believe that there is no categorical information in the attribute $R.a$, and we ignore it (line 9). In lines 3-8, the distance between any $o_i, o_j \in C$ is computed, where o_i, o_j may be the object or the cluster generated by line 6.

Algorithm 1. Detecting The Possible Categories

```
input  : $R$, a table of $\mathbb{S}$;
         $\gamma$, a threshold given by users;
output: $\mathbb{C}_{\mathbb{O}}$, the set of clustering results in attributes of $R$;
1  for each attribute $R.a$ in $A(R)$ do
2      $C = V(R.a)$; // $C$ is a set storing object clusters in an attribute.
3      for each $o_i, o_j \in C$ do
4          if $\bar{d}(o_i, o_j)$ is minimum among other objects in $C$ then
5              if $\bar{d}(o_i, o_j) < \gamma$ then
6                  group$(o_i, o_j)$; //group $o_i, o_j$ into a cluster.
7              else
8                  terminate;
9      if $|C| \neq |V(R.a)|$ then
10         $\mathbb{C}_{\mathbb{O}} = \mathbb{C}_{\mathbb{O}} \cup C$;
```

In line 5, the parameter γ is a threshold provided by users ($0 < \gamma < 1$). If the distance between any objects or clusters exceeds this threshold, the clustering will terminate. The set $\mathbb{C}_{\mathbb{O}}$ is the output that contains the clustering results in the attributes of the table R. Each cluster in $\mathbb{C}_{\mathbb{O}}$ represents a possible category. Now, we present the general form of the FC based on these clusters.

Definition 3. *Let $(R_{\mathbb{S}}.a, R_{\mathbb{T}}.b, fc)$ be a match, where $R_{\mathbb{S}} \in \mathbb{S}$ and $R_{\mathbb{T}} \in \mathbb{T}$. The filtering condition fc is defined as the form $((R_{\mathbb{S}}.a \text{ in } C_{a1}) \vee ... \vee (R_{\mathbb{S}}.a \text{ in } C_{ai}) \vee ...) \wedge ((R_{\mathbb{T}}.b \text{ in } C_{b1}) \vee ... \vee (R_{\mathbb{T}}.b \text{ in } C_{bi}) \vee ...) \wedge ...$, where C_{ai}, C_{bi} are the object clusters in the attribute $R_{\mathbb{S}}.a$, $R_{\mathbb{T}}.b$, respectively.*

As it can be seen, the general form of the FC is a composite condition that logically consists of disjunctive conditions in an attribute and conjunctive conditions among attributes. The object clusters in an attribute constitute the atomic conditions of the general form. In the next phase, we will discover the categorical semantics implicit in the clusters in $\mathbb{C}_{\mathbb{O}}$.

3.2 Finding Categorical Attributes

Here, we will mainly discuss the discovery of the categorical attributes whose instances imply the categorical semantics, like "*Computer.type*". As in section 1, we cannot directly exploit the clustering results from the previous phase to achieve this task because of the interference attributes.

Intuitively, if a relation R describes several categories of instances, the characteristics of the instances of the same category may emerge in serval attributes simultaneously. For example, the characteristics "desktop" and "laptop" in our motivating example emerge in the attributes "*Computer.type*", "*Computer.sn*" and "*Computer.price*" simultaneously. This means that while clustering the instances of R according to these attributes, we may obtain the identical clustering result. We employ *information entropy* to formalize this intuition into a technique. Specially, we consider a tuple in the table R to be a "document", and an attribute to be a "clustering technique". Then the "documents" in R can be clustered according to different "techniques". It is the ideal case that the "documents" of the same category appear together in the same cluster while performing different "clustering techniques". In the worst case, a "document" belongs to n different clusters for n "techniques". Thus, the entropy may be the best choice to measure this situation from the worst to the ideal.

We show how to obtain the tuple clusters according to different attributes. The output of Algorithm 1, the set $\mathbb{C}_{\mathbb{o}}$, is of the form $\{E_1, ..., E_i, ..., E_m\}$, where $E_i = \{e_{i1}, ..., e_{ij}, ..., e_{iq}\}$. For a relation R, the symbol E_i represents the clustering result in the attribute a_i, and e_{ij} is an object cluster in E_i. Actually, an object cluster e_{ij} corresponds to a tuple cluster according to the attribute a_i, where the values token by these tuples on the attribute a_i belong to e_{ij}. Consequently, the sets of tuple clusters has the same form as $\mathbb{C}_{\mathbb{o}}$, denoted as $\mathbb{C}_{\mathbb{t}} = \{C_1, ..., C_i, ..., C_m\}$, where C_i represents the set of tuple clusters according to the attribute a_i.

Definition 4. *Let c_1, c_2 be two tuple clusters. If they satisfy one of the following conditions*: (1) $\frac{|c_1 \cap c_2|}{\min(|c_1|,|c_2|)} \geq \varepsilon$; (2) *the* jaccard coefficient $\frac{|c_1 \cap c_2|}{|c_1 \cup c_2|} \geq \varepsilon$, *we say that there exists a* homogeneous relation *between c_1 and c_2, denoted as $c_1 \asymp c_2$.*

The *homogeneous relation* represents two kinds of relationships between tuple clusters. The first is the inclusion relationship described by the condition (1), while the equality relationship described by the condition (2). We use the parameter ε to relax the two conditions because of the intrinsic inaccuracy of the clustering techniques. We test the effect of ε on the performance of our approach in the experiments. For a tuple t, $t \in c_1, t \in c_2$, if exists $c_1 \asymp c_2$, we believe the tuple t belongs to the same category while performing different "techniques".

Definition 5. *Let $R = \{t_1, ..., t_i, ..., t_n\}$ be a relation. For any tuple t_i, let the set $\mathbb{D}_{\mathbb{i}} = \{D_{i1}, ..., D_{ij}, ..., D_{ip}\}$ be a partition of $\mathbb{C}_{\mathbb{t}}$, i.e. any two elements $D_1 \cap D_2 = \emptyset$ and $D_{i1} \cup ... \cup D_{ip} = \mathbb{C}_{\mathbb{t}}$. $\mathbb{D}_{\mathbb{i}}$ also satisfies the following condition: for any D_{ij} and $|D_{ij}| \geq 2$, for any two elements $C_1 \in D_{ij}$, $C_2 \in D_{ij}$, there exist two tuple clusters $c_1 \in C_1$ and $c_2 \in C_2$ such that $t_i \in c_1$, $t_i \in c_2$ and $c_1 \asymp c_2$. Based on $\mathbb{D}_{\mathbb{i}}$, the entropy of the relation R is defined as*:

$$E(R) = -\frac{1}{|R|} \sum_{i=1}^{|R|} \sum_{j=1}^{|\mathbb{D}_{\mathbb{i}}|} p_{ij} \log_2^{p_{ij}} \tag{2}$$

$$p_{ij} = \frac{|D_{ij}|}{|\mathbb{C}_{\mathbb{t}}|} \tag{3}$$

The possibility p_{ij} represents the distribution that the tuple t_i appears in different categories. The entropy of the relation R is the average of the entropy of its tuples. The value of $E(R)$ of the ideal case above is the minimum 0, while the value is the maximum for the worst case. The entropy $E(R)$ quantifies the degree of the decentralization that all the tuples of the relation R disperse among different clusters. We use it to find the categorical attributes.

Definition 6. *Let a_i be an attribute of a relation R, and the tuple clusters according to a_i is C_i, $C_i \in \mathbb{C}_{\mathbb{t}}$. Let $E_a(R)$ be the entropy of R after ruling out C_i in $\mathbb{C}_{\mathbb{t}}$, while $E(R)$ be the original entropy. The information gain of the attribute a_i is defined as*:

$$Gain(a_i) = E(R) - E_a(R) \tag{4}$$

The key idea of our method is to rule out the interference attributes, then obtain the categorical attributes. $Gain(a_i)$ measures the overall effect of the attribute a_i on the distribution of all tuples appearing in different categories. If the tuple cluster ($C_i \in \mathbb{C}_{\mathbb{t}}$) according to the attribute a_i is different from most of other attributes, then the effect of a_i will be larger. Thus, the information gains of the interference attributes whose clustering results are different from most of other attributes will be bigger than the categorical attributes. As a result, we iteratively rule out the attribute with the maximum information gain until $E(R) = 0$ or the information gains of all the remaining attributes are identical. The details are shown in Algorithm 2. The first termination condition $E(R) = 0$ represents that all the rest attributes imply one kind of categorical semantics, while the second shows that there exist multiple kinds of categorical semantics implicit in the rest attributes, each of which has the same effect on the overall distribution of the tuples. Finally, we will obtain a new set $\mathbb{C}_{\mathbb{o}}$, each object cluster of which represents an atomic FC. We will perform the task of our next phase based on this result.

3.3 Associating Matches with Filtering Conditions

In this section, we show how to generate the FCs in Definition 3, then associate them with the matches, namely the finally phase of our approach.

So far, we have finished the tasks of our previous two phases, i.e., in our motivating example, we have discovered the categorical attributes "*Computer.type*" etc., and the atomic FCs, "*type* in c_1" and "*type* in c_2". Now, what we need to face is to make a decision about associating the condition c_1 (c_2) with the matches m_{1-4} or m_{5-8}. This task can also be referred to as the problem of finding a mapping between the clusters c_1, c_2 and the instances of the target attributes "*Laptop.model*", "*Desktop.model*". A simple method is to compute the distance between the clusters and the instances in the corresponding target attributes. Then, a threshold is used to make this decision. However, if the target attribute implies additional categories except for the one implicit in c_1 (c_2), the distance will be larger, thus, the threshold may result in a wrong decision. For a target attribute $R_{\mathbb{T}}.b$, we use the same method in our first phase to group the objects of $R_{\mathbb{T}}.b$ into different clusters. Then, our task is turned into finding a mapping between the clusters in the categorical attribute and the clusters in the

Algorithm 2. Finding Categorical Attributes

```
   input  : $\mathbb{C}_{\mathbb{t}}, \mathbb{C}_{\mathbb{o}}$;
1  for $1 \leq i \leq m$ do
2      float $maxGain = 0.0$;
3      int $flag = 0$, $j = 0$;
4      for each $C_i \in \mathbb{C}_{\mathbb{t}}$ do
5          if $maxGain \leq Gain(a_i)$ then
6              if $maxGain == Gain(a_i)$ then
7                  $flag = flag + 1$;
8              $maxGain = Gain(a_i)$; $j = i$;
9      if $E(R) == 0$ or $flag == |\mathbb{C}_{\mathbb{t}}| - 1$ then
10         break;
11     remove $C_j \in \mathbb{C}_{\mathbb{t}}, E_j \in \mathbb{C}_{\mathbb{o}}$;
```

corresponding target attributes. We aim to find the mapping where each pair of clusters are similar, i.e., they imply the same or similar category. We employ the scoring function [10] to measure the overall similarity of the possible mappings and find the mapping that maximizes the scoring function, denoted as m^*.

Definition 7. *Let $(R_{\mathbb{S}}.a, R_{\mathbb{T}}.b)$ be a traditional match where $R_{\mathbb{S}} \in \mathbb{S}$, $R_{\mathbb{T}} \in \mathbb{T}$. Let c_a, c_b be two object clusters in $R_{\mathbb{S}}.a$, $R_{\mathbb{T}}.b$ respectively. We call (c_a, c_b) the cluster match, CM for short, and call c_a the source cluster, c_b the target cluster.*

Definition 8. *Let $m = \{cm_1, ..., cm_i, ..., cm_n\}$ be the set of the cluster matches where $cm_i = (c_{ai}, c_{bi})$. If m satisfies the following conditions: (1) for $1 \leq i \leq n$, the clusters c_{ai}, c_{bi} occur only once in m; (2) all the clusters c_{ai}, $1 \leq i \leq n$, come from the same categorical attribute, we say that m is the c-mapping.*

As the definition shows, the *c-mapping* is the one-to-one mapping. A categorical attribute corresponds to a set of possible *c-mappings* and one optimal *c-mapping* m^*. A *c-mapping* involves only one source attribute but several target attributes, and there exist the attribute matches between them.

Definition 9. *Let A, B be two object clusters, and $o \in A$, $\acute{o} \in B$. Let $\bar{d}(o, \acute{o})$ be the distance between objects o and $\acute{o}$. Let $\bar{d}(o, B)$ be the distance between the object o and the cluster B, and $\bar{d}(o, B) = \min \bar{d}(o, \acute{o})_{\acute{o} \in B}$. Similarly, $\bar{d}(\acute{o}, A) = \min \bar{d}(o, \acute{o})_{o \in A}$. Let d_m be the maximum distance between the objects in A, B. The distance between A and B is defined as:*

$$\bar{d}(A, B) = \frac{1}{2d_m}\left(\frac{1}{|A|}\sum_{o \in A} \bar{d}(o, B) + \frac{1}{|B|}\sum_{\acute{o} \in B} \bar{d}(\acute{o}, A)\right) \tag{5}$$

Any one of the two summations in brackets is Hausdorff distance [12], and we use it to define the distance between two object clusters. Based on this distance, we begin to discuss the scoring function [10]. Let k_m be the number of the CMs contained in a *c-mapping*. For a categorical attribute a, the corresponding k_{m^*}

is not always equal to the number of the object clusters in a, because there may be no corresponding category in the target instances. For example, if the table "*Laptop*" does not exist in $\mathbb{T}$, for the categorical attribute "*Computer.type*", the corresponding k_{m^*} is equal to 1 (here, c_1 is redundant). The scoring function in [10] is designed to compare between different mappings. It might be able to automatically estimate the correct value of k_{m^*} by the quantile. Given a *c-mapping* m and any CM $(c_{ai}, c_{bi}) \in m$, this function has the following form:

$$f(m) = \sum_{i=1}^{k_m} (1 - \alpha_i \bar{d}(c_{ai}, c_{bi})) \tag{6}$$

The key idea of the function is to reward or penalize the *c-mappings* according to the control variable α_i. Let d_i be the value of the distance in Equation 6. If we consider the distribution of d_i among all the possible *c-mappings*, we may expect that d_i for m^*, denoted as d_i^*, is among the smallest values in this distribution. Consequently, if d_i^{g} is the value of the g-quantile of d_i, for some small value g, then d_i^* should be smaller than d_i^{g}. Thus, if α_i is set to $\frac{1}{d_i^{\mathsf{g}}}$, the value of the difference in brackets in Equation 6, may be positive for most values of i, and therefore m^* is rewarded the most among other *c-mappings* [10]. For any *c-mapping* m whose k_m is greater than the correct value, it will contain some wrong CMs whose d_i is expected to be greater than d_i^{g}, and therefore the value of the difference will be negative and m will be penalized. The analysis is similar for the other case. In this way, this function can estimate the correct k_{m^*}.

In practice, the number of the object clusters in an attribute may be few. Thus, we enumerate all the possible *c-mappings* for a categorical attribute to find the corresponding m^* based on the scoring function above. We use the object clusters in a categorical attribute to form the disjunctive condition in Definition 3. Specially, for the categorical attribute $R_{\mathbb{S}}.a$ and its corresponding m^*, if there exists a subset of m^*, denoted as $m_1 = \{cm_1, ..., cm_i, ..., cm_k\}$, whose target clusters all come from the same target attribute $R_{\mathbb{T}}.b$, we will combine all the source clusters of m_1 into the disjunctive condition in Definition 3. The subset m_1 represents that the target relation $R_{\mathbb{T}}$ implies several categories of instances. Thus, we need to merge these atomic FCs in m_1 into a disjunctive condition. In the same way, we may find the disjunctive conditions in other categorical attributes. Because there may exists *homogeneous relation* among the atomic FCs of different categorical attributes, we need to combine these disjunctive conditions of different categorical attributes into the conjunctive condition, i.e., generate the final FC in Definition 3. Finally, we achieve the task of associating the matches with the final FCs.

3.4 Transforming Source Data

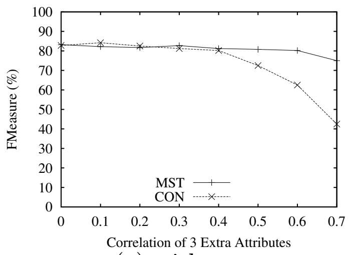
Here, we show how to use the matches with the FCs to generate schema mapping that, in essence, is to transform source data to the target schema. Schema mapping is typically expressed by the declarative language of the form $\forall \overline{x}(\phi(\overline{x}) \rightarrow \exists \overline{y}(\psi(\overline{x}, \overline{y})))$, where $\overline{x}$, $\overline{y}$ are vectors of variables, and $\phi(\overline{x})$, $\psi(\overline{x}, \overline{y})$ are the conjunction of atomic formulas over the source relation and the target relation respectively, and we call it mapping expression. The generation method of schema mapping we described here is the generalization of the technique introduced in [11]. As the first step, we need to find the *primary paths*. Because we only consider the relational schema, the *primary paths* here are the sets of the instances of each relation. The next step is concerned with generating the *logical relations*. In our scenario, a *logical relation* is a view that is made of several tables associated via the foreign key constraints. With the *logical relations* generated, a number of mapping expressions can be produced. If there exists at lest one match m between an attribute of the source *logical relation* lr_s and an attribute of the target *logical relation* lr_t, we will generate a mapping expression. The lr_s is used to generate the left-hand side of the expression, while lr_t generating the right-hand side. All the attributes in lr_s are used to generate universally quantified variables in the expression. For each attribute in lr_t which is not involved in any matches between lr_s and lr_t, we add an existentially quantified variable to the right-hand side of the expression. As an example, consider the matches m_{5-8} with the FC "*type* in c_2". Because of no foreign key constraints in this example, each table is a *logical relation*. We can obtain the following mapping expression: $\forall i, t, s, u, p, d : Computer(i, t, s, u, p, d) \wedge t \in c_2 \rightarrow \exists I, S : Desktop(I, t, s, u, p, S)$. Finally, given the source instances, the standard *chase* procedure [4] is used to chase the generated expressions to achieve the transformation of source data.

4 Experimental Evaluation

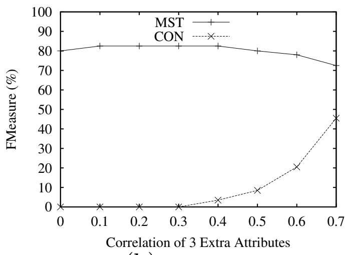
In this section, we evaluate the performance of our approach on the real-world data set [2]. First, we compare our approach with the traditional approach [6]. Then, we show the experimental results evaluating the performance of our algorithm. We evaluate the accuracy against the correct mappings determined by manual inspection of the source and target schemas. Our algorithm is implemented using C++ language and the experiments were carried on a PC compatible machine, with Intel Core Duo processor (2.33GHz).

The data set used in the experiments is from Illinois Semantic Integration Archive [2], which contains house listing information from several real estate websites. These houses can be classified into four types: tiny, small, middling and big, according to the building area, etc. We use part of these data to form the source schema including one single table which contains four types of houses, while the other part is used to create the target schema including two separate tables each of which contains two types of houses.

We compare our approach (MST) with the approach [6] (CON). We obtain the source code of CON from the original author. We set the parameters associated with CON as follows: $\tau = 0.5$, $\omega = 5$, $SrcClassInfer = true$ and $EarlyDisj = true$. We consider two cases for CON: with *context* and no *context*. As in [6], we also add 3 extra correlated attributes in the source schema. For the first case, we insert an attribute (*context*), whose domain may be {t,s,m,b} corresponding to the four types of houses. The experimental results are shown in Figure 3(a). As ρ increases, our approach has almost the same accuracy as CON when $\rho < 0.5$,

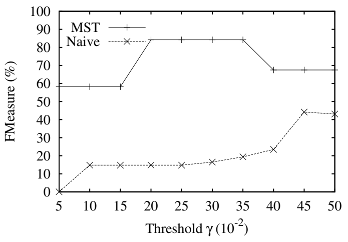


(a) with *context*



(b) no *context*

Fig. 3. Our Approach VS. Traditional Approach With Context



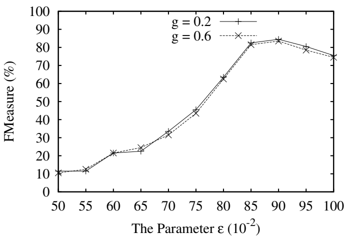


Fig. 4. MST VS. Naive

Fig. 5. Varying Parameter ε

then performs well than CON. For the second case, we remove the *context*, but preserve the correlated attributes. The results are shown in Figure 3(b). We can see that the curve for CON is close to the x-axis when $\rho < 0.4$. This is because CON cannot work without *context*. Then, the accuracy of CON increases when $\rho > 0.4$. The reason is that the correlated attributes amount to the *context* when ρ of these extra attributes are gradually close to 1. We can see that the correlated attributes have little negative effect on our approach.

In Figure 4, we compare MST with the naive approach (set g to 0.2). MST performs better than the naive one with the increase of the threshold γ. The accuracy of MST obtains the maximums between $\gamma = 0.2$ and $\gamma = 0.35$. Thus, we set γ to 0.3 in the other experiments. We study the effect of the variation in the parameter ε, which is used to relax the conditions in Definition 4, on the performance. The results are shown in Figure 5. The accuracy increases when the parameter ε changes from 0.5 to 0.9, then the accuracy decreases when $\varepsilon > 0.9$. The two curves for $\mathsf{g} = 0.2$, $\mathsf{g} = 0.6$ are almost the same. This behavior will be explained in the following experiments. We set ε to 0.9 in other experiments.

The experiment in Figure 6 studies the effect of increasing the number (β) of categories of the instances on the performance. To this end, we replace the original values of the categorical attributes with new synthetic values, then these house instances can be classified into several types. We can see that the accuracy keeps stable and high values when $\beta \leq 5$, then the two curves drop at the point $\beta = 6$. The reason is that a large number of *c-mappings* generated when $\beta = 6$ affect the performance of our third phase.

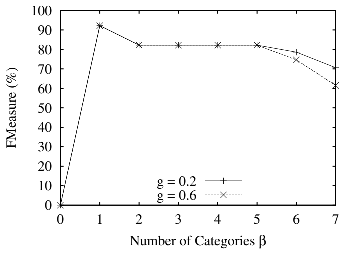


Fig. 6. Varying Categories

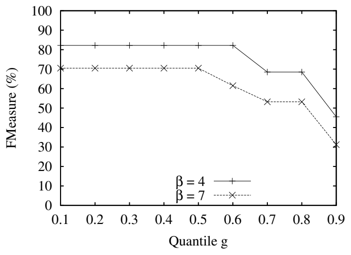
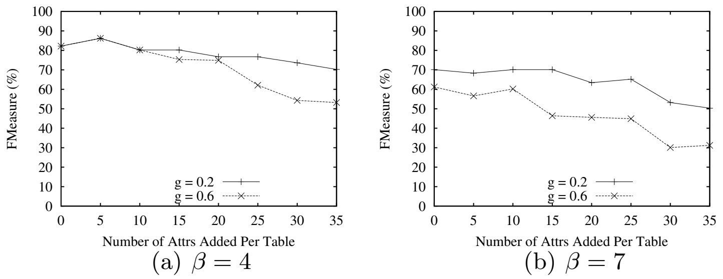
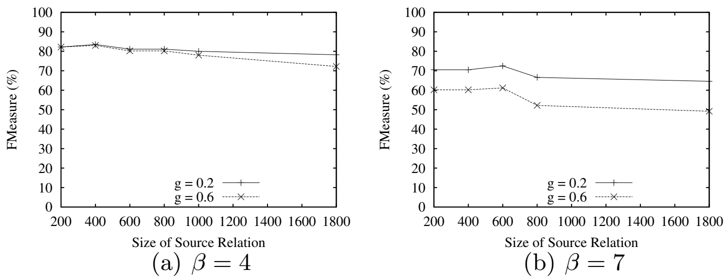


Fig. 7. Varying Quantile



(a) $\beta = 4$ (b) $\beta = 7$

Fig. 8. Scalability in Schema Size

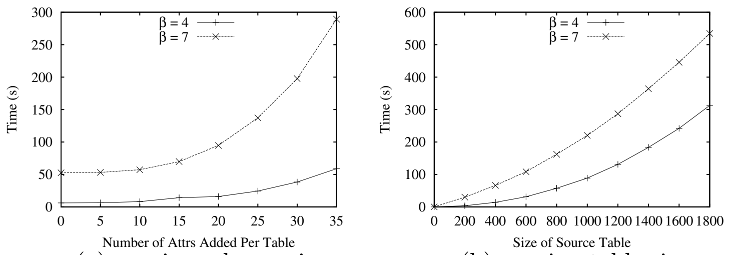


(a) $\beta = 4$ (b) $\beta = 7$

Fig. 9. Scalability in Table Size

We study the effect of the variation in the quantile on the performance. The results are shown in Figure 7. The performance stays at an invariable and high accuracy for most of the values of g. However, the accuracy decreases when the parameter g beyond the value 0.6. This behavior is consistent with the analysis in our third phase in Section 3.

To test the scalability of our approach, we vary the size of the schemas by adding extra n attributes, where $\frac{n}{5}$ is categorical attributes while the rest is non-categorical ones. The experimental results are shown in Figure 8. The accuracy decreases with the increase of the number of the attributes added. The downtrend of the curves is slow for small β in Figure 8(a). The performance for $\beta = 7$ drops more obvious than the one for $\beta = 4$. This behavior corresponds with the result of the previous experiment.



(a) varying schema size (b) varying table size

Fig. 10. Time Cost of Our Approach

In Figure 9, we can see that our approach performs well and changes slowly with the increase of the table size. The curves for $\mathsf{g} = 0.2$ and $\mathsf{g} = 0.6$ are very similar in Figure 9(a). The corresponding curves in Figure 9(b) are different, but also change slowly. We can see that the quantile is more sensitive to the number of the attributes and the parameter β than the table size.

Finally, we test the time cost in Figure 10. Our approach runs slowly with the increase of the schema size and the table size. The overall time cost of increasing the table size is more than the time of increasing the schema size, because the number of tuples in a table is much more than the number of its attributes. The time curve for $\beta = 7$ is much higher than the curve for $\beta = 4$. The reason is that there are a larger number of *c-mappings* generated when $\beta = 7$.

5 Related Work

A basic problem in data exchange is schema matching [3,5,6,7,9,10], which is a long-standing research problem. A recent work [6] proposes an approach which puts the context into schema matching. The context is actually the discrete values of some attribute, which can explicitly classify the source instances into different categories. However, their approach cannot deal with the scenario where the value domain of the attribute, which has the categorical semantics, is continuous. Our approach is proposed for this drawback of their work.

The automatic approaches to schema matching are summarized in [3]. A multi-column substring matching [5] is presented to detect complex schema translations from multiple database columns. Recently, the possible mapping is introduced to schema matching [7,9]. The work of [10] defines a new class of techniques for schema matching, which exploit information extracted from the query logs to find correspondences between attributes.

Schema matching is a preliminary step for schema mapping [1,4,8,11]. The Clio system [1] generates SQL-like mappings based on attribute correspondences. A semantic approach to discovering schema mapping expressions is proposed in the work of [8]. They uses the concept model to discover the semantics between schemas. In [11], they introduce several algorithms contributing to bridge the gap between the practice of mapping generation and the theory of data exchange.

6 Conclusions

In this paper, we discover the categorical semantics implicit in source instances, and associate them with the matches in order to improve overall quality of schema matching. Our method works in three phases. In the first phase, we detect the possible categories of the source instances by using the clustering technique. In the second phase, we aim to find the categorical attributes by ruling out the interference attributes. The *information entropy* is employed to quantify the degree of the decentralization that all the tuples of a relation disperse among different categories. Then, we use it to rule out the interference attributes. In the third phase, we achieve the associations between the matches and the semantics. Specially, the Hausdorff distance is used to measure the similarity between the source cluster and the target cluster. Based on this distance, we exploit the scoring function [10] to find the optimal *c-mapping* to achieve the task of this phase. Moreover, we translate the matches with the FCs into schema mapping expressions, and in turn, use the *chase* procedure [4] to transform source data into target schemas based on these expressions. Finally, we compare our approach with the traditional homogeneous approach [6]. The experimental results show that our approach performs well.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant No. 60773219, 60803026 and 61073063) and the 863 High Technique Program (Grant No. 2009AA01Z150).

References

1. Miller, R.J., Haas, L.M., Hernandez, M.A.: Schema Mapping as Query Discovery. In: Proc. of VLDB, pp. 77–99 (2000)
2. Doan, A.: Illinois semantic integration archive
3. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal 10(4), 334–350 (2001)
4. Fagin, R., Kolaitis, P., Miller, R., Popa, L.: Data exchange: Semantics and query answering. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 207–224. Springer, Heidelberg (2002)
5. Warren, R.H., Tompa, F.: Multicolumn Substring Matching for Database Schema Translation. In: Proc. of VLDB, pp. 331–342 (2006)
6. Bohannon, P., Elnahrawy, E., Fan, W., Flaster, M.: Putting context into schema matching. In: Proc. of VLDB, pp. 307–318 (2006)
7. Dong, X., Halevy, A.Y., Yu, C.: Data integration with uncertainty. In: Proc. of VLDB, pp. 687–698 (2007)
8. An, Y., Borgid, A., Miller, R.J.: A semantic approach to discovering schema mapping expressions. In: Proc. of ICDE, pp. 206–215 (2007)

9. Sarma, A.D., Dong, X., Halevy, A.: Bootstrapping Pay-As-You-Go Data Integration Systems. In: Proc. of SIGMOD, pp. 861–874 (2008)
10. Chan, C., Elmeleegy, H.V.J.H., Ouzzani, M., Elmagarmid, A.: Usage-Based Schema Matching. In: Proc. of ICDE, pp. 20–29 (2008)
11. Mecca, G., Papotti, P., Raunich, S.: Core Schema Mappings. In: Proc. of SIGMOD, pp. 655–668 (2009)
12. Radwan, A., Popa, L., Stanoi, I.R., Younis, A.: Top-K Generation of Integrated Schemas Based on Directed and Weighted Correspondences. In: Proc. of SIGMOD, pp. 641–654 (2009)

Expressive Power of Query Languages for Constraint Complex Value Databases

Hong-Cheu Liu

Department of Computer Science and Multimedia Design
Taiwan Shoufu University
Tainan County, 72153 Taiwan
hongcheu.liu@gmail.com

Abstract. Motivated by constraint complex values which allow us to represent nested finitely representable sets, we study the expressive power of various query languages over constraint complex value databases. The tools we use come in the form of collapse results which are well established results in the context of first-order logic. We show that active-generic collapse carries over to second-order logic for structures with o-minimality and any relational signature in the complex value model. We also consider the problem of safety in the context of embedded finite complex value models and constraint complex value databases.

1 Introduction

Database management systems are being widely used to support recent applications such as engineering design, image or voice data management, spatial information systems and bioinformatics. Many advanced applications involve voluminous data, which may sometimes be convenient to be considered 'infinite' data. To tackle applications that involve possibly infinite data, *constraint databases* are proposed to store a *finite representation* of an infinite set and allow users to express queries on such representation as if the entire infinite set was stored.

In particular, spatial or temporal databases often contain voluminous data points in a multidimensional space. These data points can be expressed as a set of constraints, where the actual data points are defined to be the points that satisfy such constraints. In many applications, spatial or temporal data are often more intuitively described as constraints. This may be the reason why constraint databases are emerging to be a unifying paradigm for conceptual representation of spatial or temporal data.

The *constraint database model*, introduced by Kanellakis, Kuper and Revesz in their seminal paper [1], is a powerful generalization of the relational data model with finitely representable infinite relations. This model facilitates efficient declarative database programming combined with efficient constraint solving: A *generalized tuple* is a finite conjunction of constraints in some constraint theory. For example, $x \leq y \wedge x \leq 0$ defines a binary generalized tuple. A *generalized* *relation* is a finite set (disjunctions) of generalized tuples. For example, the constraint relation $R(x, y) = 20 < x \vee y < 7 \vee x = y$ has three generalized tuples. The above relation R is an infinite relation. The formula of a generalized relation R can also be viewed as the formula of its generalized tuples put in disjunctive normal form (DNF), $\psi_1 \vee \psi_2 \vee \cdots \vee \psi_n$. We may use φ_R to denote a quantified formula corresponding to the relation R. A *generalized database* is a finite set of generalized relations.

Constraint databases have been an active area of database research. One of the most challenging questions in the theoretical development of constraint databases is the expressive power of their querying formalisms: what are the limitations of query languages for constraint databases, e.g., [2]? In particular, the classical techniques for analyzing the expressive power of relational query languages no longer work in the context of constraint databases. In the past several years, most questions on the expressive power based on some characterization of structures and a relational schema have been settled. These questions were reduced to those on the expressiveness of query languages over ordinary *finite* relational databases, with additional condition that databases may store numbers and arithmetic operations (which form linear or polynomial equations) may be used in queries [3].

In this paper, we investigate in detail on the expressive power of the various query languages for the constraint complex value model. Our study extends the constraint database model in two ways (to be detailed soon): (i) we allow constraint complex values and (ii) we consider second order query languages for complex values. Regarding the expressive power for languages for constraint databases, there are well-established results. Many results on expressive power use the notion of *genericity*, which comes from the classical relational database setting. Therefore, to begin our investigation, we first review the concept of *genericity* of queries in our problem setting.

Generic queries. In general, generic queries commute with permutations on the domain. For example, the answer to the parity query is the same for the input $\{1, 2, 3, 4, 5\}$ and for another input $\{a, b, c, d, e\}$, which is obtained by the mapping $1 \mapsto a$, $2 \mapsto b$, $3 \mapsto c$, $4 \mapsto d$, $5 \mapsto e$. The main tools adopted in these techniques are *collapse results* [4,3,2]. These collapse results mean that query class A has no more expressive power with respect to some characteristics (e.g., generic queries) than query class B, where query class A may appear to be much larger than query class B. For example, given the real order field $(\mathsf{R}, +, *, 0, 1, <)$ and a relational signature SC, the classes of generic queries in first-order (FO) logic over the real order field and SC, under active domain interpretation, and FO logic over the universe domain are the same. It is worth-remarking that attention on genericity of database queries with *embedded finite model* theory is very often paid, where finite structures are embedded in an infinite structures.

Complex Values. Complex values provide flexible modeling, which is often convenient for advanced database applications. They are represented by hierarchical structures rather than flat relations. Intuitively, complex value relations are relations in which their entries may themselves be tuples or (nested) relations. In other words, the complex value model allows using the *tuple* and *set* constructors recursively. It should be remarked that this model provides the core structure of object-relational databases and comprises an important component of many semantic models.

We extend the constraint database model with complex values. This extended model allows us to represent nested finitely-representable relations and sets. Thus, applications over natural spatial-temporal objects can be easily modeled, without casting the objects into flat relations. For example, many spatial databases involve hierarchical data. In particular, populations of cities, rainfall of regions, areas of river, *etc.*, are properties associated with sets of possibly infinite points. Multi-layered geographic information systems (GIS) may represent many regions and channels which in turn are represented by several atomic spatial objects, like lines and triangles. These properties occur naturally in many advanced GIS and applications of the constraint data model.

Higher order languages. In addition to the above practical considerations, we consider *constraint query languages* in the contexts of embedded finite models and constraint databases, motivated primarily by their expressive power. It is well-known from literature that there are a number of limitations of first-order logic. For example, queries such as parity, majority, connectivity, transitive closure and acyclic property are *not* definable in FO logic, with linear or polynomial constraints. It is natural to turn one's attention to a more expressive query language to bypass these limitations. *Second-order* constraint query languages for complex values appear to be a promising approach. We follow a popular technique in the research of embedded finite models and constraint databases which adopts the tool of collapse results to analyze the expressive power of query languages.

Relational database queries are required to have certain *closure* property: they return finite outputs on finite inputs. This requirement is well known in the database theory under the name of *query safety*: we identify those formulas which return finite results. In this paper, we consider this safety issue in the context of embedded finite complex value model.

Main contributions. We summarize our main results of this paper as follows.

- We extend the constraint data model to constraint complex values and propose the most important constraint query languages for embedded finite complex value model and constraint complex value databases.
- We study second-order logic over embedded finite complex value model and constraint complex value databases settings. We show that natural-active collapse with a condition and active-generic collapse carry over to second-order logic for structures with o-minimality and any relational signature in the complex value model.
- We show that complex value Datalog language with stratified negation over polynomial constraints is closed. Some topological properties, for example, connectivity, can be definable in second-order logic.

– We develop an approach to restricting constraint complex value calculus languages with negation to safe formulas that guarantee output answers in closed-form.

Related work. For a general introduction to finite model theory, see [5,6]. Constraint databases were introduced in [1]. The active generic collapse was proved independently in [4] and [7]. Benedikt et al. [4] proved that even simple recursive queries like transitive closure cannot be expressed in relational calculus with polynomial arithmetic constraints over the real numbers. The nature-active collapse is from [8]. Safety is a central notion in relational database theory, see [9]. Many researchers have attempted to tackle this safe issue in constraint query languages [10,11,12]. Some attempts to extend the definitions of safe and domain independence have already been made [13,14,15,16,17]. Grumbach and Su study the complexity and the expressive power of various query languages over dense order constraint databases [18].

Organization. We introduce notations in Section 2. In Section 3, we review the concept of complex values and extend the constraint data model to constraint complex values. Then we first give a formal definition of the embedded finite complex value setting and define second-order logic over this setting. We also propose the most important constraint query languages for constraint complex value databases. We analyze expressive power of query languages in the context of constraint complex value data model in Section 4. The techniques that we use normally come in the form of *collapse results*. These techniques reduce many questions over constraint databases or embedded finite models to the classical finite model theory setting. In Section 5, we consider the problem of safety in both settings. Finally, we give a conclusion in Section 6.

2 Notations

We first briefly review the basic concepts of constraint databases in this section, then extend it to constraint complex object databases in the next section. Most notations are adopted from the literature on constraint databases.

Structures, constraint databases, queries
Let $\mathcal{M} =< U, \Omega >$ be an infinite structure, where U is an infinite set, called a universe, and the signature Ω contains some function, predicate and constant symbols. Let SC be a relational signature $\{R_1, ..., R_l\}$ where each relation symbol R_i has arity $p_i > 0$. Then an embedded finite model is a structure $D =< A, R_1^D, ..., R_l^D >$, where each R_i^D is a finite subset of U^{p_i}, and A is the union of all elements that occur in the relations $R_1^D, ..., R_l^D$. The set A is called the *active domain* of D, and is denoted by $adom(D)$.

Given a structure $\mathcal{M} =< U, \Omega >$ and a relational signature SC, the syntax and semantics of first-order logic over $\mathcal{M}$ and SC, denoted by FO(SC, $\mathcal{M}$), are fairly standard as described in the literature. There are two different universes that can be quantified over: the universe U of the infinite structure $\mathcal{M}$, and the active domain A of the finite structure D. Given a FO(SC, $\mathcal{M}$) formula $\varphi(x_1, ..., x_n)$, and $\boldsymbol{a} = (a_1, ..., a_n) \in U^n$, the notion of satisfaction relation $(\mathcal{M}, D) \models \varphi(\boldsymbol{a})$ is defined in a standard manner.

Definition 1. *[4] Given a structure $\mathcal{M} =< U, \Omega >$, a set $X \subseteq U^n$ is called $\mathcal{M}$-definable if there exists a formula $\varphi(x_1, ..., x_n)$ in the language of $\mathcal{M}$ such that $X = \{\boldsymbol{a} \in U^n \mid \mathcal{M} \models \varphi(\boldsymbol{a})\}$. A constraint database of schema $SC = \{R_1, ..., R_l\}$ is a tuple $\boldsymbol{D} = < R_1^D, ..., R_l^D >$, where each R_i^D is a definable subset of U^{p_i}, where p_i is arity of R_i.*

Many results on expressive power use the notion of *genericity*, which is sometimes stated as data independence principle. We now define genericity of Boolean queries and non-Boolean queries. Given a function $\pi : U \to U$, we extend it to finite SC-structures D by replacing each occurrence of $a \in adom(D)$ with $\pi(a)$.

Definition 2. *[2]*

- *A Boolean query Q is totally generic (order-generic) if for every partial injective function (partial monotone injective function, resp.) π defined on $adom(D)$, $Q(D) = Q(\pi(D))$.*
- *A non-Boolean query Q is totally generic (order-generic) if for every partial injective function (partial monotone injective function, resp.) π defined on $adom(D) \cup adom(Q(D))$, $\pi(Q(D)) = Q(\pi(D))$.*

Note that the main difference between the definition of an embedded finite model and a constraint database is that in the former we interpret the SC-predicates by *finite* sets, and in the latter - by *definable* sets which may be infinite.

Classical model theory provides us with many examples of structures which have been considered in the field of constraint databases. A few are listed below.

- dense order constraints $(\mathsf{R}, <)$;
- linear constraints $\mathbf{R}_{lin} = (\mathsf{R}, +, -, 0, 1, <)$;
- polynomial constraints $\mathbf{R} = (\mathsf{R}, +, *, 0, 1, <)$; and
- exponential constraints $\mathbf{R}_{exp} = (\mathsf{R}, +, *, e^x, <)$.

The above structures admit *quantifier-elimination.*

In constraint databases, we normally consider relational calculus, or *first-order*, FO, over the underlying structure and the database schema, as a basic query language. The output of an FO($SC, \mathcal{M}$) formula $\varphi(x_1, ..., x_n)$ on a finite SC-structure D is $\varphi(D) \overset{\text{def}}{=} \{\boldsymbol{a} \in U^n \mid D \models \varphi(\boldsymbol{a})\}$

O-minimality. To impose additional restrictions on the underlying structure, we consider the model-theoretic notion of o-minimality, which plays an important role in the study of constraint query languages. An ordered structure $\mathcal{M} =< U, \Omega >$ is *o-minimality* if every definable set is a finite union of points and open intervals $(a, b) = \{x \mid a < x < b\}$, $(-\infty, a) = \{x \mid x < a\}$, and $(a, \infty) = \{x \mid x > a\}$. The above mentioned structures are known o-minimal structures.

3 Constraint Complex Value Model and Query Languages for Constraint Complex Value Databases

3.1 Constraint Complex Value Model

In this subsection, we review the concept of complex values (CV) and extend the constraint data model to constraint complex values. Complex values are formed by using two constructors: *tuple* and *set* and associated with sorts.

We first define syntax and semantics for complex values.

Definition 3. *(1) The abstract syntax of sorts is given by* $\tau = \boldsymbol{dom} \mid < A_1 : \tau, \cdots, A_k : \tau] \mid \{\tau\}$, *where* $k \geq 0$ *and* $A_1, \cdots, A_k$ *are distinct attributes. (2) The interpretation of sort* τ *(i.e., the set of values of* τ*), denoted* $[\![\tau]\!]$, *is defined recursively as follows.*

– $[\![\boldsymbol{dom}]\!] = \boldsymbol{dom}$,
– $[\![\{\tau\}]\!] = \mathcal{P}([\![\tau]\!]) = \{X | X \subseteq [\![\tau]\!]$ *and* X *finite*$\}$, *and*
– $[\![A_1 : \tau_1, \cdots, A_k : \tau_k]\!] = [\![\tau_1]\!] \times \cdots \times [\![\tau_k]\!]$

$\mathcal{P}$ denotes power-set operator. A complex value database schema SC is a finite set of relation names $\{R_1, ..., R_l\}$ with associated sorts $\tau_1, ..., \tau_l$.

We then define the constraint complex value model which allows us to represent nested finitely re-presentable infinite complex value databases. Constraint complex values are built using tuple and set constructors from generalized tuples and finite re-presentable sets. Many properties over natural spatial/temporal objects can be easily modeled as constraint complex values.

Definition 4. *In the context of the constraint complex value model, for each sort* τ*, the domain of* τ *denoted* $dom(\tau)$*, is defined recursively as follows.*

– *If* τ *is an* n*-ary flat tuple type, then* $dom(\tau)$ *is the set of all generalized* n*-ary tuple.*
– *If* $\tau = \{\tau'\}$ *is a set type, then* $dom(\tau) \stackrel{\text{def}}{=} \{\bigvee_{i=1}^{k} \psi_i | k \geq 1, \forall i \in \{1, \cdots, k\}, \psi_i \in dom(\tau')\}$.
– *If* $\tau =< \tau_1, \cdots, \tau_k >$ *is a tuple type, then*

$$\begin{aligned} dom(\tau) \stackrel{\text{def}}{=} \{ \textstyle\bigwedge_{i=1}^{k} \psi_i | \ & \psi_i \in dom(\tau_i), \text{ if } \tau_i \text{ is not a set tuple;} \\ & \psi_i \equiv x_i = \{\phi_i\}, \text{ where } \phi_i \in dom(\tau_i') \\ & \text{if } \tau_i \text{ is a set tuple } \{\tau_i'\}. \ \} \end{aligned}$$

Example 1. Let $\tau = [\{[Q, Q]\}, Q]$ be a tuple type. A constraint complex value of type τ is $\psi(x, y) \stackrel{\text{def}}{=} (x = \{(x_1, x_2) | \phi\} \wedge y = 10)$, where $\phi = (0 < x_1 < 5 \wedge x_1 < x_2 < 5)$.

3.2 Query Languages for Constraint Complex Value Databases

In this subsection, we define query languages for constraint CV databases in three different paradigms. We first define the embedded finite CV setting which is an extension of embedded finite model. Then we define second-order logic (SO) over this setting.

Embedded finite CV model

Definition 5. *Let $\mathcal{M} =< U, \Omega >$ be an infinite structure on a set U, where the signature Ω contains some functions, predicates and constant symbols. Let SC be a complex value relational signature $\{R_1, ..., R_l\}$ where each relation symbol R_i has sort τ_i. Then an embedded finite complex value model is a structure*

$$D =< A, R_1, ..., R_l >,$$

where each R_i is a finite subset of $\mathcal{U}^\tau$ ($\mathcal{U}^\tau$ denotes the set of complex value constants with sort τ), and the set A is the $adom(D)$.

In the embedded finite complex value setting, we use second-order logic. We give a formal definition of second-order logic as follows.

Definition 6. *Given a structure $\mathcal{M} =< U, \Omega >$ and a complex value relational signature SC, second-order logic (SO) over $\mathcal{M}$ and SC, denoted by SO(SC, $\mathcal{M}$), is defined as follows:*

- *Any atomic FO formula in the language of $\mathcal{M}$ is an atomic SO(SC, $\mathcal{M}$).*
- *If S is a variable that ranges over a set of elements of domain and t is a term in the language of Ω and SC in first-order form, then the expression $t \in S$ (also written $S(t)$) is an atomic SO(SC, $\mathcal{M}$).*
- *If $R \in SC$ is such a k-ary relation and $t_1, ..., t_k$ are first-order terms then the expression $R(t_1, ..., t_k)$ is an atomic SO(SC, $\mathcal{M}$).*
- *Formulas of SO(SC, $\mathcal{M}$) are closed under the Boolean connectives ($\wedge, \vee$, and $\neg$).*
- *If φ is a SO(SC, $\mathcal{M}$) formula and S is a relation symbol not in SC, then the following: $\exists x\varphi$, $\forall x\varphi$, $\exists S\varphi$, $\forall S\varphi$ are SO(SC, $\mathcal{M}$) formulas.*

The active-domain semantics of second-order logic, denoted by $\mathrm{SO}_{act}(SC, \mathcal{M})$, are those formulas in which all first-order, and second-order quantifiers range over the active domain.

Our goal is to study $\mathrm{SO}(SC, \mathcal{M})$ and investigate its applications to query languages for constraint complex value databases. Like in the first-order logic formalism, the expressive power and query evaluation issues need to be solved by new techniques and their solutions depend heavily on the model-theoretic properties of the underlying structure $\mathcal{M}$. The fundamental complex value structure and second-order logic also greatly impact on these solutions.

Constraint CV databases

We briefly describe three paradigms that can be used for querying constraint CV databases.

3.3 Algebra Queries

The first paradigm adopts the classical relational algebra approach which provides some algebraic operations for manipulating constraint tuples. In this subsection, we give a general definition of the constraint algebra operators. Let r be a relation of sort τ. $\varphi = t_1 \vee \cdots \vee t_n$ is the constraint formula which corresponds to r. The output of the selection operation is the conjunction of the constraint tuples and the selection condition. That is,

$$\sigma_\gamma \varphi = \bigvee\nolimits_{1 \le i \le n} (t_i \wedge \gamma)$$

The selection condition γ is of the form $x_i = d$, $x_i = x_j$, $x_i \in x_j$ or $x_i = x_j.C$, where d is a constant, and τ_j is a tuple sort of x_j with a C field.

When applying projection operation, we use existential quantifier to eliminate required variables from each constraint tuples. For example, let $X = \{x_1, ..., x_k\}$ be the set of variables in φ, let $x_j \in X$, and let $Y = X - \{x_j\}$.

$$\pi_Y \varphi = \bigvee\nolimits_{1 \le i \le n} \hat{t_i}$$

where $\hat{t_i}$ is semantically equivalent to $\exists x_j t_i$.

The join operation pairs each constraint tuple from two relations [11].

3.4 Calculus Queries

In the logic paradigm, we denote CALC^{cv}(SC, $\mathcal{M}$) as the set of all CALC^{cv} formulas in the language that contains all symbols of SC and $\mathcal{M}$. That is, CALC^{cv}(SC, $\mathcal{M}$) formulas are built up from the atomic positive literal $R(t)$, $t = t^{'}$, $t \in t^{'}$, or $t \subseteq t^{'}$ and $\mathcal{M}$ formulas by using Boolean connectives $\vee$, $\wedge$, $\neg$, and quantifiers $\forall$, $\exists$.

We refer to the above syntactic query languages as complex value calculus with $\mathcal{M}$ constraints. This will be denoted by CALC^{cv} + $\mathcal{M}$. When $\mathcal{M}$ is $(+, -, 0, 1, <)$ or $(+, *, 0, 1, <)$ we use standard abbreviations CALC^{cv} + LIN and CALC^{cv} + POLY. For a structure $\mathcal{M}$ and a SC-instance $\mathbf{I}$, the notion of $(\mathcal{M}, \mathbf{I}) \models \varphi$ is defined in a standard way for CALC^{cv}(SC, $\mathcal{M}$) formulas. If $\mathcal{M}$ is understood, we write $\mathbf{I} \models \varphi$.

Example 2. Let $\mathcal{M} = (\mathsf{R}, +, -, 0, 1, <)$. The following calculus query applies to schema $SC = \{R, S\}$ with sort $\tau_R = \tau_S = [\mathsf{R}, \{\mathsf{R}\}]$.

$$\varphi \equiv \exists v R(x, u) \wedge S(y, v) \wedge x \in u \wedge y \in v \wedge x + y < 10$$

It defines a subset of the join with the condition that in join-able tuples (x, u) and (y, v), x and y must be a member of the second component u and v respectively and x plus y is less than 10.

3.5 Datalog Queries

The deductive paradigm provides logic programming style for reasoning query results. It is a rule-based language. A rule is an expression of the form $p(x_1, \cdots, x_k)$

$\leftarrow L_1, \cdots, L_n$ where the head p is a derived predicate, and each L_i of the body is a literal. A Datalog query is a pair (P, q) where P is a finite set of rules, and q is a derived relation.

Example 3. Let $\tau_p = \tau_t = [\mathsf{R}]$, $\tau_r = \tau_s = [\mathsf{R}, \{\mathsf{R}\}]$ and $\tau_q = [\mathsf{R}, \mathsf{R}]$. The following is a constraint Datalog query.

$$\begin{aligned} s(x,z) &\leftarrow r(x,y) \wedge z \subseteq y \wedge 5 \notin z \\ q(x,v) &\leftarrow s(x,z) \wedge v = count(z) \\ t(x) &\leftarrow p(x) \vee (q(x,c) \wedge x + c < 100) \end{aligned}$$

A stratification of a program P is a partition $P_1, \cdots, P_n$ of the program such that no relation symbol R that is negated in a P_i is a derived relation in any P_j with $j \geq i$.

A program is stratified, if there is a stratification for it. The output of a stratified Datalog program query is called the perfect model.

4 Expressive Power

In the literature, the techniques used for analyzing expressive power of constraint databases are normally presented in the form of *collapse results* [4,3,19]. The purpose of these techniques is to reduce several important questions over constraint databases or embedded finite models to the classical finite model theory setting [2]. The focus is on expressive power of generic queries. We first summarize the definitions of collapse results in first-order logic from the literature as follows. Then we will adopt these techniques to extend them to the constraint CV model and investigate the issue of what properties of the underlying structure will impact the results.

Definition 7. *[4] We say that a structure $\mathcal{M}$ admits:*

- *natural-active collapse if FO$(SC, \mathcal{M})$ = FO$_{act}(SC, \mathcal{M})$ for any SC;*
- *active generic collapse if, for any SC, the classes of order-generic queries in FO$_{act}(SC, \mathcal{M})$ and FO$_{act}(SC, <)$ are the same;*
- *natural generic collapse if, for any SC, the classes of order-generic queries in FO$(SC, \mathcal{M})$ and FO$(SC, <)$ are the same.*

Embedded finite CV model

Establishing the natural-active collapse is difficult for the second-order logic. To overcome this problem, we consider a fragment of $SO(SC, \mathcal{M})$ in which all second-order variable were produced by FO formulas by induction. We denote this fragment as $\tilde{SO}$.

Our first goal is to show the following.

Theorem 1. *(**Natural-Active Collapse**) Let $\mathcal{M} =< \mathcal{U}, \Omega >$ be an o-minimal structure that admits quantifier elimination. Then it admits a weaker form of the natural-active collapse in the setting of embedded finite CV model, i.e., $\tilde{SO}(SC, \mathcal{M})$= $SO_{act}(SC, \mathcal{M})$ for any SC.*

Proof Sketch. Suppose the input formula is φ. In the second-order logic, we need to consider an sub-formula $\exists S \alpha(\boldsymbol{x})$ case in addition to all cases in first-order logic. As S is a set variable not in SC, it can be produced by FO formulas by induction. Those FO formulas can be expressed by $\mathrm{FO}_{act}(SC, \mathcal{M})$ formulas based on o-minimal structure property. Therefore there is an $\mathrm{SO}_{act}(SC, \mathcal{M})$ formula which is equivalent to φ. □

We then present the following result.

Theorem 2. ***(Active-generic collapse)*** *The active generic collapse holds over every structure $\mathcal{M}$ and complex value relational signature SC for second-order logic. That is every order-generic query definable in $SO_{act}(SC, \mathcal{M})$ is definable in $SO_{act}(SC)$.*

Proof Sketch. The main ideas of the proof follows the proof for first-order logic, by establishing the Ramsey property in the context of complex values. The proof is by induction on the formulas. The only issues we should consider are that of second-order quantification and complex values. We assume that every atomic sub-formula is an $SO_{act}(SC)$ formula or an $\mathrm{SO}(\mathcal{M})$. In regards to $SO_{act}(SC)$ formula, we need to consider three cases: (1) $\exists S\ \varphi$, (2) $x \in S$ and (3) $x \subseteq S$. For case (1), Let $\varphi(\boldsymbol{x}) = \exists S \in adom\ \varphi_1(S, \boldsymbol{x})$. By the hypothesis. find $T \subseteq X$ and $\psi_1(S, \boldsymbol{x})$ such that for any database D and $\boldsymbol{a}$ over T and any $Y \subset T$ we have $D \models \varphi_1(Y, \boldsymbol{a}) \leftrightarrow \psi_1(Y, \boldsymbol{a})$.

In regards to cases (2) and (3), there is no need to change the formula or find a subset. The sub-formula $\mathrm{SO}(\mathcal{M})$ has the Ramsey property [2]. □

Constraint CV databases

We now study the expressive power of standard constraint CV query languages such as CALC^{cv}+POLY.

Theorem 3. *The active generic collapse holds over every structure $\mathcal{M}$ for complex value logic, $CALC^{cv}$. That is, every order-generic query definable in $CALC_{act}(SC, \mathcal{M})$ is definable in $CALC_{act}(SC)$.*

Proof Sketch. $CALC^{cv}$ is a many-sorted calculus which is formed from a standard first-order logic. However, it facilitates *set* variables and has a second-order flavor. The key features: second-order quantification and existential and subset predicates have been proved in Theorem 2. Therefore, this is a direct consequence of Theorem 2. □

In the standard constraint databases, we sometimes want to write queries against a linear constraint input database in FO + POLY. It is known that FO + POLY has more expressive power than FO + LIN although FO + POLY has more costly evaluation procedures. One sometimes may want to use FO + POLY to write queries against semi-linear sets. Similarly, we may want to write queries against a linear constraint input database or a polynomial constraint database in a more expressive higher order language.

The fundamental topological connectivity property is important in many applications of constraint databases. As standard query languages for constraint databases lack the power to express connectivity properties [4], researchers attempted to enrich query languages by adding some extra functions, like transitive closure or fix-point operators. However, this extension may cause closure property fail for the extended languages. In [20], authors add topological connectivity property to the first-order constraint query languages and obtain new languages which are closed. However, this extension may cause the language at expense of high evaluation cost. The alternative approach is to adopt a higher-order query language to overcome this deficiency.

Definition 8. *Given a structure $\mathcal{M} =< U, \Omega >$, a set of complex values $O \subseteq U^\tau$ with complex value sort τ is called $\mathcal{M}$-definable if there exists a formula $\varphi(x)$ in the language of $\mathcal{M}$ such that $O = \{o \in U^\tau \mid SO(SC, \mathcal{M}) \models \varphi(o)\}$.*

We show that *connectivity* is definable in SO(SC, $\mathcal{M}$) where input database SC contains just an $\mathcal{M}$-definable set $S \subseteq \mathcal{R}^k$.

Proposition 1. *Some fundamental topological queries such as connectivity, having exactly one hole and having exactly k connected components, etc., are definable in $SO(SC, \mathcal{M})$.*

Theorem 4. *Datalog$^{cv,\neg}$ + POLY is closed; that is, on an $\mathcal{M}$- definable complex value constraint databases, an Datalog^{cv} + POLY query produces an $\mathcal{M}$-definable complex value set.*

Proof Sketch. A query is expressible in Datalog^{cv} with stratified negation if and only if it is expressible in $CALC^{cv}$ [13]. We know that SO$(SC, \mathcal{M})$ is closed under Boolean connectives. As $CALC^{cv} \subset SO$, $CALC^{cv}(SC, \mathcal{M})$ is also closed under Boolean connectives. Therefore $CALC^{cv}(SC, \mathcal{M})$ is closed for any input $\mathcal{M}$- definable complex value constraint databases. Accordingly, Datalog$^{cv,\neg}$ + POLY is closed on an $\mathcal{M}$- definable complex value constraint databases. □

While we expect to develop query languages having expressive power that can express most desirable properties, low complexity and easy query evaluation are major concerns in language design. The class of *elementary queries* is the complexity class of complex value query languages with power-set operator. As stated in [18], in terms of complexity, the characterizations of the calculus for constraint complex value databases are similar to the case of the classical complex values. However, we do expect a higher-order constraint query language can be effective and without too high complexity.

We present the following observation.

Theorem 5. *Every generic query in Inflationary Datalog$^{cv,\neg}_{act}(SC, \mathcal{M})$ without power-set operator can be evaluated in polynomial hierarchy time given that every generic sentence in $\mathcal{M}$ can be evaluated in AC^0/poly. .*

Proof Sketch. All operations in CALC^{cv} and accordingly in Datalog$^{cv,\neg}_{act}$ can be computed in polynomial time in the size of their arguments except for power-set which takes exponential time. □

5 Finite and Infinite Query Safety

In the previous sections, we proposed several paradigms of query languages for constraint complex value databases. As in classical databases, we normally require that queries return *finite* outputs on embedded finite models. In this section, we consider this safety issue in the context of embedded finite complex value models and constraint complex value databases. For the former, we consider whether there is a procedure that takes as input an SO$(SC, \mathcal{M})$ and determine it is safe or not. For the problem of query safety over constraint complex value databases, we show that whether it can preserve certain geometric properties in the *complex value* output.

Embedded finite CV model
In (monadic) second-order logic, some variables are quantified over subsets of the domain.

The output of a SO$(SC, \mathcal{M})$ formula $\varphi(x)$ on a finite complex value SC-structure D is $\varphi(D) \overset{\text{def}}{=} \{a \in \mathcal{U}^{\tau} \mid D \models \varphi(a)\}$, where $\mathcal{U}^{\tau}$ denotes the set of complex value constants with sort τ.

Definition 9. *An MSO(SC, $\mathcal{M}$) formula $\varphi(x)$ is safe on a finite complex value SC-structure D if $\varphi(D)$ is finite. A formula is safe if it is safe on every finite complex value structure.*

It is well known that the safety problem is undecidable even for a simple structure $\mathcal{M} =< \mathcal{U}, \emptyset >$, and a simple formula, φ, an FO$_{act}(SC)$ formula. However, as in first-order logic, we can find a procedure to identify a recursive subset of safe formulas that capture the query class that returns finite complex value outputs and every formula with this property belongs to this query class.

We now turn to the development of a syntactic condition that ensures calculus queries to be evaluable in closed-form. Our general approach to defining safe queries over embedded finite CV models is to check each input constraint subformula whether it contains a not-allowed negative constraint. For example, a negative constraint in $\exists y R(y) \wedge \neg(x^2 = y)$ will cause the formula produce infinite output.

Intuitively, if a query is in range-restricted form, then we can find an algebraic formula f which plays the role of giving an upper bound for the output of the query [10]. That means each *set* variable is range-restricted, i.e., the values assigned to set variables are finite.

As in [9], we can develop a procedure which compute the set of safe-range variables of a formula. In the procedure, we need to consider three key points: (1) $t \in S \wedge \phi$, (2) $R \subseteq S \wedge \phi$, and (3) $\neg C$, C is a constraint. In order to make the formula safe, the variables in the above three cases should be range-restricted.

Example 4. Let $\tau_p = \tau_q = [\{R\}, R]$. $p(x, y) \overset{\text{def}}{=} (x = \{x_1 \mid \phi\} \wedge y = 10)$, where $\phi \equiv (0 \leq x_1 \leq 10) \wedge x_1^2 = y$. Consider the formula

$$\psi = p(x, y) \wedge \exists t(u = x \wedge \neg q(u, t) \wedge t \in u).$$

$free(\psi) = \{x, y, u\}$ and all variables are bounded. So ψ is a safe formula.

Example 5. Let $\tau_p = [\{R\}, R]$; $\tau_s = [\{R\}, \{R\}]$. Consider the formula

$$\varphi = \exists x(\neg p(x,y) \wedge s(u,z)), \text{ where}$$
$$p(x,y) \overset{\text{def}}{=} (x = \{x_1 \mid \phi\} \wedge y = 10),\ \phi = (0 \leq x_1 \leq 10);$$
$$s(u,z) \overset{\text{def}}{=} u \subset z \wedge z = \{1,2,3\}.$$

s contains $u \subset z$ and $z = \{1,2,3\}$ which are both safe formulas. So s is a safe formula. $\exists x(\neg p(x,y)$ is not a safe formula as it contains unbounded variable y. Therefore φ is not a safe formula.

Constraint CV databases. In regards to the safety problem for constraint complex value databases, we consider two issues. The first issue is whether we still can express the output of our constraint CV query in terms of the class of constraints used to define the input database. For example, when the input database is a polynomial constraint database expressed in FO + POLY, then we intend to use CALC^{cv} + POLY to query input database. The output of an CALC^{cv} + POLY query may fail to be expressed in FO + POLY. Although we get extra expressive power for our query languages, we encounter the safety problem.

The second issue is whether the output of constraint CV query preserve certain geometry properties of regions in $\mathcal{R}^k$. Is there effective syntax for the class of constraint CV queries which preserve some geometry property? Can this problem be reduced to finite query safety for embedded finite CV models? It is known that FO cannot express topological connectivity or transitive closure, we investigate whether SO gains enough power to express such topological queries.

Theorem 6. *There is an effective syntax for the class of queries definable in $SO(SC, \mathcal{M})$ preserving topological connectivity.*

Proof sketch. This problem can be reduced to graph connectivity [2]. As CALC^{cv} can express the transitive closure of a binary relation, $SO(SC, \mathcal{M})$ can test graph connectivity. Therefore the class of safe $SO(SC, \mathcal{M})$ preserving topological connectivity has effective syntax. □

As FO + POLY ⊂ SO + POLY, a union of conjunctive SO + POLY queries preserves the property of being a convex polytopes. In regards to other topological properties, we need to do further investigation.

Finally, we show the following.

Theorem 7. *Safe CALC^{cv} + POLY, ALG^{cv}+ POLY and Inflationary Datalog$^{cv,\neg}$ +POLY have equivalent expressive power.*

Proof sketch. In the context of the complex value model, Safe CALC^{cv}, ALG^{cv} and Inflationary Datalog$^{cv,\neg}$ have equivalent expressive power. This result still holds by adding polynomial constraints.

6 Conclusion

The constraints provide a sound mathematical framework to define both data models and query languages. We investigated the expressive power of various query languages over constraint complex value databases. We showed that natural-active collapse and active-generic collapse carry over to second-order for structures with o-minimality and any relational signature in the complex value model. However, under what conditions natural-generic collapse will hold needs to be further investigated.

Some of the classical topological queries, for example, connectivity, which cannot be expressed in FO(SC, $\mathcal{M}$) can easily be expressed in SO($SC, \mathcal{M}$), given that $\mathcal{M}$ is an o-minimal expansion of the real field $\mathcal{R}$. We considered a syntactic condition that ensures constraint complex value calculus queries to be evaluable in closed-form in the embedded finite model. The main results of the paper could be helpful for designing query languages over advanced constraint databases based on an extended data model.

Acknowledgment. The author would like to thank Byron Choi for helpful discussion and suggesting numerous improvements.

References

1. Kanellakis, P., Kuper, G., Revesz, P.: Constraint query languages. Journal of Computer and System Sciences 51(1), 26–52 (1995)
2. Libkin, L.: Embedded finite models and constraint databases. In: Book:Finite Model Theory and its Applications. Springer, Heidelberg (2007)
3. Libkin, L.: Query languages with arithmetic and constraint databases. SIGACT News, Database Theory Column, 41–50 (1999)
4. Benedikt, M., Dong, G., Libkin, L., Wong, L.: Relational expressive power of constraint query languages. Journal of the ACM 45, 1–34 (1998)
5. Ebbinghaus, H.D., Flum, J.: Finite Model Theory. Springer, Heidelberg (1995)
6. Libkin, L.: Elements of Finite Model Theory. Springer, Heidelberg (2004)
7. Otto, M., den Bussche, J.V.: First-order queries on databases embedded in an infinite structure. Information Processing Letters 60, 37–41 (1996)
8. Benedikt, M., Libkin, L.: Relational queries over interpreted structures. Journal of the ACM 47, 644–680 (2000)
9. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
10. Benedikt, M., Libkin, L.: Safe constraint queries. SIAM Journal on Computing 29, 1652–1682 (2000)
11. Revesz, P.: Safe query languages for constraint databases. Transactions on Database Systems 23, 58–99 (1998)
12. Stolboushkin, A., Taitslin, M.: Safe stratified datalog with integer order does not have syntax. ACM Transactions on Database Systems 23(1), 100–109 (1998)
13. Abiteboul, S., Beeri, C.: The power of languages for the manipulation of complex values. The Very Large Data Bases Journal 4, 727–794 (1995)
14. Avron, A.: Constructibility and decidability versus domain independence and absoluteness. Theoretical Computer Science 394, 144–158 (2008)

15. Suciu, D.: Domain-independent queries on databases with external functions. Theoretical Computer Science 190(2), 279–315 (1998)
16. Liu, H.C., Yu, J.: Safe database queries with external functions. In: Proceedings of International Database Engineering and Applications Symposium, Montreal, Canada, pp. 260–269 (1999)
17. Liu, H.C., Yu, J., Liang, W.: Safety, domain independence and translation of complex value database queries. Information Sciences 178, 2507–2533 (2008)
18. Grumbach, S., Su, J.: Dense-order constraint databases. In: Proceedings of ACM Symposium on Principles of Database Systems, pp. 66–77 (1995)
19. Libkin, L.: A collapse result for constraint queries over structures of small degree. Information Processing Letter 86, 277–281 (2003)
20. Benedikt, M., Grohe, M., Libkin, L., Segoufin, L.: Reachability and connectivity queries in constraint databases. Journal of Computer and System Sciences 66, 169–206 (2003)

Scaling Up Query Allocation in the Presence of Autonomous Participants*

Jorge-Arnulfo Quiané-Ruiz[1], Philippe Lamarre[2], Sylvie Cazalens[2], and Patrick Valduriez[3]

[1] Saarland University, Saarbruecken, Germany
`Jorge.Quiane@cs.uni-saarland.de`
[2] LINA, University of Nantes, France
{`Philippe.Lamarre`,`Sylvie.Cazalens`}`@univ-nantes.fr`
[3] INRIA and LIRMM, Montpellier, France
`Patrick.Valduriez@inria.fr`

Abstract. In large-scale, heterogeneous information systems, mediators are widely used for query processing and the good operation of a system strongly depends on the way the mediator allocates queries. On the other hand, it is well known that a single mediator is a potential scalability and performance bottleneck as well as a single point of failure. Thus, multiple mediators should perform the query allocation process. This task is challenging in large-scale systems because participants typically have special interests that are not performance-related. Mediators should satisfy participants interests as if there was a single mediator in the system — i.e., with no, or almost no, additional network traffic. In this paper, we propose a virtual money-based query allocation method, called VM_bQA, to perform query allocation in the presence of multiple mediators and autonomous participants. A key feature of VM_bQA is that it allows a system to scale up to several mediators with no additional network cost. The results show that VM_bQA significantly outperforms baseline methods from both satisfaction and performance points of view.

1 Introduction

In the last decade there has been a considerable increase in computing resources requirements in different research fields as well as in the industry. These needs have motivated the development of new distributed systems allowing users to share data, services, or computing resources using Internet as a large virtual computer. In such large-scale and heterogeneous information systems, mediators are commonly used to perform query allocation. BOINC and distributed.net are only one example of large-scale information systems using a mediator. A particularity of this kind of systems is that participants (consumers and providers) are autonomous in the sense that they may leave and join the mediator at will,

* Work partially funded by the Dataring project of the *Agence Nationale de la Recherche* and the Epilog project of the PREDIT program of the french ministry of transportation.

but also, that they may have special interests (*intentions*) for some queries[1] and other participants. For example in BOINC, while a consumer may want to receive results from highly reputed providers, a provider might want to perform queries for some preferred projects.

In such systems, participants usually have certain non-performance-related expectations, which reflect their *intentions* to allocate and perform queries in the long-run, with respect to the mediator. These participants intentions are clearly illustrated by Google AdWords, which proposes relevant commercial providers to consumers and relevant consumers to commercial providers according to some keywords of their interest. In this context, the mediator must satisfy participants (i.e., to fill their intentions), because dissatisfaction may lead participants to leave the system [6]. Participants departure in turn cause some loss of system functionality and capacity to perform queries. On the other hand, it is also well known that in large-scale systems a mediator can quickly become a single point of failure as well as a potential performance and scalability bottleneck. This is why it is crucial to have several mediators performing query allocation. However, in a multi-mediator system, a mediator can no longer compute solely the satisfaction of providers as their satisfaction also depends on query allocations made by other mediators. Thus, when allocating a query, either a mediator should keep informed all other mediators of the mediation results to update the satisfaction of participants or a provider should frequently inform each mediator of their current satisfaction. Nevertheless, network traffic increases significantly in both cases, which hurts system performance. A natural way to avoid such a traffic overhead between mediators is that providers be responsible of their satisfaction and express their intentions only. But, some providers can lie about their intentions so as to monopolize queries while providers revealing real intentions can suffer from query starvation.

To overcome this problem, we propose the use of *virtual money* to scale query allocation up to several mediators while keeping providers satisfaction high. When using virtual money, providers no longer show their intentions directly, but rather bid on the incoming queries. As a result, the mediators no longer consider the providers satisfaction but only their bids. On the other side, satisfaction still depends on the work of the different mediators and hence it is still important to satisfy participants. The query allocation problem in this context is challenging for several reasons. First, a mediator cannot select providers based only on their bids because of consumers intentions. Second, a mediator must allocate each incoming query even if some cases providers do not desire to perform them (*imposition case*), which makes still more difficult the task of satisfying participants in the long-run. Third, invoicing providers is a difficult task because some providers can be imposed a query, which in turn implies dissatisfy them. Fourth, when bidding for queries, a provider must efficiently take into account the bids it made for other queries at different mediators.

[1] We use the word "query" in the general sense of service request in information systems, thus with a more general meaning than query in databases.

Contributions. The main contributions of this paper are as follows:

– We make precise the flow of virtual money within the whole system and state the network cost of using virtual money as a means of regulation (Section 3).
– We propose a *Virtual Money-based Query Allocation* (VM_bQA) method to allocate queries by considering both consumers intentions and providers bids. A salient feature of VM_bQA is that it may impose queries to some providers without significantly impacting their satisfaction in the long run (Section 4).
– We define a bidding procedure that considers the satisfaction, utilization, preferences, and virtual money balance of providers. But also, as a part of such bidding procedure, we propose three strategies to bid in multi-mediator systems (Section 5).
– We analytically demonstrate that VM_bQA allows a system to scale up to several mediators with no additional network cost with respect to a system with a single mediator using VM_bQA (Section 6).
– We experimentally demonstrate that VM_bQA significantly outperforms baseline methods from both the satisfaction and performance points of view. We also show that VM_bQA gracefully scales up in terms of number of mediators and participants (Section 7).

1.1 Related Work

Several approaches based on microeconomics have been proposed to deal with resource or query allocation [1,2,5,9]. Mariposa [9] pioneered the use of a market approach for dealing with the query allocation problem. Nevertheless, its query allocation procedure is simple and limited. Consumers cannot freely express their intentions since it is inherently assumed that they are just interested in response times and low prices for acquiring services. In [4], the authors proposed some optimization algorithms for *buying* and *selling* query answers, and for the negotiation strategy. However, this way of dealing with subqueries optimization is orthogonal to our proposal and one may combine them to improve performance. Auctions are widely recognized as a way to manage negotiation among participants. Several kinds of auction mechanisms exist [7,11]. Our approach looks like the generalized Vickrey auction, but it pushes generalization further since it considers participants intentions. In the field of distributed rational decision making [7], most of the processes are individually rational, which is not relevant when participants may be imposed a query, which implies having lower utilities. In work [6], we proposed a query allocation framework that considers participants satisfaction and system performance, but we assumed mono-mediator systems. In this work, we complete the puzzle by allowing the proposed query allocation framework in [6] to gracefully scale up to as many mediators as required without any loss in participants satisfaction nor system performance.

2 Problem Definition

We assume a distributed system to be a set $\mathcal{I}$ of participants registered in a mediator of the system, forming then a *Virtual Organization* (VO). Each participant of a VO can play one or more of the following roles: *consumers* which send queries ; *providers* which answer queries ; and *mediators* which allocate queries to providers. The set of participants playing the role of consumer, resp. provider and mediator, is noted C, resp. P and M, with $C \cup P \cup M = \mathcal{I}$. We assume that a VO may be *x-redundant* [12], i.e. there are several mediators acting as a single one by behaving cooperatively. We formally define a x-redundant VO as follows.

Definition 1. x-redundant VO. *A VO is said to be $x - redundant$ if and only if there is a set M ($x > 1$, with $||M|| = x$) of participants playing the role of mediator and each provider in P is connected to each mediator in M.*

Notice that, the above definition does not specify the way mediators are connected among them. This is because the inter-mediator network should be as independent as possible of the query allocation process. As a result, one can assume any kind of inter-mediation network without caring of the query allocation method used by mediators.

Providers have different *capacity* and hence some providers can treat more queries per time unit than others. The *utilization* of a provider $p \in P$ at a given time t, $\mathcal{U}_t(p)$, is defined as the query load of p regarding its capacity. In other words, function $\mathcal{U}_t(p)$ denotes the total cost of the queries that have been allocated to p and have not already been treated at time t. Consumers send their queries to a single mediator in a format abstracted as a triple $q = < c, d, n >$ such that $q.c \in C$ is the identifier of the query initiator (the consumer) ; $q.d$ is the description of the task to be done, and; $q.n \in N^*$ is the number of providers to which consumer $q.c$ wishes to allocate its query. Indeed, a consumer may want to query different providers, in particular in the case they can provide different answers. Mediators should allocate queries so that good system performance is obtained. If performance (such as load balancing and response time) is linked to clear notions in distributed systems, satisfaction is less usual. However, satisfaction has a deep impact on a system general behavior, particularly when the participants are autonomous. In [6], we proposed a model that characterizes autonomous participants by formally defining the adequation, satisfaction, and other notions related to participant's intentions. We briefly present below the satisfaction notions required in this paper.

Because of autonomy, we assume a consumer to be interested in the way its query is treated. Those intentions are represented as a vector $\overrightarrow{CI}_q$ such that $\overrightarrow{CI}_q[p]$ represents the consumer intention towards provider p. Values are, by convention, in the range $[-1..1]$. Consumer satisfaction, $\delta_s(c)$, allows a consumer to evaluate if a mediator allocates queries with respect to its intentions. Formally,

$$\delta_s(c) = \frac{1}{||IQ_c^k||} \sum_{q \in IQ_c^k} \frac{1}{n} \Big(\sum_{p \in \widehat{P_q}} (\overrightarrow{CI}_q[p] + 1) \Big/ 2 \Big) \qquad (1)$$

where n stands for $q.n$, IQ_c^k denotes the set of k last queries issued by c, $\widehat{P_q}$ is the set of providers that performed q. Similarly, the provider satisfaction allows a provider to evaluate if a mediator givies it queries according to its expectations.

Thus, as for consumers, a provider is not satisfied when it does not get what it expects. A provider $p \in P$ computes its satisfaction, $\delta_s(p)$, as follows,

$$\delta_s(p) = \begin{vmatrix} \frac{1}{||SQ_p^k||}\Big(\sum_{q \in SQ_p^k} (\overrightarrow{PPI}_p[q] + 1) \Big/ 2 \Big) & if\ SQ_p^k \neq \emptyset \\ 0 & otherwise \end{vmatrix} \tag{2}$$

where SQ_p^k denotes the set of queries performed by p and $\overrightarrow{PPI}_p[q]$ is the intention expressed by p towards query q, whose values are in $[-1..1]$. Notice that the way a participant computes its intentions is out of scope of this paper and thus we simply assume that a participant uses, for example, the techniques proposed in [6] to do so.

Let P_q denote the set of providers registered to mediator m, which does not appear in the notation for simplicity, and that can deal with an incoming query q. Many matchmaking techniques have been proposed in the literature and thus we simply assume there is one in the system to find out the set P_q. The allocation of a query q is denoted by vector $\overrightarrow{Alloc}_q$ of length N (with $N = ||P_q||$) such that,

$$\forall p \in P_q,\ \overrightarrow{Alloc}[p] = \begin{vmatrix} 1 & \text{if } p \text{ gets the request} \\ 0 & \text{otherwise} \end{vmatrix}$$

Query Allocation Problem. Given a x-redundant VO with a set M of mediators confronted to autonomous participants, $\forall m \in M$, m should allocate each incoming query q to a set $\widehat{P_q}$ such that short response times and participants satisfaction are ensured in the long-run, with a low network cost independently of the number of mediators in the x-redundant VO.

3 VM$_b$QA Overview: The Flow of Virtual Money

Overall, our approach consists in appropriately defining a mediation process for a single mediator and in defining the way the virtual money should circulate within the whole system to support multiple mediators. This section focuses on the latter point. The general query allocation architecture is depicted in Figure 1. Given an incoming query q to any mediator $m \in M$, m asks the consumer $q.c$ for its intentions and also asks set P_q of relevant providers for their bid. Having obtained this information, the mediator must allocate q to a set $\widehat{P_q}$ of providers. Finally, as we consider that providers have to "pay" for performing or receiving queries, the mediator invoices relevant providers after the allocation of q. In all this process, the "virtual" money we use is purely virtual and is totally disconnected from the real money we use in current life. We could speak of tokens or jetons as well. This point has to be stressed upon for two main reasons. First, we do not focus on any particular business model: we only use the virtual money as a means of regulation. Indeed, after a consumer has decided which providers it chooses, it might give *real money* to them because it uses their services. However, this point is far beyond the focus of this paper. Second, when using real money, one can assume that participants get money from elsewhere.

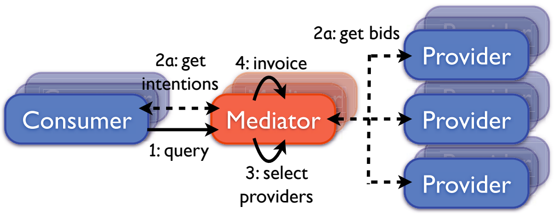


Fig. 1. General query allocation architecture

When dealing with virtual money, one can no longer make such assumptions: one must precise the way in which the virtual money circulates within the system.

To stress the way in which virtual money circulates within the system is a difficult task since it is a macroeconomic concern and hence one must have a clear idea of the global system behavior. Furthermore, the policy used to regulate the virtual money flow also depends on the query allocation method. Hence, we adopt a simple solution. Providers spend and earn virtual money through a mediator only. On the one hand, they spend money by bidding on queries and to compensate other providers that have been imposed a query. On the other hand, they earn money when they are imposed a query that they do not desire to perform. Every time a provider has been allocated a query, has been imposed a query, or has been required to compensate an imposed provider, it is informed, by the concerned mediator, of the amount of virtual money it payed or won (in the case of imposition). Of course, a provider is completely responsible of its virtual money balance and hence no provider can spend the virtual money of another one. Therefore, a provider always has an exact mirror of its virtual money balance in local. In contrast to providers, mediators never looses money but tends to accumulate money coming from the providers in the course of time. Thus, in case a mediator has earned an amount of virtual money above a defined threshold, it distributes such an amount of virtual money to providers in an equitable way. For providers, this is another regular way of earning money.

In this paper, we assume that there exists a *trusted third-party* in the system that plays the role of bank. Several ways to implement the bank exist (using a DHT [8] for example), but this is well beyond the scope of this paper. Thus, for clarity, we omit the bank in the remainder of this paper when we talk about the virtual money balance of a provider. Indeed, the flow of virtual money requires some network messages. We state this cost in the following proposition.

Proposition 1. *Only* 3 *messages per query are required by any mediator to control the flow of virtual money with any provider.*

Proof (Sketch). First of all, we assume a *vickrey* auction to allocate queries and hence no message is required by a provider to discover the bids of other providers. Similarly, at first glance, a provider may require a network message to know its current virtual money balance so as to bid for queries. However, a mediator informs a provider of any change in its virtual money balance, thereby allowing a provider to always know its current virtual money balance. Thus, no network message is required by a provider to know its virtual money balance. Now, before a query mediation, two network messages are exchanged between a mediator and the bank in order to validate the bids of providers, i.e. to verify if they have enough virtual money that support their bids. After a query allocation, the concerned mediator sends another network message to the bank, which is in charge of invoicing providers. Therefore, a mediator requires 3 network messages only: 2 to validate provider bids and 1 to invoice providers. □

4 VM$_b$QA: Mediating Queries

To allocate a query, a mediator ask relevant providers to bid and stores such bids in vector $\overrightarrow{B}$. Intuitively, if a bid is positive, the higher it is, the more p wants to be allocated q. If it is negative on the other hand, the lower it is, the less p wants to treat q. Thus, one may consider only the bids of providers to select providers, such as in several other approaches [2,5]. Nevertheless, this is a provider-centric approach that can easily result into the consumers dissatisfaction. This is because the intentions of consumers are not considered at all as well as some of the incoming queries may stay untreated (because no provider wants to treat them). Therefore, in VM_bQA, a mediator: *(i)* directly considers the intentions of consumers, and *(ii) imposes* a query when not enough providers desire to perform it. With this in mind, we introduce the *providers level* notion, denoted by vector $\overrightarrow{L}$, which results from merging the consumers intentions with the providers bids. We formally define this in Definition 2 where parameter ω, whose values are in the interval [0..1], ensures the balance between consumers intentions and providers bids. If $\omega = 0$, only the consumers intentions are considered by the mediator, thus leading to providers dissatisfaction. Conversely, if $\omega = 1$, the mediator only considers bids, leading to consumers dissatisfaction. This is why the mediator should set this parameter ω according to the balance between both consumers and providers satisfaction that it wants to reach.

Definition 2. Provider Level

$$\overrightarrow{L}[p] = \left| \begin{array}{ll} (\overrightarrow{B}[p]+1)^{\omega} \times (\overrightarrow{CI}_q[p]+1)^{1-\omega} & if\ \overrightarrow{B}[p] \geq 0 \\ \\ -(-\overrightarrow{B}[p]+1)^{\omega} \times (\overrightarrow{CI}_q[p]+1)^{\omega-1} & otherwise \end{array} \right.$$

Given an incoming query q, a mediator ranks providers in accordance to their level, resulting in ranking vector $\overrightarrow{R}$. Intuitively, $\overrightarrow{R}[1] = p$ if and only if p is the best ranked, $\overrightarrow{R}[2]$ stands for the second best ranked and so on. Then, the mediator allocates q to the $min(n, N)$ best providers, i.e., $All\overrightarrow{oc}_q[p] = 1$ if and only if $\exists i$, $\overrightarrow{R}[i] = p$ and $i \leq min(n, N)$. After allocating query q, a mediator must invoice those providers that participated in the mediation of q. A natural strategy to invoice a provider that has been allocated q is that it pays what it bids (a.k.a. *first-price* invoice mechanism). However, this incites providers to shade their bids below their true value with the aim of maximizing their revenues (or satisfaction), while those providers revealing true bids may suffer from query starvation (or dissatisfaction). Therefore, we adopt a *second-price* invoice mechanism (a.k.a. *vickrey*), which has been proved to incentivize providers to reveal their true bid values [10]. However, in contrast to vickrey, we cannot directly compare providers bids because of consumers intentions. To overcome this difficulty, we introduce the *theoretical bid* notion, which corresponds to the amount a provider should bid for reaching a given level. Equation 3 formally states this notion, where $\omega \neq 0$ and $\alpha = 1$ if $l \geq 0$ or $\alpha = -1$ otherwise.

$$b^{th}(p,l) = \alpha \cdot max(((\alpha \times l)^{\frac{1}{\omega}}(\overrightarrow{CI}_q[p]+1)^{\frac{\alpha(\omega-1)}{\omega}} - 1), 0) \tag{3}$$

A mediator may then invoice providers by simply considering their theoretical bid. However, given the providers level definition, a mediator is confronted to the special case that q is allocated to a provider having a negative bid (i.e., that does not desire the query): we call this an imposition case, otherwise, we have a competition case. Obviously, being imposed a query q does not meet at all the expectations of a provider. To keep a imposed provider satisfied in the long run, we then distribute the cost of the imposition of query q on *all* the relevant providers. Having obtained a reward, an imposed provider is more likely, in the future, to obtain the queries it expects (because it has more virtual money) so leading to its satisfaction. Thus, in an imposition case, a mediator computes the partial bill of provider p regarding the imposition of provider p' as follows.

Definition 3. Provider Partial Invoice: the Imposition Case

$$bill_q(p,p') = \left| \begin{array}{ll} \dfrac{-b^{th}(p, \overrightarrow{L}[\overrightarrow{R}_q[q.n+2]])}{N_q} & if \;\; p \neq p' \\ b^{th}(p, \overrightarrow{L}[\overrightarrow{R}_q[q.n+1]]) - \frac{b^{th}(p, \overrightarrow{L}[\overrightarrow{R}_q[q.n+2]])}{N_q} & else \end{array} \right.$$

In a competition case, a provider p allocated q only owes the amount of its theoretical bid to reach the level of the best provider that has not been allocated the query. That is, p does not have to share the cost of selecting other providers. Formally, a mediator computes the partial bill of a provider p concerning the selection of a non-imposed provider p' as follows.

Definition 4. Provider Partial Invoice: the Competition Case

$$bill_q(p,p') = \left| \begin{array}{ll} b^{th}(p, \overrightarrow{L}[\overrightarrow{R}_q[q.n+1]]) & if \;\; p = p' \wedge \overrightarrow{B}[\overrightarrow{R}_q[q.n+1]] \geq 0 \wedge q.n < N_q \\ 0 & otherwise \end{array} \right.$$

Having the partial bills, a mediator simply bills a provider p with respect to the allocation of a query q by aggregating all its partial bills.

$$bill_q(p) = \sum_{p' \in \widehat{P_q}} bill_q(p,p') \tag{4}$$

Overall, a provider that has not been allocated a query never pays for it, except if another provider has been imposed. Moreover, the invoicing process we presented here never requires a mediator from a financial point of view.

5 VM$_b$QA: Bidding for Queries

A simple way for a provider to compute bids is to maintain a local bulletin board containing a billing rate for its resources, based on its preferences to perform queries (denoted by function $prf \in [-1..1]$). Then, a provider bid for getting a query can simply be the product of its current utilization by the billing rate, such as in [9]. However, in our case, the context is more complex because a provider must consider its current satisfaction and current virtual money balance (denoted by bal_p) in addition to its preferences and load. The reader may think that a provider may obtain its bid by computing the product of all these parameters. Nevertheless, this procedure can lead a provider to spend all, or almost all, its money on only one query. To avoid this, a provider should (i) offer at most only a defined percent of its current virtual money balance, denoted by constant c_0 whose values are in $]0..1]$, and (ii) weight its preferences and utilization using its current satisfaction. We thus formally define a provider bid, denoted by $bid_p(q)$, as in the following definition.

Definition 5. Provider Bid

$$bid_p(q) = \left| \begin{array}{ll} (prf_p(q)^{1-\delta_s(p)}) \times (1 - \mathcal{U}_p(t))^{\delta_s(p)} \times (bal_p \cdot c_0) & if(prf_p(q) > 0) \\ & \wedge (\mathcal{U}_p(t) < 1) \\ -\big((1 - prf_p(q) + 1)^{1-\delta_s(p)} \times (\mathcal{U}_p(t) + 1)^{\delta_s(p)}\big) \times c_1 & otherwise \end{array} \right.$$

The idea behind the above definition is that a provider always sets a positive bid when it desires to perform queries and it is not overutilized, otherwise it sets a negative bid. As an imposed provider is awarded according to its negative bid (see the previous Section), constant c_1 is set to its initial virtual money balance so that the award of an imposed provider be always less, or equal, than the amount it got when joining the system. Notice that, in traditional micro-economic-based mediation approaches, providers do not bid (or give a null bid) when they do not desire to perform a query. However, this does not allow them to express how much unpleasant it is for them to perform a query and how loaded they are.

Now, as in x-redundant VOs a provider receives queries from different mediators, it must pay special attention to its virtual money balance so that it never bids more. To deal with this, we propose 3 strategies: the *optimistic*, *preventive*, and *pessimistic* strategies. Before going to present these heuristics, let us say that after bidding for a query q, a provider p locally stores in vector $\overrightarrow{CB}_p$ its bid $bid_p(q)$ and removes such a bid from $\overrightarrow{CB}_p$ when it receives the invoice for q.

Optimistic Strategy. An optimistic provider assumes that it gets all those queries to which it bids positively and that it does not get those to which it expresses a negative bid. Thus, an optimistic provider p modifies its current virtual money balance after the bidding phase of a given query q as follows,

$$bal_p = \left| \begin{array}{ll} bal_p - bid_p(q) & if\ bid_p(q) > 0 \\ bal_p & else \end{array} \right. \tag{5}$$

and when p receives the final invoice, it sets its virtual money balance as follows,

$$bal_p = \left| \begin{array}{ll} bal_p + \overrightarrow{CB}_p[q] - bill_q(p) & \mathit{if}\ bid_p(q) > 0 \\ bal_p - bill_q(p) & \mathit{otherwise} \end{array} \right. \tag{6}$$

Preventive Strategy. A preventive provider assumes that it gets all those queries to which it bids, independently of its bid value. In other words, conversely to an optimistic provider, it also assumes that it gets those queries to which it made a negative bid. Thus, a preventive provider p modifies its current virtual money balance after the bidding phase of a given query q as follows,

$$bal_p = bal_p - bid_p(q) \tag{7}$$

and when p receives the final invoice, it sets its virtual money balance as follows,

$$bal_p = bal_p + \overrightarrow{CB}_p[q] - bill_q(p) \tag{8}$$

Pessimistic Strategy. In contrast to a preventive provider, a pessimistic provider assumes it never gets the queries to which it bids. Thus, a pessimistic provider p does not modifies its virtual money balance after bidding for queries. It therefore modifies its current virtual money balance when it receives the final invoice from the a mediator as follows.

$$bal_p = bal_p - bill_q(p) \tag{9}$$

6 VM$_b$QA: Communication Cost

In the following theorem, we state the communication cost in terms of number of network messages that should be transferred to perform a query.

Theorem 1. *The total number of transferred messages by VM_bQA to perform a query is* $3(N+2)+n$.

Proof. Given a query q and a set P_q of relevant providers, a mediator first asks for both the intentions of consumer $q.c$ and the bids of set P_q of relevant providers, which return such information to the mediator. The number of exchanged messages at this phase is $mssg_0 = 2N+2$. Once received such an information, the mediator verifies if the relevant providers have enough virtual money to support their bids. This requires $mssg_1 = 2$ messages between the mediator and the bank. Next, it computes the level of each relevant provider and ranks them according to their level. Having done this, the mediator invoices each relevant provider and informs them of the mediation result. It then waits for results from the n selected providers. The number of transferred messages at this phase is $mssg_2 = 1$ to invoice providers, $mssg_3 = N$ to inform providers, and $mssg_4 = n$ to receive results from selected providers. Finally, the mediator sends the results to consumer $q.c$, which implies one more network message, $mssg_5 = 1$. Thus, $Mssg = mssg_0 + mssg_1 + mssg_2 + mssg_3 + mssg_4 + mssg_5 = 3(N+2)+n$ total messages are exchanged to perform a query. □

The great advantage of VM_bQA is that it has no network cost when dealing with several mediators and continues to perform, from a satisfaction point of view, as in a mono-mediator VO. The following theorem demonstrates this by showing sthat VM_bQA allows a VO to scale up to as many mediators as desired with no loss in system performance.

Theorem 2. *VM_bQA always satisfies (i) consumers and (ii) providers in a x-redundant VO as in a mono-mediator VO with no additional network cost.*

Proof. Consider a x-redundant VO, denoted by S_{vo} and a mono-mediator VO, denoted by S_m, consisting of the same set of participants P. Consider also that the incoming queries in S_{vo} are the same as those arriving in S_m. We prove both (i) and (ii) by contradiction. (i) Assume to the contrary that, for the allocation of some query q, consumer $q.c$ is not equally satisfied by S_{vo} and S_m. If this is the case, we can know, by Equation 1, that S_{vo} allocated q to a set $\widehat{P_q}$ such that there exists at least a provider $p \in \widehat{P_q}' : p \notin \widehat{P_q}$, where $\widehat{P_q}'$ is the set of providers selected by S_m. Hence, we can know that the set of relevant providers found by S_{vo} is different from the set found by S_m. This implies that provider p is not connected to the mediator that allocated q in S_{vo}, which contradicts the definition of an x-redundant VO. (ii) Assume to the contrary that a provider $p \in P$ is not equally satisfied by S_{vo} and S_m. Then, by Equation 2, we can know that p did not perform the same set of queries in S_{vo} as in S_m. This means that p is not connected to all mediators in S_{vo} so as to receive all queries it can perform, which contradicts the definition of a x-redundant VO.

Finally, given the providers level definition, a mediator does not directly deal with providers satisfaction because it is up to a provider to manage its virtual money balance so as to be satisfied in the long-run (Definition 5). Thus, the mediator has no message to exchange among mediators to update providers satisfaction. As stated in proposition 1, 3 messages are required by a mediator to control the flow of virtual money. As mediators do not communicate after query allocations, this cost remains constant independently of the number of mediators in a VO. Therefore, no additional network message is required by a VO to scale up to several mediators. □

7 Experimental Validation

Our main objectives in this experimental validation are: *(i)* to evaluate how well VM_bQA selects and invoices providers, *(ii)* to analyze the impact of using virtual money as a means of regulation when performing query allocation, *(iii)* to analyze if VM_bQA satisfies participants in x-redundant VOs as well as in VOs with a single mediator, and *(iv)* to evaluate the performance of VM_bQA when dealing with x-redundant VOs.

7.1 Setup and Methodology

We run our experiments in a computer running Linux Ubuntu 4.0.3 with a Pentium IV processor of 3 GHz and 1 GB in RAM. The system consists of 200 consumers, 400 providers. We assume that consumers compute their intentions by considering only their preferences such as defined in [6]. Concerning a provider, we assume that it computes its bids as in Definition 5. Participants compute their satisfaction as presented in Section 2 and initializes their satisfaction with a value of 0.5, which evolves with their last 200 issued queries and

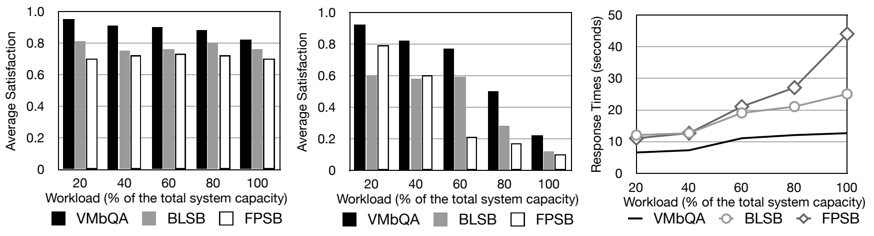
500 queries that have passed through providers (i.e. $k = 200$ for a consumer and $k = 500$ for a provider). Providers compute their bids following the optimistic strategy. We consider that queries arrive to the system in a *Poisson* distribution, as found in dynamic autonomous environments [3]. We implemented the bank on top of a DHT network (based on Chord [8]) and the time that the bank takes to validate providers bids and to invoice providers is included in the response time results we present here. We run each series of experiments 10 times and present the average results of all these experimentations.

To see the possible loss of performance that VM_bQA may have, from the provider point of view, we compare it with a *first-price sealed-bid* method ($FPSB$). $FPSB$ allocates queries to those providers having made the highest bids and invoices providers the amount of virtual money that they offered for queries. To study the efficiency of the way in which VM_bQA invoices providers, we compare it with a query allocation method that selects providers based on their level (such as VM_bQA does), but invoices them as $FPSB$ does. We call this query allocation method as *best-level sealed-bid* ($BLSB$). To validate VM_bQA, from a satisfaction point of view, we compare it with S_bQA, which has been shown to perform well in autonomous environments [6]. For this, we extend S_bQA to support several mediators using satisfaction as a means of regulation. Then, we proceed in two steps for the experiments. First, we aim at evaluating our mediation and invoicing mechanisms from both the satisfaction and performance points of view. To this end, we compare VM_bQA against the baseline methods: $FPSB$ and $BLSB$. Second, we vary the number of mediators and participants to evaluate the scalability of VM_bQA.

7.2 Quality Results in Mono-mediator VOs

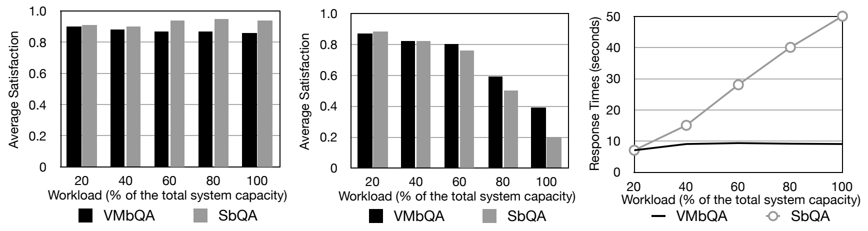
To avoid any impact in performance of having multiple mediators, we run these experiments with a single mediator to better evaluate the performance of VM_bQA. We release this assumption in the next section. We assume the mediator has enough resources so that it does not become a performance bottleneck.

Figure 2 illustrates how these three methods satisfy participants for different workloads. As expected, we observe in Figure 2(a) that $FPSB$ is completely neutral to consumers because it does not take into account their intentions, which is not the case for $BLSB$ and VM_bQA. In particular, we observe that VM_bQA outperforms the $FPSB$ and $BLSB$ methods by ensuring higher satisfaction for consumers and providers. In fact, we observed during our experiments that $FPSB$ and $BLSB$ methods have some problems to balance queries because most adequate and preferred (by consumers) providers tend to monopolize incoming queries. VM_bQA does not suffer from this phenomenon by establishing a more sophisticated invoice mechanism. Of course, this impacts on performance with long response times for $FPSB$ and $BLSB$ (see Figure 2(c)). From results, we can conclude that we can introduce virtual money, without any loss of system performance, to regulate a system as long as we care about the way in which providers are selected and invoiced.



(a) Consumers satis. (b) Providers satis. (c) Response time.

Fig. 2. Results in monomediator VOs for different workloads



(a) Consumer satis. (b) Provider satis. (c) Response time.

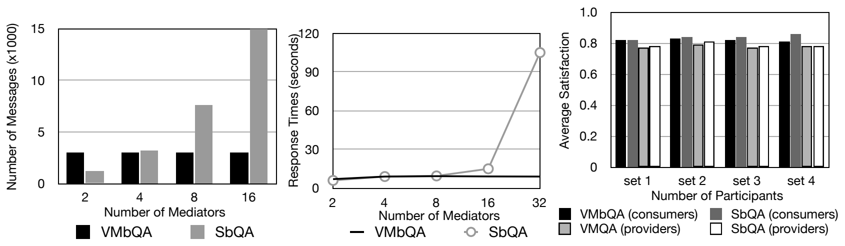
Fig. 3. Quality results with captive participants for different workloads in a x-redundant VO with 8 mediators

7.3 Dealing with x-Redundant VOs

We now compare VM_bQA against S_bQA — which is the baseline for satisfying participants — when scaling up the query allocation up to several mediators. We also demonstrated in the previous section that VM_bQA correctly deals with participants departures. Thus, in these experiments, we consider captive participants to better study the scalability of both methods.

We start by evaluating the impact of having several mediators allocating queries could have. We run a series of experiments for different workloads in a x-redundant VO with 8 mediators. We observe in Figure 3(a) that S_bQA better satisfies consumers than VM_bQA, because VM_bQA makes a better balance between the satisfaction of consumers and providers. We can clearly observe this in Figure 3(b) where VM_bQA outperforms S_bQA. In particular, we observe that the satisfaction of providers decreases as the workload increases. This is because providers becomes more utilized and hence it is more difficult to satisfy them. S_bQA cannot ensure the same providers satisfaction as VM_bQA because of the time it takes to update providers satisfaction at all mediators. Furthermore, the network messages generated by S_bQA consume computational resources at mediators side, which degrades the response times (see Figure 3(c)). This is not the case for VM_bQA, which does not need to update providers satisfaction and hence it significantly outperforms S_bQA.

We also run a series of experiments with the aim of analyzing the impact, from a performance point of view, of having several mediators allocating queries. In



(a) Exchanged mssgs. (b) Several mediators (c) Several participants

Fig. 4. For a workload of 60% of the total system capacity, (a) illustrates the number of exchanged messages, (b) illustrates the response times with different number of mediators, and (c) illustrates the average satisfaction with different number of participants: 50 consumers and 100 providers (set1), 100 consumers and 200 providers (set2), 200 consumers and 400 providers (set3) and 400 consumers and 800 providers (set4)

Figure 4(a), we plot every 1000 incoming queries the number network messages exchanged by VM_bQA and S_bQA to manage virtual money and satisfaction, respectively. Notice that, in these results, we do not plot all the network messages required by VM_bQA and S_bQA to allocate a query since such a number is the same if we discard the number of network messages to manage virtual money and satisfaction. This why we just plot this difference in number of messages. We can observe that VM_bQA always generates 3 network messages per query while the number of network messages generated by S_bQA depends on the number mediators. We observe that from 4 mediators S_bQA generates more number of messages than VM_bQA. We also measure the impact in response times of varying the number of mediators (see Figure 4(b)). We observe, on the one hand, that the performance of VM_bQA does not depend on the number of mediators and hence its performance is constant. On the other hand, we observe that S_bQA cannot perform well for a high number of mediators because of the number of messages it generates. Finally, we analyze how well S_bQA and VM_bQA satisfy participants when the number of participants in a VO varies. To this end, we run several experiments with a single mediator with a workload of 60% of the total system capacity, but with different number of participants. Figure 4(c) illustrates these results. We observe that VM_bQA has, on average, the same performance as S_bQA. In other words, VM_bQA can scale up, in terms of number of participants, without any loss in satisfaction of participants. All these results demonstrate that VM_bQA can scale up in terms of number of participants, mediators, and incoming queries, while satisfying participants as in VOs with a single mediator.

8 Conclusion

We considered large-scale distributed information systems where several mediators allocate queries and participants are free to leave the system and have special interests towards queries. In particular, we addressed the problem of scaling up the query allocation process to several mediators by keeping participants satisfied. To overcome this problem, we proposed VM_bQA, a query allocation method that uses virtual money as a means of regulation.

In summary, we made the following contributions. First, we formalized the query allocation problem for multi-mediator systems with autonomous participants. Second, we demonstrated that only a few number of network messages (3 per query) are required by a mediator to control the flow of virtual money. Third, we proposed VM_bQA, a *Virtual Money-based Query Allocation* method that considers both consumers intentions and providers bids. As a part of VM_bQA, we defined a bidding procedure that considers the satisfaction, utilization, preferences, and virtual money balance of providers, and proposed three techniques that allow a provider to bid for queries in multi-mediator systems. Fourth, we analytically and experimentally demonstrated that VM_bQA can efficiently scale up to several mediators with no additional network cost (compared to a VO with a single mediator). We experimentally showed that VM_bQA significantly outperforms baseline methods from a satisfaction and performance points of views. The results also show that, in contrast to many microeconomic approaches, a query allocation method should pay special attention to the invoicing mechanism, otherwise it may have a great negative impact in performance. Last but not least, to our knowledge this is the first work that evaluated microeconomic approaches through a measure (participants satisfaction) that is outside the microeconomic scope (besides load balancing and response time).

References

1. Dash, R.K., et al.: Market-Based Task Allocation Mechanisms for Limited Capacity Suppliers. IEEE Transactions on Systems 37(3), 391–405 (2007)
2. Ferguson, D., et al.: Economic Models for Allocating Resources in Computer Systems. In: Clearwater, S.H. (ed.) Market-Based Control: A Paradigm for Distributed Resource Allocation. World Scientific, Singapore (1996)
3. Markatos, E.P.: Tracing a large-scale peer to peer system: An hour in the life of gnutella. In: CCGRID (2002)
4. Pentaris, F., Ioannidis, Y.: Query Optimization in Distributed Networks of Autonomous Database Systems. TODS 31(2) (2006)
5. Pentaris, F., Ioannidis, Y.: Autonomic Query Allocation Based on Microeconomics Principles. In: ICDE (2007)
6. Quiané-Ruiz, J.-A., Lamarre, P., Valduriez, P.: A Self-Adaptable Query Allocation Framework for Distributed Information Syst. VLDBJ 18(3), 649–674 (2009)
7. Sandholm, T.W.: Distributed Rational Decision Making. In: Multiagent Systems, a modern approach to Distributed Artificial Intelligence. The MIT Press, Cambridge (2001)
8. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable Peer-To-Peer lookup service for internet applications. In: SIGCOMM (2001)
9. Stonebraker, M., et al.: Mariposa: A Wide-Area Distributed Database System. VLDBJ 5(1), 48–63 (1996)
10. Vickrey, W.: Counterspeculation, Auctions, and Competitive Sealed Tenders. International Journal of Finance 16(1) (1961)
11. Wolfstetter, E.: Auctions: and introduction. Economic Surveys 10(4) (1996)
12. Yang, B., Garcia-Molina, H.: Designing a Super-Peer Network. In: ICDE (2003)

Generating Preview Instances for the Face Validation of Entity-Relationship Schemata: The Acyclic Case

Maria Amalfi[1], Alessandro Artale[1], Andrea Calì[2,3], and Alessandro Provetti[2,4]

[1] Computer Science, Free Univ. of Bozen-Bolzano, I-39100 Bozen-Bolzano Italy
[2] Oxford-Man Institute for Quantitative Finance, Oxford OX2 6ED, UK
[3] Dept. of Comp. Science & Inf. Sys. Birkbeck, Univ. of London, London WC1E, UK
[4] Dept. of Physics, Informatics Section, University of Messina, I-98166 Messina, Italy

Abstract. We describe a mapping of Extended Entity-Relationship schemata to Answer Set Programming that allows us to generate informative example instances of the relational database that is implied by the conceptual schema. Such instances may be submitted to people involved in the requirements phase as a glimpse of what instances are allowed by the E-R schema at hand, thus enhancing database comprehension and so-called face validation.

1 Introduction

In the design and implementation of database systems (DB) one important step is the creation of the conceptual model, namely the Entity-Relationship (ER) schema [14]. The graphical language of ER schemata normally captures (almost) all aspects of the specification. ER schemata are useful both to i) communicate ideas, choices and requirements among humans involved in a project and ii) as a first-step in the process of (semi-)automated generation of the database and of the needed software.

DB designers draw the first version of ER schema starting from a set of informal specifications, normally given in natural language and/or catalogue table describing names, attributes etc. The ER schema preparation often goes through several refinements. A refinement normally follows the realization that some aspects of the informal specification are not properly reflected by the current ER schema. Such process may be tedious and error-prone, but can hardly be made totally automated due to the inherent ambiguity and incompleteness of practically any natural-language specification. In other words, it is very hard, to give a formal proof of correctness of a ER schema w.r.t. its informal specification. Hence, formal methods and the mathematical assessment of the ER are not readily applied to the face validation of a putative ER schema.

In this work we propose a methodology and a set of tools for supporting the designer in the validation phase: given an ER schema, we automatically generate a *sample snapshot* of a state of affairs permitted by the conceptual schema.

Such sample, described in terms of individual instances of entities (and relationships), may be suggestive of how the relational database, once it is created and populated, would look like. By seeing a preview sample, the designer (or other less-skilled participants to the design phase, viz. potential users) may make up their mind whether the proposed sample is an acceptable database state, i.e., it reflects a realistic situation and obeys the constraints that were stated informally.

If what they see is deemed *correct* (in an informal sense of the word), then the ER can be finalized and the subsequent phases of the design (restructuring, translation to the logical level and so on) can be carried out in a formalized setting where correctness can be proved as a mathematical property. The process described above is aptly called *face validation* in fields e.g. Expert Systems [13] and Multi-agent systems [12]. Miller et al. [1] use a similar terminology in the context of distributed databases.

In this paper we lay the technical foundations of our project with the following results:

1. we define the notion of *preview instance,* intended as a sample snapshot of a typical state of affairs implied by the ER schema, i.e., we define the properties that a preview instance should possess;
2. we define a mapping from the EER syntax to Logic programs (with function symbols) under Answer Set Semantics [10] and prove its properties (correctness and completeness).

Due to lack of space, in this paper we shall assume the reader acquainted with the main technical definitions of ER, database constraints and logic programming under Answer Set semantics[1].

2 Related Research

To the best of our knowledge, this work is the first attempt to generate example instances out of EER schemata. Nonetheless, several research efforts have addressed generating example instances, mainly at the relational level. Let us summarize some of the best known here. The generation of relational instances had been studied (also with some implementation effort) in the 80's and early 90's by several researchers, e.g., [9,11,2] who were interested in the mathematical properties of Armstrong databases (ADBs). ADBs, in short, can be described as synthetic instances that i) respect all constraints given (or logically implied) by the designer and ii) violate all other possible constraints.

As it has been discussed in the Introduction, our work remains somewhat orthogonal to those on ADBs, as we focus on the conceptual level and on properties of EER schemata. In particular, the known results on existence and finiteness of the ADB for arbitrary relational schemata do not apply readily here since we are considering only a subclass of key and inclusion dependencies (INDs) called *conceptual dependencies* [6].

[1] A companion Web site with software, examples and the long version of this article is at `http://mag.dsi.unimi.it/EER2instances/`

In the area of information integration, the recent *Muse* (Mapping Understanding and deSign by Example) project of Alexe et. al [1] exploits example instances to assist designers in understanding and refining a schema mapping between relational databases. MUSE aims to guide the database designer in the specification of the desired grouping of data, which are derived from related data source-sets. Moreover, Muse drives the designer to choose the more suitable mapping among ambiguous mapping. A schema mapping is deemed *ambiguous* when it specifies, in more than one way, how an atomic element (or attribute) of the target schema is to be obtained. Muse uses small examples to construct a small number of *yes-no questions* such that, by answering them, the designer comes to construct, step by step, the mapping she had in mind. Miller et al. make a compelling case for the benefits of making designers work with data rather than with complex specifications, since they have overwhelming empirical evidence that mapping designers usually understand their data better than they understand mapping specifications. To some extent such consideration carries over to our case though the data we can present to the designer is purely synthetic.

In the context of data mining and database understanding, De Marchi and Petit [8] have introduced the notion of *illustrative database instance.* They consider a data-mining scenario where a database instance is available but the relative relational schema may not be accessible or even lost. Their aim is to provide information about the structure of the database by letting the user inspect a suitably small instance. Given an existing, normally very large, database instance, DeMarchi et al. *carve out* a small database *sample* that i) has *real* (as opposed to synthetic) tuples drawn from the existing DB, and ii) shows the same FDs and IDs as the original.

There are two main differences between work and De Marchi-Petit's. The first, and fundamental, is that, in typical data mining fashion, they are working out their instance *backwards* from an existing one, focusing on the instance itself rather than the relational schema. In our work, we work *forward* from the conceptual schema to a sample instance of a not-yet-existing database. As a result, whereas their attribute values are real, ours are synthetic. The second difference is the notion of minimality. DeMarchi-Petit's Informative instances are minimal w.r.t. inclusion. Our illustrative instances are not necessarily minimal, by design. Let's discuss minimality now.

Even when a conceptual schema would lead to a relational schema that admits the empty instance, we consider it to be scarcely illustrative of how the implementation is going to look like during its life-cycle. A small example with, say, only one tuple per relation would give a better illustration of how data in each table connect to each other. Our translation is defined starting from such concept: each relation representing entities and relationships must be non-empty from the beginning. Indeed, the notion of *full satisfiability,* studied, e.g., in [3] is along the same lines. The enforcement of inclusion dependencies may later impose higher cardinalities. However, the minimal model semantics of Answer Set Programming guarantees that cardinalities are always minimal w.r.t. the initial instantiation of one tuple per relationship.

2.1 Extended Entity Relationship

We adopt from [6] an others an extension of Chen's Entity-Relationship graphical language. It consists of a common extension that allows for *IS-A* relations between entities or even relationships. Due to the lack of space, we rely on the reader being acquanited with Entity-Relationship and database constraints, and proceed to formulate an example.

Example 1. Consider the EER schema shown in Figure 1, depicted in a slightly extended graphical notation for the EER model (components are indicated by integers for the relationships). The schema describes employees working in departments of a firm, and managers that are also employees, and manage departments. Managers who manage a department is also deemed to work in it.

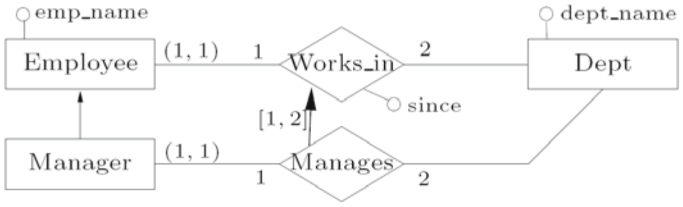


Fig. 1. A small (but non-trivial) EER schema

The *(1, 1)* participation constraint on *Employee* in *Works_In* imposes that every instance of *Employee* participates at least once and only once (functional participation) in *Works In*; the same constraints hold on the participation of *Manager* in *Manages*. Vice versa, the participation of entity *Dept* to relationships *Works_in* and *Manages* is optional (0,N). The *[1, 2]* label in the *is-a* relation between the two relationships denotes that the components of *Manages* correspond, in that order, to components 1 and 2 of *Works In*. Several straightforward inclusion dependencies can be read out of this schema, e.g., $Manager[1] \subseteq Employee[1]$ from the IS-A between entities; by lack of space they cannot be listed here.

3 Preview Instances

Let us now define preview instances of EERs schemata. First of all, an EER *instance* is defined as a set of object constants, called occurrences assigned to entities and a set of tuples assigned to relationships. We assume that the unique-name assumption holds, so different objects are identified by different constants. An EER instance is a *preview instance* if:

1. all constraints are satisfied;
2. there is at least one distinct occurrence for each of the entities and for each of the relationships;

3. any two entities are either in containment (i.e., there is an IS-A constraint between them) or disjoint; vice versa, relationships are not necessarily disjoint[2], and
4. the extension of each entity is minimal (modulo constant renaming) wrt. the constraints.

4 Computational Complexity Issues

Given an EER schema $\mathcal{C}$, what is the (worst-time) complexity of deciding whether $\mathcal{C}$ admits a legal instance, i.e., one that respects all constraints? Artale et al. [4] have assessed the computational complexity of deciding whether a given ER schema admits a model (in their semantics for ER), which they term *satisfiability.* They study complexity of several variations of the satisfiability problem that are of practical relevance for DB design[3]. In particular, their concept of *full satisfiability* comes very close to our concept of preview instance.

However, our working EER syntax is slightly different from Artale et al.'s, (and such differences may have a huge impact on complexity); it falls between that of ER_{full} (the version with the most expressive language, where checking consistency has EXPTIME complexity) and ER_{isaR}. On one hand, here we are considering EERs that do not allow for the so-called boolean constraints: covering and disjunction between classes. On the other hand, we always allow functional participation of an entity to a relationship; the dual constraint of mandatory participation of an entity to a relationship is not considered here because of the acyclicity restriction. As a result, of EER syntax, and its complexity is not directly comparable with Artale et al. taxonomy, which is summarized in Table 1.

Table 1. Complexity of the consistency problem for some variations of ER, from [3,5]

Language	part. c.			entities			relationships			Complexity
	0,1	1,k	1,1	ISA	DISJ	COV	ISA	DISJ	COV	
				$C_1 \sqsubseteq C_2$	$C_1 \sqcap C_2 \sqsubseteq \bot$	$C = C_1 \sqcup C_2$	$R_1 \sqsubseteq R_2$	$R_1 \sqcap R_2 \sqsubseteq \bot$	$R = R_1 \sqcup R_2$	
ER_{full}	−	−	−	+	+	+	+	+	+	ExpTime
ER_{isaR}	−	−	−	+	+	+	+	−	−	ExpTime
ER_{bool}	−	−	−	+	+	+	−	−	−	NP
ER_{ref}	−	−	−	+	+	−	−	−	−	NLogSpace
ER_{ref}+ISA$_R$	−	−	−	+	+	−	+	−	−	NLogSpace
ER_{ref}+ISA$_R$+(0,1)	+	−	−	+	+	−	+	−	−	acyclic
ER_{ref}+ISA$_R$+(1,k)	−	+	−	+	+	−	+	−	−	cyclic
ER_{ref}+ISA$_R$+both	+	+	+	+	+	−	+	−	−	ExpTime

[2] Notice how this provision disallows entities with *multiple inheritance.*

[3] Also, [3] proves exact measures of the worst-case complexity for the problem of finding models of UML diagram that has each class assigned to a non-empty set of symbols.

Whereas our syntax corresponds to case ER_{ref}+ISA$_R$+*both*, the acyclicity restriction adopted here reduces us to consider the ER_{ref}+ISA$_R$+*(0,1)* case, with a worst-case complexity that is still indeterminate. Hence, our embedding into ASP programs provides a loose upper-bound of Σ_2^p; in any case since complexity here is functional to the number of entities/relationships given in input, higher worst-case complexities are still manageable for small instances.

5 From EER to Answer Set Programs

The embedding of EER into ASP programs with function symbols is composed of three sets of rules: i)a line-by-line translation were each constraint of the input EER schema are mapped to an equivalent rule –or constraint– of the output ASP program; ii) a set of facts and rules is introduced once and for all at the beginning of the translation to reconstruct the EER signature (entity names, relationship names, arities etc.), within the signature of the output logic program, and iii) a metapredicate *participates* that describes, via reification of entity and relationship names, the order of participation of entities to a relationship. The thus-assembled ASP program is fed to the DLV-complex inferential engine for ASP programs (please refer to [7] for the details of this novel system), which will compute an answer set representative of the preview instance.

Let $\mathcal{C}$ be an EER schema (sometimes called *EER project*) and let us define its mapping into logic program $\pi_{\mathcal{C}}$ as follows. First, the signature of $\pi_{\mathcal{C}}$, i.e. $\mathcal{L}_{\pi_{\mathcal{C}}}$, shall reflect the names used in the EER:

a: for each entity name E in $\mathcal{C}$
- let $\mathcal{L}_{\pi_{\mathcal{C}}}$ contain the pred. constant e of arity 1, henceforth denoted *e/1*;
- add to $\pi_{\mathcal{C}}$ a fact $e(c)$ where c is a fresh constant.

b: for each attribute name A_E for an Entity E in $\mathcal{C}$ let $\mathcal{L}_{\pi_{\mathcal{C}}}$ contain the predicate a_E of arity 2 ($a_E/2$), where the first component refers to the entity E and the second one is a function defined on it (the attribute hence is functional);

c: for each relationship name R of arity n in $\mathcal{C}$, let $\mathcal{L}_{\pi_{\mathcal{C}}}$ contain predicate r/n where there is one component for each entity which participates in the relationship, and

d: for each attribute name A_R for a Relationship R of arity n in $\mathcal{C}$ let $\mathcal{L}_{\pi_{\mathcal{C}}}$ contain the predicate $a_R/n+1$ where the first n components refer to the entities participating in the relationship and the last one is an 'attribute' function defined on the participating entities.

Next, we define the translation that populates $\pi_{\mathcal{C}}$ with rules and constraints.

1. for each attribute name A_E for an Entity E in $\mathcal{C}$ let $\pi_{\mathcal{C}}$ contain a constraint which imposes that the first component of the predicate a_E must be a term that also occurs in predicate e/n (which represents E)[4];

[4] Step 1 makes explicit a form of *closed-world* assumption on attribute values that is common but somewhat left implied.

2. for each attribute name A_R for a Relationship R of arity n in $\mathcal{C}$ let $\pi_\mathcal{C}$ contain a constraint that the first n components of the predicate a_R must be an instance of predicate r/n (which represents *R);*
3. for each relationship name R of arity k in $\mathcal{C}$, let $\pi_\mathcal{C}$ contain k constraints imposing that each of the k entities in a relationship instance must also be an instance of the relative entity $e_i, 1 \leq i \leq n$;
4. for each mandatory attribute name A_E for an Entity E in $\mathcal{C}$ let $\pi_\mathcal{C}$ contain a consistency constraint that disallows an instance of e which is not an instance of the connected predicate a_E;
5. for each mandatory attribute name A_R for a Relationship R of arity n in $\mathcal{C}$ let $\pi_\mathcal{C}$ contain a consistency constraint that disallows an instance of r which is not an instance of the connected predicate a_R;
6. for each functional attribute name A_E for an Entity E in $\mathcal{C}$ let $\pi_\mathcal{C}$ contain a key constraint (expressed as a consistency constraint);
7. for each functional attribute name A_R for a Relationship R of arity n in $\mathcal{C}$ let $\pi_\mathcal{C}$ contain a key constraint;
8. for each *IS-A* relation between two entities, called, say, E_{mother} and $E_{daughter}$, i.e., respectively, destination and origin of the IS-A arrow, in $\mathcal{C}$ let $\pi_\mathcal{C}$ contain an "inheritance" rule by which every occurrence of $E_{daughter}$ will also be an occurrence of E_{mother};
9. for each *IS-A* relation between two relationships, say, R_{mother} and $R_{daughter}$ in $\mathcal{C}$ let $\pi_\mathcal{C}$ contain an inheritance rule by which every occurrence of $R_{daughter}$ will also be an occurrence of R_{mother};
10. for each mandatory participation of the i-th component of an entity E in a relation R of $\mathcal{C}$, let $\pi_\mathcal{C}$ contain a consistency constraint by which an instance of E is forced to appear in the i-th position of one instance of R. To capture this requirement, we need to introduce an auxiliary predicate *participates(e, r, i)* which reifies predicate r into the (fresh) term r in order to represent its parameters.
11. for each functional participation of the i-th component of an entity E in a relation R of $\mathcal{C}$, let $\pi_\mathcal{C}$ contain a consistency constraint on mandatory participation of the i-th component of an entity in a relation. Such requirement is captured, as in rule 10, by introducing a *participates* auxiliary predicate that reifies the predicate representing *R.*

6 Results

Thanks to the embedding described in the previous section, we can now prove that for any EER schema $\mathcal{C}$ which is *acyclic* (in short: IDs do not form loops) and *consistent* (in short: there exist legal instances, pl. see again Artale et al.) the associated DLV-Complex program $\pi_\mathcal{C}$ has an answer set. The answer set of $\pi_\mathcal{C}$ contains atoms from which a preview instance for $\mathcal{C}$ can be easily read out.

Currently, the class of EER for which the translation is proved correct, i.e., yields preview instances whenever they exist, can be characterized by the formal statement below, where by cyclic definitions we intend a cycle on the EER graph.

The formal definition is as follows. Given a project $\mathcal{C}$, we define the underlying graph $\mathcal{G}_{\mathcal{C}}$ as a directed graph that has

- one node for each entity (box) and one node for each relationship (diamond) of $\mathcal{C}$;
- one arc from the node representing a relationship R to the node representing entity E whenever E participates to R;
- an arc from the node representing an entity E to the node representing a relationship R whenever E participates to R with cardinality constraint (1,1) or (1,N), and
- an arc between each two nodes representing entities (resp. relationships) that are in IS-A relation in $\mathcal{C}$ (same orientation of the arrow).

Proposition 1. *For any EER $\mathcal{C}$ for which the underlying graph $\mathcal{G}_{\mathcal{C}}$ is acyclic the answer sets of $\pi_{\mathcal{C}}$ are in one-to-one correspondence with preview instances of $\mathcal{C}$.*

To prove it, we shall take advantage of the set-minimality property of answer sets [10]. So, let $\mathcal{S}$ be an answer set of $\pi_{\mathcal{C}}$, and let $\mathcal{S}'$ be its *projection* on predicates e_i, r_i and a_i representing entities, relationships and attributes, respectively. $\mathcal{S}'$ trivially satisfies conditions 1 and 3 of the preview instance condition: the constraints introduced into $\pi_{\mathcal{C}}$ by rules 1-7, 10 and 11 are satisfied by $\mathcal{S}$.

By construction, i.e., steps A and B of the translation, each entity predicate e has at least an instance. Also, by the inclusion rules described in step 3 of the translation and by the above fact, it is easy to conclude that S must also contain at least an instance for each relationship relation r (condition 2 of preview instances) On the other hand, minimality (condition 4 of preview instances) is guaranteed by the set-minimality property of answer sets.

Proposition 2. *For any EER that contains neither cyclic definitions nor IS-A between relationships, $\pi_{\mathcal{C}}$ there is a 1-1 correspondence, modulo the language between answer sets and illustrative instances of the ER schema.*

6.1 Translation of the Example EER

When applied to the schema in the Example, our translation produces 33 formulae, 18 of which are constraints. It is not possible to list the complete translation here, but it can be found in the long version of this work[5]. However, it possible to comment on the tricky points of the translation of this particular example:

- step C is composed of two parts: first we create the individual instances of each relation, then we add rules stating that the constants used in those instances are actually individuals of the entities participating to the given relationship;
- steps 6 and 7 do not apply to this example, as there are no keys;

[5] Available from http://mag.dsi.unimi.it/EER2instances

– step 10 may be tricky to grasp because it involves the definiton of tha *participates* predicate that describes the syntactic strucure of our predicates, rather than the instance itself:

$participates(E, works\_in, 1) :- dept(D), works\_in(E, D).$
$participates(D, works\_in, 2) :- employee(E), works\_in(E, D).$
$participates(E, manages, 1) :- dept(D), manages(E, D).$
$participates(D, manages, 2) :- manager(M), manages(M, D).$

6.2 The Model Generation Phase

DLV-Complex is an extension of the well-known DLV inferential engine[6] for deductive databases and disjunctive logic programming which supports syntactic functions and lists manipulation, allowing for a powerful (possibly recursive) use of function terms together with disjunction and negation.

The ability to treat function symbols, which was not present in the basic DLV inferential engine is here essential for a straightforward and efficient treatment of inclusion dependencies, e.g. in rules like:
$emp\_name(E, emp\_name\_of(E)) :- employee(E).$
If we feed program $\pi_{\mathcal{C}}$ obtained from our translation of the previous Example to DLV-Complex we obtain the following Answer Set (with some abbreviation):

$\{manages(man, dept), employee(emp), employee(man), manager(man), dept(dept),$
$emp\_name(emp, emp\_name\_of(emp)), emp\_name(man, emp\_name\_of(man)),$
$dept\_name(dept, dept\_name\_of(dept)), works\_in(emp, dept), works\_in(man, dept),$
$since(emp, dept, w\_in\_since(emp, dept)), since(man, dept, w\_in\_since(man, dept)),$
$part(emp, works\_in, 1), part(man, works\_in, 1), part(man, manages, 1),$
$part(dept, works\_in, 2), part(dept, manages, 2)\}$

7 Conclusions

We have defined a new mapping of EER schemata to logic programs that enables the automated creation of preview instances. Even though such type of mappings are known in the literature, our translation is the first, to the best of our knowledge, that permits to characterize and computer preview instances of an EER project. On the theory side, we believe that our mapping, once we allow for the needed adjustments to the variety of ER graphical languages now available, may also support other types of reasoning about specifications, such as evaluating the properties of conjunctive queries against the EER proposed by Calì in [6]. However, several interesting open questions arise wrt. i) the relation between our instances and Armstrong databases and ii) the possibility that an EER schema, even though it admits legal finite instances, may induce our program to the construction of an infinite instance. The latter case could happen in presence of a circular sequence of inclusion dependencies, due to the interaction of *is_a* and obligatory participation constraints in the EER schema.

[6] The software and related literature are available at `http://www.dlvsystem.com/`

By its very nature, this work is not suitable for evaluation in terms of traditional benchmarks/scalability analysis. Rather, face validation tests, e.g., with panels of DB students would give us a feedback on the impact on schema comprehension obtained by submitting preview instances back to those who laid out the first specification.

The translation to DLV-Complex programs that is the core of our work has been implemented and will be released as a branch project of the ERW tool [15].

References

1. Alexe, B., Chiticariu, L., Miller, R.J., Tan, W.C.: Muse: Mapping understanding and design by example. In: ICDE, pp. 10–19. IEEE, Los Alamitos (2008)
2. Armstrong, W., Delobel, C.: Decompositions and functional dependencies in relations. ACM Trans. on Database Systems (TODS) 5(4), 404–430 (1990)
3. Artale, A., Calvanese, D., Ibanez-Garcia, A.: Full satisfiability of UML class diagrams. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, Springer, Heidelberg (2010)
4. Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: Reasoning over extended ER models. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 277–292. Springer, Heidelberg (2007)
5. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-lite family and relations. J. Artif. Intell. Res. (JAIR) 36, 1–69 (2009)
6. Calì, A.: Containment of conjunctive queries over conceptual schemata. In: Li Lee, M., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, pp. 628–643. Springer, Heidelberg (2006)
7. Calimeri, F., Cozza, S., Ianni, G., Leone, N.: Enhancing asp by functions: Decidable classes and implementation techniques. In: Fox, M., Poole, D. (eds.) AAAI. AAAI Press, Menlo Park (2010)
8. DeMarchi, F., Petit, J.M.: Semantic sampling of existing databases through informative armstrong databases. Information Systems 32(3), 446–457 (2007)
9. Fagin, R., Vardi, M.Y.: Armstrong databases for functional and inclusion dependencies. Information Processing Letters 16(1), 13–19 (1983)
10. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing, 365–387 (1991)
11. Gottlob, G., Libkin, L.: Investigations on armstrong relations, dependency inference and excluded functional dependencies. Acta Cybernetica 9(4), 385–402 (1990)
12. Klügl, F.: A validation methodology for agent-based simulations. In: Wainwright, R.L., Haddad, H. (eds.) SAC, pp. 39–43. ACM, New York (2008)
13. O'Keefe, R.M.: The validation of expert systems revisited. The Journal of the Operational Research Society 40(5), 509–511 (1989)
14. Thalheim, B.: Entity-Relationship Modeling: Foundations of Database Technology. Springer, Berlin (2000)
15. Vigna, S.: Reachability problems in entity-relationship schema instances. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 96–109. Springer, Heidelberg (2004)

Dynamic Skylines Considering Range Queries

Wen-Chi Wang[1], En Tzu Wang[2], and Arbee L.P. Chen[3,*]

[1] Telecommunication Laboratories Chunghwa Telecom Co., Ltd., Taipei, Taiwan, R.O.C.
lovelynoir@gmail.com
[2] Cloud Computing Center for Mobile Applications, Industrial Technology Research Institute, Hsinchu, Taiwan, R.O.C.
m9221009@em92.ndhu.edu.tw
[3] Department of Computer Science, National Chengchi University, Taipei, Taiwan, R.O.C.
alpchen@cs.nccu.edu.tw

Abstract. Dynamic skyline queries are practical in many applications. For example, if no data exist to fully satisfy a query q in an information system, the data "closer" to the requirements of q can be retrieved as answers. Finding the nearest neighbors of q can be a solution; yet finding the data not *dynamically dominated* by any other data with respect to q, i.e. the dynamic skyline regarding q can be another solution. A data point p is defined to dynamically dominate another data point s, if the distance between each dimension of p and the corresponding dimension of q is no larger than the corresponding distance regarding s and q and at least in one dimension, the corresponding distance regarding p and q is smaller than that regarding s and q. Some approaches for answering dynamic skyline queries have been proposed. However, the existing approaches only consider the query as a point rather than a range in each dimension, also frequently issued by users. We make the first attempt to solve a problem of computing dynamic skylines considering range queries in this paper. To deal with this problem, we propose an efficient algorithm based on the grid index and a novel variant of the well-known Z-order curve. Moreover, a series of experiments are performed to evaluate the proposed algorithm and the experiment results demonstrate that it is effective and efficient.

Keywords: Dynamic attributes, Dynamic skyline computation, Range queries, Grid index, Z-order curve.

1 Introduction

In recent times, the skyline computation [1][2][3][4][5][6][7][8][9][11][13][14][15][16][17]has received considerable research attention due to playing an important role in the applications involving multi-criteria decision making. Given an n-dimensional dataset D, the skyline of D is a set of data points (objects) not dominated by any other data points in D. A data point t with n dimensions ($t[1]$, $t[2]$, …, $t[n]$) is defined to *dominate* another data point s ($s[1]$, $s[2]$, …, $s[n]$) iff 1) $t[i] \leq s[i]$, $\forall$ i = 1 to n, and 2) at least in one dimension, say j, $t[j] < s[j]$. This definition of domination is under the assumption that smaller values are preferred. An illustration of the skyline of a hotel dataset is shown in Fig. 1. In this illustration, each hotel has two attributes (dimensions)

[*] Corresponding author.

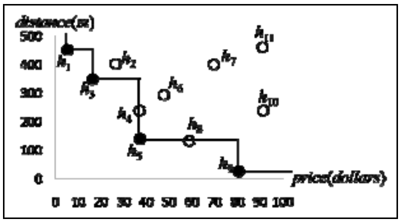


Fig. 1. The traditional skyline of a hotel dataset

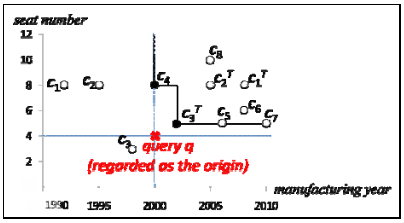


Fig. 2. The dynamic skyline of a car dataset

including the price and distance to the beach. Since users usually prefer choosing the hotels with distances closer to the beach and lower prices, the skyline of the dataset is $\{h_1, h_3, h_5, h_9\}$. For example, hotel h_2 is not in the skyline since neither the price nor distance to the beach is smaller than that of hotel h_3. Many approaches have been proposed for efficient skyline computation, such as BNL [1], D&C [1], Bitmap [16], SFS [3], LESS [8], BBS [11], SaLSa [2], ZSearch [9], and OSP [17]. However, most of them only focus on *static attributes*, which means the attributes are rarely changed such as "the distance to the beach" of a hotel.

The *dynamic skyline* mentioned in [11][5][13] is a variant of the original skyline, where the attributes of each data point in a given dataset are calculated *dynamically* according to a query issued. More specifically, given an n-dimensional dataset D and a query q with n dimensions ($q[1]$, $q[2]$, …, $q[n]$), the dynamic skyline query regarding q retrieves the data points not *dynamically dominated* by any other data points, with respect to q. A data point t is defined to dynamically dominate another data point s, with respect to q, iff 1) $|t[i] - q[i]| \leq |s[i] - q[i]|$, $\forall\ i = 1$ to n, and 2) at least in one dimension, say j, $|t[j] - q[j]| < |s[j] - q[j]|$. An illustration of the dynamic skyline of a car dataset is shown in Fig. 2. Suppose that a user prefers the car whose manufacturing year and seat number are 2000 and 4, respectively. Then, q can be regarded as a *query point* of (2000, 4). If no data points exist to fully satisfy q, the data points with attributes closer to the requirements of q, i.e. 2000 and 4, can be retrieved to recommend the user instead. The nearest neighbors of q could be retrieved as answers. However, if we consider user preference, the *dynamic skyline points* are more representative than the nearest neighbors of q, where a dynamic skyline point is a data point in the dynamic skyline. Accordingly, we turn to find the skyline in a *transferred dataset* in which all of the data points in the original space are transferred to the other space whose origin is equal to (2000, 4). For example, $c_1 = (1992, 8)$ is transferred to $c_1^T = (|1992 - 2000|, |8 - 4|) = (8, 4)$, and moreover, c_2 and c_3 are transferred to $c_2^T = (5, 4)$ and $c_3^T = (2, 1)$, respectively. Then, the skyline in the transferred space, i.e. the dynamic skyline, is equal to $\{c_4, c_3\}$.

In many situations, users describe their needs in an imprecise way, i.e. issuing range queries. For example, a user may want a car equipped with 2 to 4 seats and manufactured between 2006 and 2008. If some data points exist to fully satisfy the range queries, those data points are exactly what the user wants. If no data points exist, finding the dynamic skyline is then performed. Since range queries are also frequently issued, we make the first attempt to study a new problem on computing dynamic skylines considering rangy queries in this paper. Similar to the dynamic skyline regarding a query point, we want the skyline in a transferred dataset in which all of the data points in the original space are transferred to the other space according to the range query. If the user query is a multi-dimensional point, we just need to transfer the data points into the space whose origin is equal to the query point, accordingly. While the user query has ranges in the dimensions, the definition of transformation needs to be reconsidered. We think that since the range in a dimension represents the user preference in this dimension, the data point with the corresponding dimension closer to the range, i.e. closer to any position in the range, is appropriate to recommend to the user if no data points fall into the range regarding the corresponding dimension. The transformation can therefore be described as follows. To a data point p in a dataset, we find a position s in the range query, closest to p, and transfer p to the other point p^T according to s. The formal definition related to this transformation is detailed in Section 3. A straightforward solution to compute the dynamic skyline regarding a range query is to transfer all of the data points according to the range query and then apply one of the existing skyline algorithms, e.g. SFS, to obtain the skyline in the transferred dataset. Obviously, it is costly to scan the whole dataset to generate the transferred dataset and then again scan the whole transferred dataset for dominance checking. Therefore, we propose an efficient approach based on the grid index and a variant of the well-known Z-order curve [10] to avoid the need to generate the whole transferred dataset, thus also reducing the times of dominance checking.

Our contributions can be summarized as follows. 1) We propose a new problem on dynamic skyline computation regarding a range query. Moreover, the semantics of the novel dynamic skyline query is also well explained as above. 2) To efficiently answer this query, we propose an approach based on the gird index and a newly designed variant of the well-known Z-order curve. By these two components, three efficient pruning strategies are devised, thus avoiding the need to scan the whole dataset for generating the transferred dataset and also reducing the times of dominance checking. 3) The correctness guarantees of the three pruning strategies are provided. 4) A series of experiments are performed to evaluate our approach and the experimental results demonstrate that our approach is effective and efficient. The remainder of this paper is organized as follows. The related works of the dynamic skylines are reviewed in Section 2. Then, the preliminaries of this paper are introduced in Section 3, including all the terms used and the kernel component of our solution. The proposed approach is detailed in Section 4. After that, the experiment results and performance analyses of our approach are described in Section 5. Finally, Section 6 concludes this work.

2 Related Works

Different from the original skyline query considering data points with static attributes, the dynamic skyline query is related to a variety of skyline queries considering data points with *dynamic attributes* [4], which means the attributes of each data point are dynamically calculated according to the queries specified by distinct users. The variant skyline queries regarding dynamic attributes can be roughly categorized into the *multiple-queries driven* type and *single-query driven* type. Given a set of data points $P = \{p_1, p_2, \ldots, p_m\}$ and a set of query points $Q = \{q_1, q_2, \ldots, q_k\}$, each data point p in the multiple-queries driven type is represented by an attribute vector $\langle dist(p, q_1), dist(p, q_2), \ldots, dist(p, q_k)\rangle$, where $dist(\cdot, \cdot)$ is a distance function. Those data points whose attribute vectors are not dominated by the attribute vectors of the other data points are returned as skyline results. Focusing on distinct distance functions, a series of literatures in the multiple-queries driven type are proposed, including [14][6][4][7]. Sharifzadeh and Shahabi propose the *spatial skyline queries* in [14], in which *dist*(·, ·) is the Euclidean Distance. Deng et al. propose the *multi-source skyline query* in [DZS09] and consider the distance measure as the distance of the shortest path in road networks. [4] and [7] solve the *metric skyline query*, where *dist*(·, ·) is a metric distance.

On the other hand, the concept of the dynamic skyline [11][5][13] belonging to the single-query driven type is first mentioned in [11]. Papadias et al. briefly describe how to extend BBS [11] to solve the dynamic skyline query. In [5], Dellis and Seeger use the semantics of the dynamic skyline to introduce the *reverse skyline query* but do not propose any solutions to solve the dynamic skyline queries. Given a point q, the reverse skyline query [5] returns the data points whose corresponding dynamic skylines contain q. Sacharidis et al. propose a caching mechanism in [13], using the information of the past queries to speed up the dynamic skyline processing regarding a new query. Different from the dynamic skyline considering a query point as described in [5][13], we make the first attempt to study a new problem on computing dynamic skylines considering range queries in this paper. The BBS algorithm [11] is based on R-tree and using the concept of nearest-neighbor search. To decide the expanding order of MBRs in the R-tree, BBS needs to compute the *mindist* for each entity, i.e. either a data point or an MBR. To compute the original skyline, the *mindist* of a data point is defined as the sum of its all dimensions while the *mindist* of an MBR is equal to the *mindist* of the lower-left corner point of the MBR. It is difficult to extend BBS to compute the dynamic skylines regarding range queries since the *mindist* of an MBR is ambiguous and undefined, with respect to a range query.

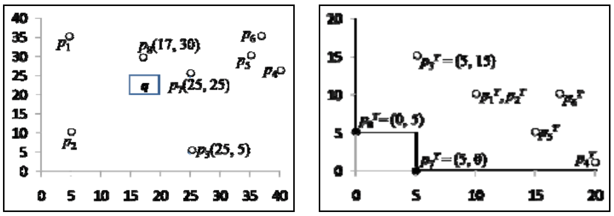
3 Preliminaries

In this section, the problem of computing the dynamic skyline regarding a range query is formally defined. Thereafter, the kernel component used in our solution to this problem, i.e. *multidirectional Z-order curves*, is also described in this section, followed by a brief introduction to *Z-order curve* [10].

3.1 Problem Formulation

Given an n-dimensional dataset D and a range query q ($[q_1, q_1'], [q_2, q_2'], \ldots, [q_n, q_n']$), where $[q_i, q_i']$ is an interval representing the user interests in the i^{th} dimension, $\forall\, i = 1$ to n, the dynamic skyline query regarding q returns the data points from D, not *dynamically dominated* by any other data points, with respect to q.

Definition 1 (*Dynamic Domination*): Give an n-dimensional dataset D and a range query q ($[q_1, q_1'], [q_2, q_2'],\ldots, [q_n, q_n']$), a data point $p_1 \in D$ is defined to dynamically dominate another data point $p_2 \in D$, with respect to q iff 1) $\underset{x_i\in[q_i,\, q_i']}{Min} | p_1[i]-x_i | \leq \underset{y_i\in[q_i,\, q_i']}{Min} | p_2[i]-y_i |$, $\forall\ i = 1$ to n, and 2) $\exists\ j$, $1 \leq j \leq n$, such that $\underset{x_j\in[q_j,\, q_j']}{Min} | p_1[j]-x_j | < \underset{y_j\in[q_j,\, q_j']}{Min} | p_2[j]-y_j |$, where $p[i]$ is the i^{th} dimension of a point p and $\underset{x_i\in[q_i,\, q_i']}{Min} | p[i]-x_i |$ is the minimum distance between $p[i]$ and the i^{th} dimension of q. ■



(*a*) the original space (*b*) the transferred space

Fig. 3. The dynamic skyline regarding a range query *q*

Example 1: A 2-dimensional dataset with 8 data points is shown in Fig. 3. Let a range query *q* be ([15, 20], [20, 25]). According to the minimum distance function mentioned in Definition 1, the data point p_8 = (17, 30) can be transferred to (|17 – 17|, |30 – 25|) = (0, 5), regarding *q*. Moreover, p_7 and p_3 can be respectively transferred to (|25 – 20|, |25 – 25|) = (5, 0) and (|25 – 20|, |5 – 20|) = (5, 15), regarding *q*. p_3 is dynamically dominated by p_8 and p_7 since each transferred dimension of p_3 is larger than that of p_8 and that of p_7. By the definition of dynamic domination, the results of the dynamic skyline query are p_8 and p_7 since the transferred dimensions of p_8 (the transferred p_8 for short in the following discussion), i.e. (0, 5) and those of p_7 cannot be dynamically dominated by any other transferred data points, with respect to *q*. ■

3.2 Data Structures Used in Our Solution

To solve this problem, a straightforward solution works as follows. All of the data points are transferred using the minimum distance function according to *q* and then, the existing skyline algorithms, e.g. SFS [3], are applied to find the dynamic skyline from the transferred dataset. It is costly to scan the whole dataset to generate the transferred dataset and then again scan the transferred dataset for dominance checking. In this paper, an efficient algorithm based on the *grid index* and newly designed *multidirectional Z-order curves* is proposed to solve this problem, avoiding the need to generate the whole transferred dataset and also reducing the times of dominance checking. Next, we describe the data structures used in our solution.

The grid index. The grid index is a *space-driven* indexing method opposed to the *data-driven* indexing methods such as *R*-tree, partitioning the space into grids in advance. More specifically, each dimension of the *n*-dimensional space is partitioned into *b blocks*, each associated with an equal domain range of *r*. The block *i* is related to the domain of $[ir, (i + 1)r)$. The whole space can therefore be regarded as being partitioned into b^n *n*-dimensional *cells* (cells for short in the following discussion). If a data point falls into a cell, the information of the data point is kept in this cell. Moreover, each cell has its own *coordinates* related to the block in each dimension. For example, if a 2-dimensional cell is associated with the blocks 0 and 1 in the first and second dimensions, respectively, the coordinates of this cell are (0, 1).

Definition 2 (*Query Cells*): Given a range query *q* ($[q_1, q_1']$, $[q_2, q_2']$, …, $[q_n, q_n']$) in a grid-indexed *n*-dimensional space, the cells whose coordinates (c_1, c_2, …, c_n) satisfy that $c_i \in [\lfloor q_i / r \rfloor, \lfloor q_i' / r \rfloor]$, $\forall$ *i* = 1 to *n*, are defined as query cells. ■

Definition 3 (*Pivot Cells*): Given a range query q ($[q_1, q_1']$, $[q_2, q_2']$, …, $[q_n, q_n']$) in a grid-indexed n-dimensional space, the cells whose coordinates $(c_1, c_2, …, c_n)$ satisfy the conditions: 1) in only one dimension, say i, $c_i \notin [\lfloor q_i / r \rfloor, \lfloor q_i' / r \rfloor]$ and 2) in the other dimensions j, $\forall j = 1$ to n and $j \neq i$, $c_j \in [\lfloor q_j / r \rfloor, \lfloor q_j' / r \rfloor]$, are defined as pivot cells. ■

Definition 4 (*Query-imported Orthants*): Given a range query q in a grid-indexed n-dimensional space, the space can be partitioned into 2^n regions by the query cells and pivot cells. These regions are defined as query-imported orthants (q-orthants). ■

Example 2: A grid-indexed 2-dimensional space is shown in Fig. 4. Each dimension of the space is partitioned into 8 blocks. Accordingly, the space can be regarded as being partitioned into 64 cells. Moreover, the coordinates of the cell C are (7, 0). Since q overlaps four cells, the cells with coordinates equal to (3, 4), (3, 5), (4, 4), and (4, 5) are query cells. For a concise presentation, we use a range form of ([3, 4], [4, 5]) to denote these cells. Moreover, the pivot cells with respect to q are the cells with coordinates equal to ([0, 2], [4, 5]), ([5, 7], [4, 5]), ([3, 4], [0, 3]), and ([3, 4], [6, 7]). According to the query cells and pivot cells, the space is partitioned into 4 q-orthants. Notice that, the cells in the q-orthants are neither query cells nor pivot cells. ■

Z-order curve. *Z-order curve* [10] is a *space-filling curve*, commonly used in mapping data points in a multidimensional space to a one dimensional space by representing each data point using a unique number named *Z-address* and then sorting the data points into an increasing order of Z-address. More specifically, each dimension of a data point in an n-dimensional space is represented using a *binary sequence*, e.g. $5 = (101)_2$. Accordingly, a data point is related to n binary sequences, e.g. the coordinates of a data point are $(5, 4) = (101, 100)_2$. The Z-address of a data point is a sequence of bits, generated by interlacing the bits from the n binary sequences related to the data point, e.g. the Z-address of (5, 4) is (110010), where the 1st, 3rd, and 5th bits come from $(101)_2$ and the 2nd, 4th, and 6th bits come from $(100)_2$.

In our solution, cells rather than data points are represented using Z-addresses. As mentioned above, since each dimension of the space is partitioned into b blocks, we use v bits to represent a block of a dimension, where $v = \log_2 b$. A Z-address of a cell therefore consists of $n \times v$ bits, within a domain of $[0, 2^{nv} - 1]$. Let the coordinates of a cell be represented using binary sequences. Then, the i^{th} bit of the Z-address of a cell comes from the $\lceil i/n \rceil^{th}$ bit of the $(i \bmod n)^{th}$ coordinate[1] of the cell. For example, in a 2-dimensional space, let the coordinates of a cell, represented using binary sequences, be $(x_1x_2 … x_v, y_1y_2 … y_v)_2$, where x_i and $y_i \in \{0, 1\}$, $\forall i = 1$ to v. The Z-address of the cell is $(x_1y_1x_2y_2…x_vy_v)$. Z-order curve has a good property, i.e. *monotonic ordering* [9], which can be used in efficient skyline computation.

Property 1 (*Monotonic Ordering of Z-order curve*): While being applied to the traditional skyline computation, Z-order curve formed by sorting the cells using an increasing order of Z-address is monotonic such that a data point in a cell with a former order cannot be dominated by the data points in the cells with the latter order. ■

Multidirectional Z-order curves. In reality, the original skyline computation can be regarded as the dynamic skyline computation with respect to a query point q equal to

[1] If $0 \equiv i \bmod n$, it indicates the n^{th} coordinate of the cell.

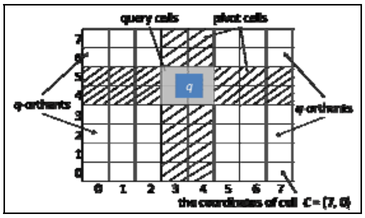


Fig. 4. A grid-indexed 2-dimensional space

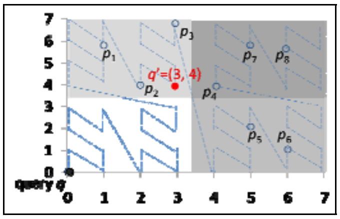


Fig. 5. The original *Z*-order curve

the *origin*, as shown in Fig. 5. Although the monotonic ordering of *Z*-order curve in dominance can be used in efficient skyline computation, the benefit of *Z*-order curve is excluded once the query is not equal to the origin. As shown in Fig. 5, although p_4 = (4, 4) is accessed latter than p_1 = (1, 6) according to the *Z*-addresses, the transferred p_4 = (1, 0) dynamically dominates the transferred p_1 = (2, 2), with respect to the query point *q*' equal to (3, 4). It means that the property of the monotonic ordering of the original *Z*-order curve may be destroyed while being applied to dynamic skyline computation. We therefore design a variant of *Z*-order curve, i.e. *multidirectional Z-order curves* for dynamic skyline computation regarding a range query.

Definition 5 (*MZ-address and MZ-imported coordinates of a cell*): Given a range query q ([q_1, q_1'], [q_2, q_2'], …, [q_n, q_n']) in a grid-indexed n-dimensional space and a cell with coordinates (c_1, c_2,…, c_n), where each coordinate c_i is represented as a binary sequence $(b_{i1}b_{i2}\ldots b_{iv})_2$, the binary sequence $(b_{i1}b_{i2}\ldots b_{iv})_2$ of c_i is transferred to an opposite sequence, i.e. $(\overline{b_{i1}}\overline{b_{i2}}..\overline{b_{iv}})_2$, if $r \times (c_i + 1) \le q_i$, $\forall$ i = 1 to n. Let the decimal representation of the opposite sequence corresponding to c_i be denoted $\overline{c_i}$. The *MZ*-address of the cell is then defined as a sequence of bits, interlacing the bits from each coordinate of the cell, represented as a (transferred or non-transferred) binary sequence. Moreover, the *MZ*-imported coordinates (*MZI*-coordinates) of a cell are defined as pseudo n-coordinates (d_1, d_2, …, d_n), where $d_i = \overline{c_i}$ if $r \times (c_i + 1) \le q_i$ and $d_i = c_i$, otherwise. ■

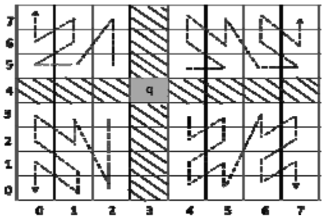


Fig. 6. Multidirectional *Z*-order curves in *q*-orthants

Example 3: Let *b* and *r* be 8 and 10, respectively. Give a cell with coordinates (2, 4) = $(010, 100)_2$ and a query *q* equal to ([30, 35], [20, 25]), since (2 + 1) × 10 ≤ 30, the binary sequence of 2, i.e. $(010)_2$, is transferred to $(101)_2$. Accordingly, the *MZ*-address of this cell is (110010), different from its original Z-address equal to (011000). Moreover, the *MZI*-coordinates of the cell, generated from $(101, 100)_2$ are (5, 4). ■

As mentioned in Definition 4, the space is partitioned into 2^n *q*-orthants, with respect to *q*. By independently sorting the cells in each *q*-orthant into an increasing order of

MZ-address, *multidirectional Z-order curves* (*MZ*-order curves) are generated. An example of *MZ*-order curves is shown in Fig. 6. We use an example of the one dimensional cells to express the semantics of generating an opposite sequence related to a coordinate of a cell. Let b be 8. Then, the coordinate of a cell is between $0 = (000)_2$ and $7 = (111)_2$. If a range query q $[q_1, q_1']$ exactly falls into the cell 3, the other cells can be separated into two categories, i.e. the cells with coordinates larger or smaller than 3. To the cells with coordinates larger than 3, i.e. the cells 4, 5, 6, and 7, the larger the coordinate is, the further from q the corresponding cell is. Alternatively, to the cells with coordinates smaller than 3, i.e. the cells 0, 1, and 2, the smaller the coordinate is, the further from q the corresponding cell is. In other words, we prefer to earlier access the cells with larger coordinates, i.e. the access order of 2, 1, and 0, if the corresponding coordinates are smaller than 3. Using the opposite sequences to represent these cells, e.g. 2 by $(101)_2$, 1 by $(110)_2$ and 0 by $(111)_2$, and then sorting the bit sequences into an increasing order of values can achieve the goal of inverse accessing. Accordingly, following the property of the monotonic ordering of the original *Z*-order curve, *MZ*-order curves also have the similar property.

Property 2 (*Monotonic Ordering of MZ-order curves*): If the cells in a q-orthant are accessed according to an increasing order of *MZ*-address, the data points in a cell with a former order cannot be dynamically dominated by the other data points in the cells with the latter order, with respect to a range query q. ■

4 Dynamic Skyline Processing

Our solution to the problem of computing the dynamic skyline regarding a range query is proposed in this section. We describe the principle of the pruning strategies used in our solution in Subsection 4.1 and present the algorithm in Subsection 4.2.

4.1 Principle of Pruning Strategies

Lemma 1: Given a range query q ($[q_1, q_1']$, $[q_2, q_2']$, …, $[q_n, q_n']$) in a grid-indexed n-dimensional space, if a data point p ($p[1]$, $p[2]$, …, $p[n]$) in a query cell with coordinates = (c_1, c_2, …, c_n) has only one transferred dimension, say i, not equal to 0, i.e. $\underset{x_i \in [q_i, q_i']}{Min} |p[i]-x_i| \neq 0$, the data points in the cells whose coordinates = (d_1, d_2, …, d_n) satisfying either 1) $d_i > c_i$ or $0 \le d_i < \lfloor q_i/r \rfloor - 1$ if $c_i = \lfloor q_i'/r \rfloor$ or 2) $d_i < c_i$ or $d_i > \lfloor q_i'/r \rfloor + 1$ if $c_i = \lfloor q_i/r \rfloor$ can be dynamically dominated by p, with respect to q. ■

Proof: Since $\underset{x_i \in [q_i, q_i']}{Min} |p[i]-x_i| \neq 0$ and $\underset{x_j \in [q_j, q_j']}{Min} |p[j]-x_j| = 0$, $\forall\ j = 1$ to n and $j \neq i$, p must dynamically dominate any data point whose transferred i dimension is larger than $\underset{x_i \in [q_i, q_i']}{Min} |p[i]-x_i|$, with respect to q. If $c_i = \lfloor q_i'/r \rfloor$, the transferred i dimensions of any data points in the cells with coordinates = (d_1, d_2, …, d_n) and $d_i > c_i$ or $0 \le d_i < \lfloor q_i/r \rfloor - 1$ must be larger than $\underset{x_i \in [q_i, q_i']}{Min} |p[i]-x_i|$. Moreover, if $c_i = \lfloor q_i/r \rfloor$, the transferred i dimensions of any data points in the cells with coordinates = (d_1, d_2, …, d_n) and $d_i < c_i$ or $d_i > \lfloor q_i'/r \rfloor + 1$ must be larger than $\underset{x_i \in [q_i, q_i']}{Min} |p[i]-x_i|$. Therefore, the data points in the cells whose coordinates = (d_1, d_2, …, d_n) satisfying either 1) $d_i > c_i$ or $0 \le d_i < \lfloor q_i/r \rfloor - 1$ if $c_i = \lfloor q_i'/r \rfloor$ or 2) $d_i < c_i$ or $d_i > \lfloor q_i'/r \rfloor + 1$ if $c_i = \lfloor q_i/r \rfloor$ can be dynamically dominated by p, with respect to q. ■

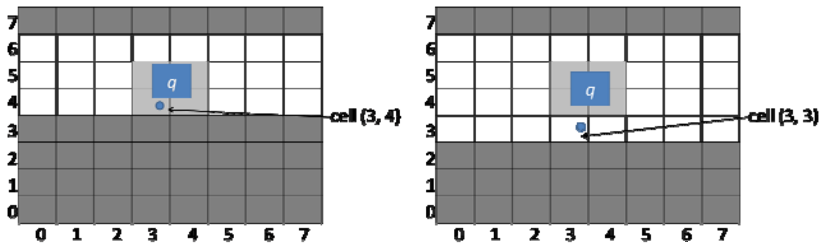


Fig. 7. Illustrations of Lemmas 1 and 2

Lemma 2: Given a range query q ($[q_1, q_1']$, $[q_2, q_2']$, …, $[q_n, q_n']$) in a grid-indexed n-dimensional space, if a data point p ($p[1]$, $p[2]$, …, $p[n]$) in a pivot cell with coordinates = (c_1, c_2, …, c_n) has only one transferred dimension, say i, not equal to 0, i.e. $\underset{x_i \in [q_i,\, q_i']}{Min} |p[i] - x_i| \neq 0$, where $1 \leq i \leq n$, the data points in the cells whose coordinates = (d_1, d_2, …, d_n) satisfying either 1) $d_i > c_i$ or $0 \leq d_i < \lfloor q_i / r \rfloor + \lfloor q_i' / r \rfloor - c_i$ if $c_i - \lfloor q_i' / r \rfloor > 0$ or 2) $d_i < c_i$ or $d_i > \lfloor q_i / r \rfloor + \lfloor q_i' / r \rfloor - c_i$ if $c_i - \lfloor q_i / r \rfloor < 0$ can be dynamically dominated by p, with respect to q. ■

Proof: Since $\underset{x_i \in [q_i,\, q_i']}{Min} |p[i] - x_i| \neq 0$ and $\underset{x_j \in [q_j,\, q_j']}{Min} |p[j] - x_j| = 0$, $\forall\, j = 1$ to n and $j \neq i$, p must dynamically dominate any data point whose transferred i dimension is larger than $\underset{x_i \in [q_i,\, q_i']}{Min} |p[i] - x_i|$, with respect to q. If $c_i - \lfloor q_i' / r \rfloor > 0$, the transferred i dimensions of any data points in the cells with coordinates = (d_1, d_2, …, d_n) and $d_i > c_i$ or $0 \leq d_i < \lfloor q_i / r \rfloor + \lfloor q_i' / r \rfloor - c_i$ must be larger than $\underset{x_i \in [q_i,\, q_i']}{Min} |p[i] - x_i|$. Moreover, if $c_i - \lfloor q_i / r \rfloor < 0$, the transferred i dimensions of any data points in the cells with coordinates = (d_1, d_2, …, d_n) and $d_i < c_i$ or $d_i > \lfloor q_i / r \rfloor + \lfloor q_i' / r \rfloor - c_i$ must be larger than $\underset{x_i \in [q_i,\, q_i']}{Min} |p[i] - x_i|$. Therefore, the data points in the cells whose coordinates = (d_1, d_2, …, d_n) satisfying either 1) $d_i > c_i$ or $0 \leq d_i < \lfloor q_i / r \rfloor + \lfloor q_i' / r \rfloor - c_i$ if $c_i - \lfloor q_i' / r \rfloor > 0$ or 2) $d_i < c_i$ or $d_i > \lfloor q_i / r \rfloor + \lfloor q_i' / r \rfloor - c_i$ if $c_i - \lfloor q_i / r \rfloor < 0$ can be dynamically dominated by p, with respect to q. ■

Example 4: As shown in Fig. 7(a), if a data point falls into the cell with coordinates = (3, 4) and moreover, the x-coordinate of the data point satisfies the corresponding requirement of the range query q, (i.e. the transferred x-coordinate = 0), all of the data points in the cells ([0, 7], [0, 3]) and the cells ([0, 7], 7) can be dynamically dominated by this data point, with respect to q according to Lemma 1. Moreover, as shown in Fig. 7(b), if a data point falls into the cell with coordinates (3, 3) and the corresponding transferred x-coordinate is equal to 0, then all of the data points in the cells ([0, 7], [0, 2]) and the cells ([0, 7], 7) can be dynamically dominated by this data point, with respect to q according to Lemma 2. ■

Observation 1: Suppose that the coordinates of two cells C and D in a grid-indexed n-dimensional space are (c_1, c_2, …, c_n) and (d_1, d_2, …, d_n), respectively, and moreover, the query is equal to the origin (i.e. equivalent to the original skylines). Then, any data points in C can dominate any data points in D if $c_i < d_i$, $\forall\, i = 1$ to n. ■

Observation 1 comes from that the block j of a coordinate of a cell is related to the domain of $[jr, (j + 1)r)$. Since the domain of c_i is $[c_i r, (c_i + 1)r)$ and that of d_i is $[d_i r, (d_i +$

1)r), the i dimension of any data point in C must be smaller than that of any data point in D, if $c_i < d_i$. Accordingly, any data points in C can dominate any data points in D if $c_i < d_i$, $\forall$ i = 1 to n. To *MZI*-coordinates, we have the similar property.

Lemma 3: Suppose that the range query q is ($[q_1, q_1']$, $[q_2, q_2']$, …, $[q_n, q_n']$) and moreover, the *MZI*-coordinates of two cells C and D in the same q-orthant are (c_1, c_2, …, c_n) and (d_1, d_2, …, d_n), respectively. Then, any data points in C can dynamically dominate any data points in D, with respect to q if $c_i < d_i$, $\forall$ i = 1 to n. ■

Proof (by contradiction): We assume that a data point s in D cannot be dynamically dominated by a data point t in C. Since $C \neq D$, at least in one dimension, say i, $\underset{x_i \in [q_i,\, q_i']}{Min} | s[i] - x_i | < \underset{y_i \in [q_i,\, q_i']}{Min} | t[i] - y_i |$. Since C and D are in the same q-orthant, either the condition of $s[i] < q_i$ and $t[i] < q_i$ or the condition of $s[i] > q_i'$ and $t[i] > q_i'$ holds.

Case 1 ($s[i] < q_i$ and $t[i] < q_i$): $|s[i] - q_i| = \underset{x_i \in [q_i,\, q_i']}{Min} | s[i] - x_i | < \underset{y_i \in [q_i,\, q_i']}{Min} | t[i] - y_i | = |t[i] - q_i|$. We obtain that $q_i - s[i] < q_i - t[i]$, which implies $\lfloor t[i]/r \rfloor < \lfloor s[i]/r \rfloor$. From $\lfloor t[i]/r \rfloor < \lfloor s[i]/r \rfloor$, $s[i] < q_i$, and $t[i] < q_i$, we can infer that $c_i > d_i$, where a contradiction occurs.

Case 2 ($s[i] > q_i'$ and $t[i] > q_i'$): $|s[i] - q_i'| = \underset{x_i \in [q_i,\, q_i']}{Min} | s[i] - x_i | < \underset{y_i \in [q_i,\, q_i']}{Min} | t[i] - y_i | = |t[i] - q_i'|$. We obtain that $s[i] < t[i]$, which implies $\lfloor s[i]/r \rfloor < \lfloor t[i]/r \rfloor$. Since $s[i] > q_i'$ and $t[i] > q_i'$, $d_i = \lfloor s[i]/r \rfloor < \lfloor t[i]/r \rfloor = c_i$, where a contradiction occurs.

Therefore, any data points falling into C can dynamically dominate any data points falling into D, with respect to q if $c_i < d_i$, $\forall$ i = 1 to n. ■

Example 5: As shown in Fig. 8, the original coordinates of C are (2, 2) and its *MZI*-coordinates are (5, 5). Moreover, any data points in C can dynamically dominate any data points in the cells with original coordinates ([0, 1], [0, 1]), since their corresponding *MZI*-coordinates are (6, 6), (6, 7), (7, 6), and (7, 7) with both coordinates smaller than (5, 5). ■

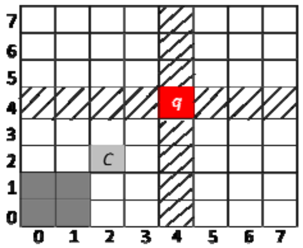


Fig. 8. An illustration of Lemma 3

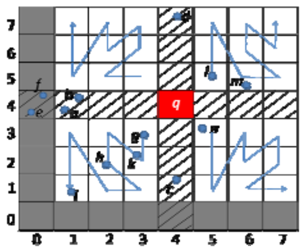


Fig. 9. An illustration of the running *MDS*

4.2 The MDS Algorithm

By using the pruning strategies coming from Lemmas 1, 2 and 3, our algorithm of computing the dynamic skyline regarding a range query, named *MDS* (**M**ulti **D**irectional **S**canning) is devised. The pseudo code of *MDS* is shown in Algorithm 1 and we describe the main steps of *MDS* as follows. Let the range query q specified by users be ($[q_1, q_1']$, $[q_2, q_2']$, …, $[q_n, q_n']$).

Step 1: We first identify the query cells regarding *q* and then, check to see whether any data points in the query cells satisfy *q*. If yes, return these data points as results. Otherwise, all of the data points in the query cells are transferred using the minimum distance function mentioned in Definition 1 and then, placed in the *candidate point set*. If a data point in a certain query cell has only one transferred dimension not equal to 0, some corresponding cells can be pruned according to Lemma 1, which means that the data points in the cells pruned need not be checked again.

Algorithm 1 (The MDS Algorithm)

Input: a dataset *D* in a grid-indexed *n*-dimensional space and a range query *q*

Output: the dynamic skyline regarding *q*

Variable: the candidate point set *CandP* and the candidate cell set *CandC* *CandP* = ϕ and *CandC* = ϕ, initially

```
1. Find the data points satisfying $q$ and return those data points as exact results.
2. If there are no data points satisfying $q$
3.     For each query cell $C$
4.         If there is a data point $p$ in $C$ with only one transferred dimension $\neq 0$
5.             Prune the corresponding cells using Lemma 1
6.     Transfer all the data points in $C$ and place them in $CandP$
7.     For each un-pruned pivot cell C
8.         If there is a data point $p$ in $C$ with only one transferred dimension $\neq 0$
9.             Prune the corresponding cells using Lemma 2
10.        Transfer all the data points in $C$ and place them in $CandP$
11.    For each $q$-orthant $Q$
12.        Compute the corresponding $MZ$-address and $MZI$-coordinates for each cell in $Q$ and sort the cells into
           an increasing order of $MZ$-address
13.        For each cell $C$ with $MZI$-coordinates = $(c_1, c_2, \ldots, c_n)$ in $Q$
14.            If ($C$ is empty or a cell $D$ with $MZI$-coordinates = $(d_1, d_2, \ldots, d_n)$ in $CandC$ exists such that $d_i < c_i\ \forall$
               $i = 1$ to $n$)
15.                $C$ is pruned using Lemma 3
16.            Else
17.                $C$ is placed in $CandC$
18.        For each cell $C$ in $CandC$
19.            Transfer all the data points in $C$ and place them in $CandP$
20.        $CandC = \phi$
21.    Apply a existing skyline algorithm, e.g., SFS, to find the skyline in $CandP$ and return the corresponding
       skyline points
```

Step 2: For each un-pruned pivot cell, if it is non-empty, all of the data points in the pivot cell are transferred and placed in the candidate point set. Again, if a data point in a certain pivot cell has only one transferred dimension not equal to 0, some corresponding cells can be pruned using Lemma 2. Notice that the pivot cells with adjacent coordinates are sequentially accessed and moreover, the accessing order follows an increasing order of the distance to the query. That is, the pivot cells closer to the query cells are earlier accessed. This is because the pivot cells closer to the query cells may have the better pruning capability, thus early terminating the accessing.

Step 3: For the cells in each *q*-orthant, we compute their corresponding *MZ*-addresses and then sort them into an increasing order of *MZ*-address. After that, each cell is sequentially accessed. The corresponding *MZI*-coordinates of the first accessed and non-empty cell are placed in the *candidate cell set*. During the process of sequentially accessing the cells, if a cell *C* with *MZI*-coordinates = $(c_1, c_2, \ldots, c_n)$ is empty or we can find the other cell, say *D*, with *MZI*-coordinates = $(d_1, d_2, \ldots, d_n)$ from the candidate cell set such that $d_i < c_i\ \forall\ i = 1$ to *n*, the cell *C* can be pruned using Lemma 3. Otherwise, *C* with its *MZI*-coordinates is placed in the candidate cell set. After all of the cells in the current *q*-orthant are either accessed or pruned, all of the data points in the cells kept in the candidate cell set are transferred and placed in the candidate point set. Then, we clear the candidate cell set and turn to process another *q*-orthant.

Step 4: After the cells in all *q*-orthants are processed, we can apply the existing skyline algorithms, e.g. SFS[2] [3], to find the dynamic skyline regarding *q* from the transferred data points in the candidate point set.

Example 6: As shown in Fig. 9, the cells with coordinates = ([0, 7], 0) and (0, [0, 7]) can be pruned by the cells with coordinates = (4, 1) (or (4, 7)) and (1, 4), respectively. Moreover, after checking the pivot cells, the candidate point set = {*a*, *b*, *c*, *d*}. While checking the lower-left *q*-orthant, the cell with *MZI*-coordinates = (5, 5), i.e. the cell where *h* falling into is pruned since the cell with *MZI*-coordinates = (4, 4), i.e. the cell where *g* falling into is kept in the candidate cell set. Moreover, the cell with *MZI*-coordinates = (6, 6), i.e. the cell where *i* falling into, is also pruned due to the same reason. After all the cells in the lower-left *q*-orthant are processed, the candidate cell set = {the cells with *MZI*-coordinates = (4, 4) and (4, 5)}. Accordingly, the candidate point set becomes {*a*, *b*, *c*, *d*, *g*, *k*}. ■

Using *MDS* to find the dynamic skyline regarding a range query can avoid the need to generate the complete transferred dataset by scanning the whole dataset, thus reducing the times of final dominance checking. Moreover, the other advantage of *MDS* is that it is easy to be parallelized. Since the cells in each *q*-orthant are processed independently, *MDS* can be easily performed in the system with multi-processors.

5 Performance Evaluation

In this section, a series of experiments are performed to evaluate our approach and moreover, the experiment results are also presented and analyzed.

5.1 Experiment Setup

Following [13], the test datasets used in the experiments are synthetic datasets generated using the data generator from [12]. The distributions of the test datasets [13] are shown in Table 1. To the best of our knowledge, there are no existing approaches specially focusing on the dynamic skylines regarding range queries. Accordingly, since MDS is the first work and applies the SFS algorithm [3] to find the skyline in the final partial transferred dataset in our implementation, MDS is therefore compared with a naïve approach (Naïve) transferring the whole dataset according to the range query and then applying the SFS algorithm to find the skyline. Both of the two algorithms are implemented in C and performed on a PC with the Intel Core 2 Quad 2.66GHz CPU, 2GB of main memory, and under the Ubuntu v9.10 Operating system. In order to simulate no data points exist to fully satisfy the range queries in the experiments, after a range query is issued, the exact results of the corresponding query are eliminated from the datasets. Moreover, the running time of each approach shown in the following experiment results is an average processing time among processing 50 range queries. Notice that, the same as [13], the issued queries are randomly generated, with a distribution identical to the distribution of

[2] The other approaches on skyline computation, without indexing, can substitute SFS.

the corresponding test dataset. On the other hand, since setting the block number *b* to partition each dimension for index construction may affect the performance of MDS, we therefore use distinct *b* in the experiments. $b = 32$ while $d = 2$ to 4 and $b = 16$ while $d = 5$. The experiment factors are summarized in Table 2.

Table 1. The distributions of the test datasets

distribution	description
Independent	The attributes of each data point are generated uniformly and randomly.
Correlated	If a data point has an attribute with low value, the other attributes of this data point may likely have low values as well.
Anti-Correlated	If a data point has an attribute with a low value, the other attributes of this data point may likely have high values.

Table 2. Experimental factors

Factor	Range	description
d	2 ~ 5	the number of dimensions of each data point
# of data points	500K ~ 2M	the sizes of the datasets
b	16/32	the number of blocks in each dimension

5.2 Experiment Results

The experiment results regarding the test datasets with different distributions are shown in Figs. 10-13. Since the larger sizes of datasets cause the longer processing time, the processing time of MDS and that of Naïve are increased as the increasing of the sizes of datasets. Moreover, as shown in Figs. 10-12, the performance of MDS is better than that of Naïve in all of the cases. As shown in Fig. 13, we can find that the pruning power of MDS is very strong; in general, the pruning strategies used in MDS can prune over 90% data points from the test datasets in most of the cases, making the number of data points needing to be transferred few. In addition, this also greatly reduces the times of dominance checking, therefore outperforming Naïve.

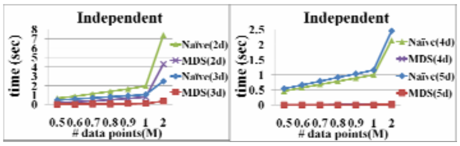


Fig. 10. The independent datasets

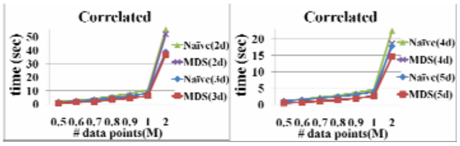


Fig. 11. The correlated datasets

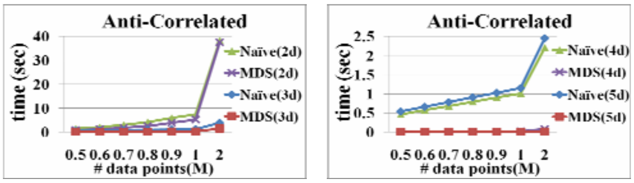
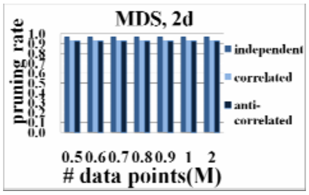
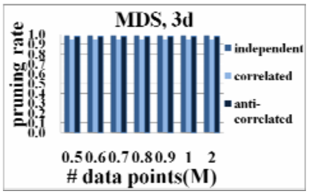


Fig. 12. The anti-correlated datasets

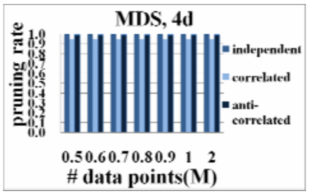
It is worth mentioning that in all the distributions of the test datasets, the processing time of either MDS or Naïve with respect to the higher dimensions, e.g. 5, is shorter than that with respect to the lower dimensions, e.g. 2. This is because the sizes of skylines with respect to the lower dimensions are much larger than those with respect to the higher dimensions, and the performance of SFS (both used in MDS and Naive) is highly affected by the size of a skyline. Conceptually, the sizes of skylines may increase with the increasing of the number of dimensions since a data point with higher dimensions may not be dominated with the higher probability (referred to the definition of dominance). However, in the experiments, since the domain range of each dimension is identical, the datasets with lower dimensions are much denser than those with higher dimensions under the condition of the same sizes of datasets. Therefore, the datasets with lower dimensions may have skylines with the larger sizes, making the processing time regarding the lower dimensions higher than that regarding the higher dimensions.



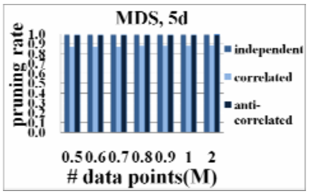
(*a*) 2 dimensions



(*b*) 3 dimensions



(*c*) 4 dimensions



(*d*) 5 dimensions

Fig. 13. The pruning rate of *MDS*

Moreover, although MDS prunes most of the data points (i.e. over 90%) in most of the cases, in some cases regarding the correlated distribution as shown in Figure 11, the performance of MDS is not greatly better than that of Naïve (say 10 times better). Again, this is affected by the sizes of skylines. The sizes of skylines in the test datasets with the correlated distribution are much larger than those regarding the other distributions. Under the cases with the larger sizes of skylines, the performance bottleneck of MDS is at the step of dominance checking (step 4) since the SFS approach is applied to the candidate checking of MDS.

6 Conclusions

In this paper, we make the first attempt to solve a new problem on dynamic skyline computation regarding a range query. Moreover, the semantics of this novel skyline query is also well explained. Given a range query, the dynamic skyline query returns the data points not dynamically dominated by any other data points, with respect to the given query. Since it is costly to scan the whole dataset for transferring all the data points from the original space to the other space according to the range query and then scan the whole transferred dataset for dominance checking, we propose an efficient approach, i.e. *MDS* based on the gird index and newly designed multidirectional *Z*-order curves to avoid generating the whole transferred dataset. The experiment results demonstrate that *MDS* is very efficient, outperforming a naïve approach using SFS only. Moreover, since the three pruning strategies are used in *MDS*, most of the data points i.e. over 90% need not be transferred to the other space for dominance checking. In addition to high efficiency, the other main advantage of *MDS* is that it is easy to perform *MDS* in the system with multi-processors since the cells in each *q*-orthant can be independently processed.

References

[1] Borzsonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of the 17th International Conference on Data Engineering, ICDE 2001, Heidelberg, Germany, pp. 421–430 (2001)

[2] Bartolini, I., Ciaccia, P., Patella, M.: SaLSa: Computing the skyline without scanning the whole sky. In: Proceedings of the 2006 ACM International Conference on Information and Knowledge Management, CIKM 2006, Arlington, Virginia, USA, pp. 405–414 (2006)

[3] Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: Proceedings of the 19th International Conference on Data Engineering, ICDE 2003, Bangalore, India, pp. 717–719 (2003)

[4] Chen, L., Lian, X.: Efficient processing of metric skyline queries. IEEE Trans. Knowl. Data Eng. 21(3), 351–365 (2009)

[5] Dellis, E., Seeger, B.: Efficient Computation of Reverse Skyline Queries. In: Proceedings of the 33nd International Conference on Very Large Data Bases, VLDB 2007, Vienna, Austria, pp. 291–302 (2007)

[6] Deng, K., Zhou, X., Shen, H.T.: Multi-source skyline query processing in road networks. In: Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, Istanbul, Turkey, pp. 796–805 (2007)

[7] Fuhry, D., Jin, R., Zhang, D.: Efficient skyline computation in metric space. In: Proceedings of the 12th International Conference on Extending Database Technology, EDBT 2009, Saint-Petersburg, Russia, pp. 1042–1051 (2009)

[8] Godfrey, P., Shipley, R., Gryz, J.: Maximal vector computation in large data sets. In: Proceedings of the 31st International Conference on Very Large Data Bases, VLDB 2005, Trondheim, Norway, pp. 229–240 (2005)

[9] Lee, K.C.K., Zheng, B., Li, H., Lee, W.C.: Approaching the skyline in Z order. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007, Vienna, Austria, pp. 279–290 (2007)

[10] Orenstein, J.A., Merret, T.H.: A class of data structures for associate searching. In: Proceedings of the 3rd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, PODS 1984, Waterloo, Canada, pp. 294–305 (1984)

[11] Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. 30(1), 41–82 (2005)

[12] Random dataset generator for SKYLINE operator evaluation, `http://randdataset.projects.postgresql.org/`

[13] Sacharidis, D., Bouros, P., Sellis, T.K.: Caching dynamic skyline queries. In: Ludäscher, B., Mamoulis, N. (eds.) SSDBM 2008. LNCS, vol. 5069, pp. 455–472. Springer, Heidelberg (2008)

[14] Sharifzadeh, M., Shahabi, C.: The spatial skyline queries. In: Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB 2006, Seoul, Korea, pp. 751–762 (2006)

[15] Su, H.Z., Wang, E.T., Chen, A.L.P.: Continuous Probabilistic Skyline Queries over Uncertain Data Streams. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6261, pp. 105–121. Springer, Heidelberg (2010)

[16] Tan, K.L., Eng, P.K., Ooi, B.C.: Efficient progressive skyline computation. In: Proceedings of the 27th International Conference on Very Large Data Bases, VLDB 2001, Roma, Italy, pp. 301–310 (2001)

[17] Zhang, S., Mamoulis, N., Cheung, D.W.: Scalable skyline computation using object-based space partitioning. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, pp. 483–494 (2009)

EcoTop: An Economic Model for Dynamic Processing of Top-k Queries in Mobile-P2P Networks

Nilesh Padhariya[1], Anirban Mondal[1], Vikram Goyal[1], Roshan Shankar[2], and Sanjay Kumar Madria[3]

[1] IIIT, Delhi, India
{nileshp,anirban,vikram}@iiitd.ac.in
[2] NSIT, Delhi, India
roshan@nsitonline.in
[3] Missouri University of Science and Technology, Rolla, USA
madrias@mst.edu

Abstract. This work addresses the processing of top-k queries in mobile ad hoc peer to peer (M-P2P) networks using economic schemes. Our proposed economic model, designated as EcoTop, issues economic rewards to the mobile peers, which send *relevant* data items (i.e., those that contribute to the top-k query result), and penalizes peers for sending *irrelevant* items, thereby incentivizing the optimization of communication traffic. The main contributions of our work are three-fold. First, we propose the EcoTop economic model for efficient top-k query processing in M-P2P networks. Second, we propose two schemes, namely ETK and ETK+, for assigning rewards/penalties to peers and for enabling peers to re-evaluate the scores of their data items for item re-ranking purposes. Third, we conduct a performance study, which demonstrates that EcoTop is indeed effective in improving the performance of top-k queries, while minimizing the communication traffic. Notably, our novel economic incentive model also discourages free-riding in M-P2P networks.

1 Introduction

In a Mobile ad hoc Peer-to-Peer (M-P2P) network, mobile peers (MPs) interact with each other in a peer-to-peer (P2P) fashion. Proliferation of mobile devices (e.g., laptops, PDAs, mobile phones) coupled with the ever-increasing popularity of the P2P paradigm strongly motivate M-P2P network applications.

Suppose an MP wants to find the top-k restaurants with "happy hours" (or "manager's special hours") within 1 km of her current location. A broker can facilitate the MP by soliciting information (in terms of both distance and time) from peers in its vicinity, and it can then integrate this information with its global ranking list of restaurants (e.g., using Michelin's restaurant guide). In a parking lot application, MPs can collect information about available parking slots and charges, and then they can inform the brokers. Parking lot availability information has to be current and the broker can integrate this information with its (global) list of parking slots. The broker can then provide the top-k available slots to the MP in terms of charges and distance (from the MP's current location). Similarly, an MP may want to find the top-k stores selling Levis jeans in a shopping mall with criteria such as (low) price.

Such *spatio-temporal* queries cannot be directly answered by the broker without obtaining information from other MPs. Such M-P2P interactions are generally not freely supported by existing wireless communication infrastructures. The inherently ephemeral nature of M-P2P environments suggests that *timeliness* of data delivery is of paramount importance in these applications, thereby necessitating query deadlines. For example, an MP looking for top-k restaurants in her vicinity would generally prefer query results within a specified deadline.

Existing economic models for distributed systems [8] and static P2P networks [3,7] do not address top-k queries and M-P2P issues such as frequent network partitioning and mobile resource constraints. Incentive schemes for mobile environments [2,12,14] do not address top-k queries. Top-k querying approaches [4,5,6,9,10,11,13] do not consider economic schemes and M-P2P architecture.

Data availability in M-P2P networks is typically lower than in fixed networks due to frequent network partitioning arising from peer movement and/or peers autonomously switching 'off' their mobile devices. Data availability is further exacerbated due to rampant free-riding [3,7], which is characteristic of P2P environments. Furthermore, MPs generally have limited resources (e.g., bandwidth, energy, memory space). Since sending/receiving messages expend the limited energy of MPs, minimizing communication traffic becomes a necessity to address peers' energy constraints. Thus, economic incentive schemes become a necessity to entice resource-constrained MPs to improve data availability.

This work proposes an economic model, designated as EcoTop, which addresses the efficient processing of top-k queries in M-P2P networks. In EcoTop, brokers facilitate top-k query processing in lieu of a *commission.* EcoTop requires a query-issuing MP to pay a *price* (in *virtual currency*), which is application-dependent, for obtaining its top-k query result. This price is used for making payments to *rankers* (i.e., MPs that sent items to answer the query), brokers and relay peers in order to incentivize them in answering the top-k query. Thus, an MP has to earn adequate currency by providing *service* (as a broker, ranker or relay peer) before it can issue its own queries, thereby discouraging free-riding.

EcoTop issues economic rewards to the rankers, which send *relevant* items (i.e., those which contribute to the top-k results) and penalizes them for sending *irrelevant* items. This incentivizes MPs to send only those items (to the broker), which have a higher probability of being in the top-k results, while making them careful about not sending irrelevant items. This optimizes communication traffic. MPs use the rewards/penalties as a feedback to re-evaluate their items' scores.

The main contributions of our work are three-fold:

1. We propose the EcoTop economic model for efficient top-k query processing in M-P2P networks.
2. We propose two schemes, namely ETK and ETK+, for assigning rewards/ penalties to MPs and for enabling them to re-evaluate the scores of their data items for item re-ranking purposes.

3. We conduct a performance study, which demonstrates that EcoTop is indeed effective in improving the performance of top-k query processing, while minimizing the communication traffic.

ETK and ETK+ differ in that ETK performs equal distribution of rewards/ penalties across the relevant items, while ETK+ uses a weighted distribution. Both schemes also discourage free-riding due to their economic nature.

2 Related Work

Economic models have been discussed in [8] primarily for resource allocation in distributed systems. However, these works do not address M-P2P issues such as frequent network partitioning and mobile resource constraints. Incentive schemes for static P2P networks [3,7] are too static to be deployed in M-P2P networks as they assume peers' availability and fixed topology.

Incentive schemes for mobile ad-hoc networks (MANETs) [2] encourage MPs in forwarding messages. They do not consider top-k query processing, rewards/ penalties to rankers and M-P2P architecture. The proposal in [4] discusses a message processing method for top-k queries in MANETs with the objective of reducing communication traffic, while maintaining the accuracy of the query result. However, it does not consider economic schemes. Incentive schemes for opportunistically disseminating spatio-temporal data in M-P2P networks have been discussed in [12,14]. In contrast, our approach disseminates data on-demand to optimize peers' energy consumption. Furthermore, the proposals in [12,14] do not address top-k queries and rewards/penalties to rankers.

Top-k query processing approaches focus on semantic-based probabilistic formulations [11], cost-based optimization for middleware access [5] and location-based methods using R-tree variants [6]. The work in [9] presents a top-k structured mobile Web search engine. Top-k query processing approaches have also been proposed for wireless sensor networks [10,13]. However, these works do not consider economic schemes and M-P2P architecture. The tutorial in [15] details top-k query processing in wireless sensor networks.

3 EcoTop: An Economic Model for Dynamic Processing of Top-k Queries in M-P2P Networks

This section discusses our proposed EcoTop economic model for top-k query processing in M-P2P networks.

Architecture of EcoTop

EcoTop's architecture comprises MPs that can assume one of the four following roles: *query-issuer*, *broker*, *ranker* and *relay*. These roles are interchangeable e.g., an MP can be a broker for a given query, but a ranker for another query.

Query-issuer QI issues queries of the form (k, L, τ_Q, ρ), where k is the number of data items that are requested in the top-k query. L represents the query location, and is of the form of $\{(x, y),\ rad\}$. Here, (x, y) represents the spatial coordinates associated with a given query Q, while rad represents the radius. For example QI may want to find restaurants within 1 km of its current location L. τ_Q is the deadline time of Q. Notably, the ephemerality of M-P2P environments necessitates timely responses, and consequently query deadlines. ρ is the query price that QI will pay to obtain the top-k query result[1]. *Broker* B acts as a mediator, which facilitates efficient top-k query processing in lieu of a commission. As we shall see in Section 4, B also performs economic incentive functions i.e., distribution of rewards/penalties. For the sake of convenience, we shall henceforth use the term **payoffs** to refer to rewards/penalties.

Rankers are MPs, which provide data items for answering the top-k query. Rankers are rewarded if their items contribute to the top-k result, otherwise they are penalized. Relay MPs forward messages in multi-hop M-P2P networks in lieu of a small constant commission. Notably, payments to rankers are typically higher than that of broker commissions in order to better incentivize MPs to provide data. This is because MPs providing data generally contribute significantly more to data availability than brokerage functions. Furthermore, relay commission is lower than that of broker commission to better incentivize brokerage functions as compared to relay functions.

Query Processing in EcoTop

QI broadcasts a top-k query Q. Each *broker* replies to QI with information about its remaining energy En, bid price ρ_{bid}, current currency $Curr$ and distance $Dist$ from QI. Then QI computes a score η for each broker, and selects the broker with the highest value of η for processing Q. η is computed as follows:

$$\eta = (w_1 \times En) + (w_2 / \rho_{bid}) + (w_3 / Curr) + (w_4 / Dist) \tag{1}$$

where w_1 to w_4 are weight coefficients such that $0 < w_1, w_2, w_3, w_4 \leqslant 1$ and $\sum_{i=1}^{4} w_i = 1$. Thus, EcoTop prefers relatively high-energy brokers because they are less likely to run out of energy, while processing the query. Lower values of bid prices are preferred by QI since it wants to obtain the query result at lower cost. Brokers with less currency are given higher preference to facilitate revenue-balancing across brokers. This prevents low-currency brokers from starvation, which may result in decreased number of brokers in the network. Moreover, QI prefers relatively nearby brokers to obtain the query result in a timely manner.

Now, the broker broadcasts Q over the network with time-to-live (TTL) of n hops[2]. Each ranker R has an *individual* item ranking list T_{fR}, each data item of which is associated with an item rank r and a selection probability μ. Notably, the value of r is subjective because it is autonomously assigned to an item by a given ranker. The implication is that the same item may be ranked differently at different rankers. As we shall see in Section 4, μ facilitates the adjustment

[1] Query results received by QI after the deadline time of τ_Q entail no payments.

[2] Results of our preliminary experiments showed that $n = 6$ is a reasonable value for our application scenarios.

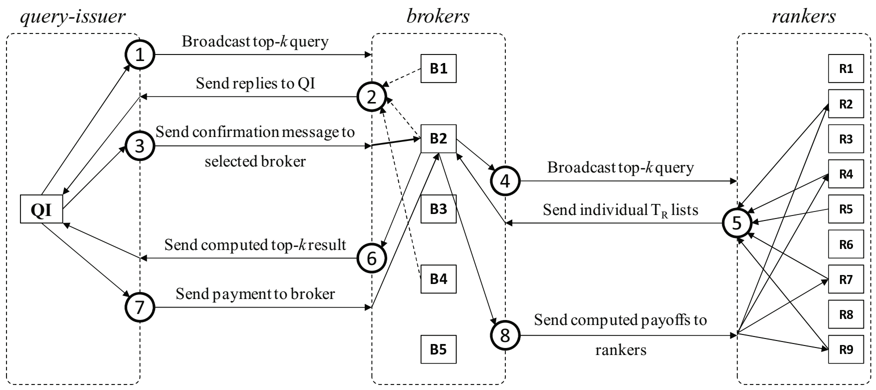


Fig. 1. Illustrative example of query processing in EcoTop

of item selection probability based on recent payoffs assigned to a given item. Using the values of μ and r, each ranker R computes a score γ and selects items with relatively higher values of γ to send to the broker. γ is computed below:

$$\forall i \in T_{fR}: \quad \gamma_i \; = \; (\; w_1 \times (N_{T_{fR}} - r_i)/N_{T_{fR}} \;) \; + \; (\; w_2 \times \mu_i \;) \qquad (2)$$

where r_i and μ_i are the rank and the selection probability of item i respectively. $N_{T_{fR}}$ is the total number of items in T_{fR}. Here, w_1 and w_2 are weight coefficients such that $0 < w_1, w_2 \leqslant 1$ and $w_1 + w_2 = 1$. EcoTop stipulates that $w_2 > w_1$ to give higher weightage to the item selection probability than to the rank of the item. As we shall see in Section 4, this is consistent with the overall objective of our incentive model i.e., linking item re-ranking with payoffs. We set $w_1 = 0.2$ and $w_2 = 0.8$ for all the MPs. Each ranker is associated with a risk profile δ, where $0 < \delta \leqslant 1$. Only items, whose respective values of γ exceed δ, are consolidated by the ranker in a list T_R and sent to the broker. As the value of δ increases, ranker's risk of getting penalized decreases.

Each broker has a global ranking list T_G, which can differ across brokers. (In our top-k restaurant application scenario, the global ranking list could be any standard restaurant guide such as the Michelin's guide.) Upon receiving T_R lists from possibly multiple rankers, the broker B collates and compares them with T_G. B parses T_G in a top-down fashion as follows. If an item i in T_G occurs in at least one of the individual T_R lists, it is added to the top-k result set T_A along with the unique identifiers of the rankers that sent i. (If i does not occur in any of the individual T_R lists, B simply traverses downwards to the next item in T_G.) B continues parsing T_G until the result set T_A contains k items. Then B sends T_A to QI[3]. Observe how the global ranking list is used as a benchmark to evaluate the top-k relevance of the items sent by autonomous rankers, whose individual item ranking lists may differ.

[3] If T_A contains less than k items, the result set is deemed to be *incomplete*, and it is not sent to QI.

Upon receiving T_A, QI pays B, which deducts its own commission before distributing the payoffs to rankers and commissions to relay MPs. Then each ranker R re-evaluates the selection probability μ of each item in its own T_R based on received payoffs, and then re-computes the values of γ for these items.

Figure 1 depicts an illustrative example of query processing in EcoTop. Figures 2a, 2b and 2c present the algorithms executed by query-issuers, brokers and rankers respectively.

```
Algorithm EcoTop_Query_Issuer (QI)
Input: $Q$: top-$k$ query
(1) Broadcast $Q$
(2) Wait for replies from brokers
(3) Receive replies from brokers
     /* Each reply contains broker's energy level, bid price, currency and distance from $QI$*/
(4) Consolidate the replies into a list $L_B$
(5) if $L_B$ is not empty
(6)    Compute the value of $\eta$ for each broker in $L_B$
(7)    Select the broker with the highest value of $\eta$
(8)    Send confirmation message to the selected broker $Sel_B$
(9)    Wait for the query result from $Sel_B$
(10)   Receive the top-$k$ result $T_A$ from $Sel_B$
(11)   if $T_A$ is received within the deadline time $\tau_Q$
(12)      Send the payment $\rho$ to $Sel_B$
end
```

(a) Algorithm for *Query-issuer*

```
Algorithm EcoTop_Broker (B)
Input: $Q$: top-$k$ query
(1) Receive $Q$ from query-issuer $QI$
(2) Send bid to $QI$ with details of energy level,
     bid price, currency and distance
(3) Wait for reply from $QI$
(4) if Selected by $QI$
(5)    Broadcast $Q$ to obtain top-$k$ individual
       rank lists $T_R$ from rankers
(6)    Wait $\epsilon$ time units for replies from rankers
(7)    Receive replies (within $\epsilon$ time units)
       from interested rankers
       /* Each reply contains top-$k$ individual
       rank list $T_R$ */
(8)    Consolidate received $T_R$ lists into a set $L_{T_R}$
(9)    if $L_{T_R}$ is not empty
(10)      Compute top-$k$ result set $T_A$ from $L_{T_R}$
(11)      if $|T_A| < k$
(12)         Do not send $T_A$ to $QI$
(13)      else
(14)         Send $T_A$ to $QI$
(15)         Receive payment $\rho$ from $QI$
(16)         Compute and send payoffs
             to the rankers
end
```

(b) Algorithm for *Broker*

```
Algorithm EcoTop_Ranker (R)
Input: $Q$: top-$k$ query
(1) Receive $Q$ from broker $B$
(2) for each item $i$ in rank list $T_{fR}$
(3)       Compute $\gamma_i$ for $i$
(4)       if $\gamma_i > \delta$
          /* $\delta$ quantifies ranker's risk profile */
(5)          Add $i$ to a list $T_R$
(6)          Sort the items in $T_R$
             in decending order of $\gamma$
(7) if $|T_R| < k$
(8)    Do not send $T_R$ to $B$
(9) else
(10)   Delete the bottom $|T_R|-k$ items from $T_R$
(11)   Send $T_R$ to $B$
(12)   Wait for payoffs from $B$
(13)   Receive payoffs from $B$
(14)   for each item $i$ in $T_R$
(15)      if $i$ is rewarded
(16)         Increase $\mu$ of $i$
(17)      else
(18)         Decrease $\mu$ of $i$
(19)      Recompute $\gamma_i$
end
```

(c) Algorithm for *Ranker*

Fig. 2. Algorithms in EcoTop

4 Economic Schemes in EcoTop

This section discusses two economic schemes used by EcoTop for assigning payoffs to rankers to facilitate them in re-evaluating their item scores. We designate these schemes as ETK (i.e., EcoTopK) and ETK+ respectively.

We define an item i to be *relevant* to a query Q if it occurs in the top-k query result set T_A. We define a *successful* ranker w.r.t. data item i if i is relevant to the query Q, otherwise the ranker is deemed to be *unsuccessful*[4].

ETK and ETK+ differ in that ETK performs equal distribution of rewards to the relevant items, while ETK+ uses a weighted distribution. Furthermore, both schemes also differ in their way of assigning penalties to unsuccessful rankers i.e., ETK and ETK+ assign equal and weighted penalties respectively. Incidentally, both schemes also discourage free-riding due to their economic nature.

In both ETK and ETK+, the total payment ρ_R to be distributed to the successful rankers is computed as follows:

$$\rho_R = \rho - \rho_B - \rho_{RL} \tag{3}$$

where ρ is the query price paid by QI to the broker, ρ_B is the broker commission and ρ_{RL} is the total amount of relay commission that the broker will pay to the relay MPs in the respective successful query paths. Notably, the value of ρ_B is application-dependent. For both ETK and ETK+, we defined ρ_B as 10% of the query price ρ. Although our schemes can be intuitively generalized to work with other values of ρ_B, results of our preliminary experiments showed that our schemes perform best when ρ_B is in the range of 5% to 15% of ρ. Observe that this is also consistent with our overall objective of providing better incentives to rankers than to brokers.

As we shall see shortly, the rewards to be assigned to the successful rankers are computed based on the value of ρ_R. Similarly, the penalties to be assigned to the unsuccessful rankers are also computed based on the value of ρ_R. The broker receives the penalty payments from the unsuccessful rankers, and sends the total amount of penalty payments back to QI. Thus, it is possible for the effective payment made by QI to the broker to be less than ρ.

ETK

In ETK, ρ_R is equally divided among all the relevant items. Then each ranker, which successfully sent item i, receives a reward P_i that is equal to the total reward for item i divided by the total number f_i of successful rankers w.r.t. item i. Given that the top-k result set is T_A, P_i is computed as follows:

$$\forall i \in T_A : \quad P_i = \frac{1}{f_i}\left(\frac{\rho_R}{k}\right) \tag{4}$$

The reward REW_{Rj} assigned to a given ranker Rj is the total amount that it obtains for each of its relevant items i.e., those that occur in the $T_A \cap T_{Rj}$,

[4] A ranker may be successful w.r.t. item i, but unsuccessful w.r.t. another item j.

where T_{Rj} is the individual rank list of Rj. Given the set of rankers S_{Ranker}, the computation of REW_{Rj} follows:

$$\forall j \in S_{Ranker}: \quad REW_{Rj} = \sum_{i \in (T_A \cap T_{Rj})} P_i \tag{5}$$

ETK defines penalties based on the notion of *opportunity cost.* This is because for all *relevant* items, which were not sent by ranker Rj, Rj would have earned currency if it had sent those items. Hence, the penalty PEN_{Rj} assigned to Rj equals $\sum P_i$, where i represents items that occur in $T_A - T_{Rj}$. The computation of PEN_{Rj} follows:

$$\forall j \in S_{Ranker}: \quad PEN_{Rj} = \psi \times \left[\sum_{i \in (T_A - T_{Rj})} P_i \right] \tag{6}$$

where ψ is the factor that represents the trade-off between communication overhead and peer participation. If the value of ψ is high, communication overhead would reduce because peers would be wary of sending data to the broker due to the higher penalties assigned to unsuccessful rankers. However, this would also reduce peer participation in the M-P2P network. On the other hand, if the value of ψ is low, peer participation would increase albeit at the cost of increase communication overhead due to lower disincentives for sending items that do not contribute to the top-k result. In this work, we set the value of ψ to 1.3, which implies that the penalties for sending irrelevant items is 30% more than the reward for sending relevant items. This creates disincentives for sending out irrelevant items, while keeping the peer participation at a reasonable level. We leave the determination of an optimal value for ψ to future work.

The net payment NET_{Rj} received by Rj is the difference between its total reward and its total penalty. Hence, NET_{Rj} equals $REW_{Rj} - PEN_{Rj}$.

Now, based on the payoffs received, Rj will re-evaluate the selection probability of all the items in its individual T_{Rj}. ETK performs rank-weighted increase/decrease in μ for each item, depending on whether the item is rewarded or penalized. For each item i in T_{Rj}, the value of μ_{ij} is computed as follows:

$\forall j \in S_{Ranker}, \forall i \in T_{Rj}$:

$$\mu_{ij} = \begin{cases} min(\ \mu_{ij} + \alpha_{up} \left(\frac{|T_{Rj}| - r_{ij}}{|T_{Rj}|} \right),\ 1\ ) & \text{if } i \text{ is rewarded} \\ max(\ \mu_{ij} - \alpha_{down} \left(\frac{|T_{Rj}| - r_{ij}}{|T_{Rj}|} \right),\ 0\ ) & \text{if } i \text{ is penalized} \end{cases} \tag{7}$$

where r_{ij} is the rank of item i in T_{Rj}. Observe that, μ_{ij} increases slightly for higher-rank items that received rewards but decreases significantly in case of penalty. Similarly, μ_{ij} increases significantly for lower-rank items that received rewards but decreases relatively slightly in case of penalty. Here, α_{up} and α_{down} represent the weight coefficients for assigning rewards and penalties respectively. ETK stipulates that $0 < \alpha_{up}, \alpha_{down} \leqslant 1$ and $\alpha_{up} < \alpha_{down}$ to ensure that penalties exceed rewards, thereby creating these incentives for rankers in terms of sending out items that are not relevant. In this work, we set the values of α_{up} and α_{down} to 0.1 and 0.3 respectively. We leave the determination of optimal values of α_{up} and α_{down} to future work.

ETK+

In ETK+, ρ_R is divided among all the items in the top-k result T_A based on their respective rank-weights i.e., each item i with its associated rank r_i has weight $w_i = (k - r_i)$, where highest to lowest rank counts are from 0 to $(k-1)$. Furthermore, total number W of weights of all items in T_A is computed as $W = \sum_{i=1}^{k} w_i = k\ (k+1)/2$. Similar to ETK, each ranker, which successfully sent item i, receives a reward P_i that is equal to the total reward for item i divided by the total number f_i of successful rankers w.r.t. item i. Thus, in ETK+, P_i is computed as follows:

$$\forall i \in T_A : \quad P_i = \frac{1}{f_i}\left(\frac{w_i}{W} \times \rho_R\right) \tag{8}$$

Consequently, rewards and penalties assigned to each ranker Rj are computed as in Equations 5 and 6 respectively, using the value of P_i from Equation 8. Hence, R_j's net payment is the difference between its received rewards and penalties.

Now, each ranker Rj will re-evaluate the score (effectively the selection probability μ) of each item i in its top-k rank list T_{Rj} on the basis of its received payoffs. The effective change in the selection probability of an item depends upon two factors: (a) the notion of item selection potential w.r.t. the risk profile (δ) (b) earning potential of the ranker Rj. Item selection potential increases as the *difference* between μ and δ increases. Average selection potential for rewarded and penalized items for each ranker Rj are computed as s_j and s'_j respectively. The computations of s_j and s'_j are shown below:

$$\forall j \in S_{Ranker} : \ s_j = \frac{1}{|T_{Rj} \cap T_A|}\left[\sum_{i \in (T_{Rj} \cap T_A)} (\mu_{ij} - \delta_j)\right] \tag{9}$$

$$\forall j \in S_{Ranker} : \ s'_j = \frac{1}{|T_{Rj} - T_A|}\left[\sum_{i \in (T_{Rj} - T_A)} (\mu_{ij} - \delta_j)\right] \tag{10}$$

where T_{Rj} is the top-k rank list of Rj, T_A is the top-k result of a query Q, μ_{ij} is the selection probability of item i in T_{Rj} and δ_j is the risk profile of Rj.

Earning potential e_j of each ranker Rj is a measure of its selection efficiency. $e_j = |\ (REW_{Rj} - PEN_{Rj})/(REW_{Rj} + PEN_{Rj})\ |$. Based on the payoff of each item i in T_{Rj}, increase/decrease in μ_{ij} is computed as follows:

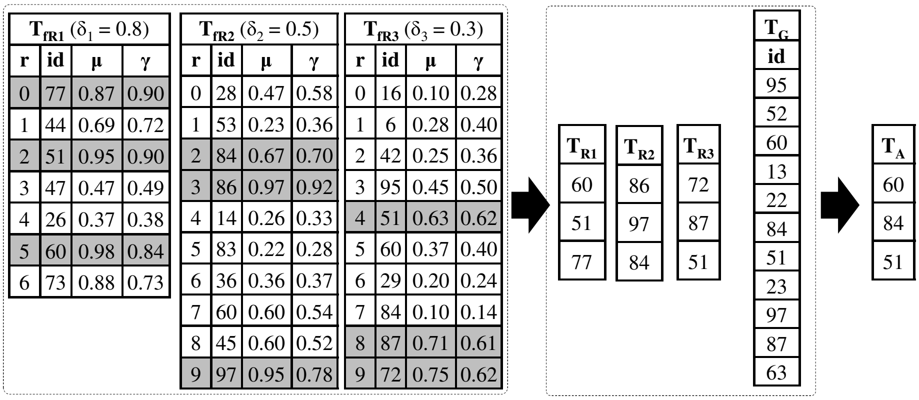
$\forall j \in S_{Ranker}, \forall i \in T_{Rj}$:

$$\mu_{ij} = \begin{cases} min(\ \mu_{ij} + \alpha_{up}\left(\frac{s_j + e_j}{2}\right)\ ,\ 1\ ) & \text{if } i \text{ is rewarded} \\ max(\ \mu_{ij} - \alpha_{down}\left(\frac{s'_j + e_j}{2}\right)\ ,\ 0\ ) & \text{if } i \text{ is penalized} \end{cases} \tag{11}$$

where α_{up} and α_{down} are the weight coefficients discussed in Equation 7.

Illustrative Example for ETK and ETK+

Figure 3 depicts an illustrative example of the computations in ETK and ETK+. In Figure 3a, observe how each ranker R computes the value of γ using Equation 2 with $w_1 = 0.2$ and $w_2 = 0.8$. For ranker $R1$, the selected elements in T_{R1} are shaded i.e., T_{R1}={77,51,60} because their respective values of γ exceed 0.8 ($\delta_1 = 0.8$). Figure 3b depicts the payoff computations with $\psi = 1.3$. Observe that ETK+ assigns high penalties to rankers for sending irrelevant items e.g. ETK+ assigned 97.50 to $R3$ as compared to 78.00 in ETK. Figure 3c depicts the re-evaluation of the selection probability μ with $\alpha_{up} = 0.1$ and $\alpha_{down} = 0.3$.



T_{fR1} (δ_1 = 0.8)			
r	id	μ	γ
0	77	0.87	0.90
1	44	0.69	0.72
2	51	0.95	0.90
3	47	0.47	0.49
4	26	0.37	0.38
5	60	0.98	0.84
6	73	0.88	0.73

T_{fR2} (δ_2 = 0.5)			
r	id	μ	γ
0	28	0.47	0.58
1	53	0.23	0.36
2	84	0.67	0.70
3	86	0.97	0.92
4	14	0.26	0.33
5	83	0.22	0.28
6	36	0.36	0.37
7	60	0.60	0.54
8	45	0.60	0.52
9	97	0.95	0.78

T_{fR3} (δ_3 = 0.3)			
r	id	μ	γ
0	16	0.10	0.28
1	6	0.28	0.40
2	42	0.25	0.36
3	95	0.45	0.50
4	51	0.63	0.62
5	60	0.37	0.40
6	29	0.20	0.24
7	84	0.10	0.14
8	87	0.71	0.61
9	72	0.75	0.62

T_{R1}	T_{R2}	T_{R3}
60	86	72
51	97	87
77	84	51

T_G
id
95
52
60
13
22
84
51
23
97
87
63

T_A
60
84
51

(a) Compilation of top-k result T_A ($k = 3$)

	P_j (ETK)				P_j (ETK+)			
T_A	ρ_R/k	R1	R2	R3	ρ_R/k	R1	R2	R3
60	30	30	-	-	45	45	-	-
84	30	-	30	-	30	-	30	-
51	30	15	-	15	15	7.5	-	7.5
	REW_{Ri}	45	30	15	REW_{Ri}	52.5	30	7.5

Ranker	T_A-T_{Ri}	PEN_{Ri} (ETK)	PEN_{Ri} (ETK+)
R1	{84}	1.3 x 30 = 39.00	1.3 x 30 = 39.00
R2	{60, 51}	1.3 x (30+15) = 58.50	1.3 x (45+7.5) = 68.25
R3	{60, 84}	1.3 x (30+30) = 78.00	1.3 x (45+30) = 97.50

(b) Computation of rewards and penalties

R1 (δ_1 = 0.8)			
id	μ	μ_{ETK}	μ_{ETK+}
77	0.87	0.57	0.82
44	0.69	0.69	0.69
51	0.95	1.00	0.96
47	0.47	0.47	0.47
26	0.37	0.37	0.37
60	0.98	1.00	0.99
73	0.88	0.88	0.88

R2 (δ_2 = 0.5)			
id	μ	μ_{ETK}	μ_{ETK+}
28	0.47	0.47	0.47
53	0.23	0.23	0.23
84	0.67	0.75	0.71
86	0.97	0.76	0.89
14	0.26	0.26	0.26
83	0.22	0.22	0.22
36	0.36	0.36	0.36
60	0.60	0.60	0.60
45	0.60	0.60	0.60
97	0.95	0.92	0.87

R3 (δ_3 = 0.3)			
id	μ	μ_{ETK}	μ_{ETK+}
16	0.10	0.10	0.10
6	0.28	0.28	0.28
42	0.25	0.25	0.25
95	0.45	0.45	0.45
51	0.63	0.69	0.69
60	0.37	0.37	0.37
29	0.20	0.20	0.20
84	0.10	0.10	0.10
87	0.71	0.65	0.53
72	0.75	0.72	0.57

(c) Updates in the selection probabilities

Fig. 3. Illustrative example of EcoTop schemes

5 Performance Evaluation

This section reports our performance evaluation by means of simulation in OMNeT++ (www.omnetpp.org). Our experiments consider a region with area of

Table 1. Parameters of our performance study

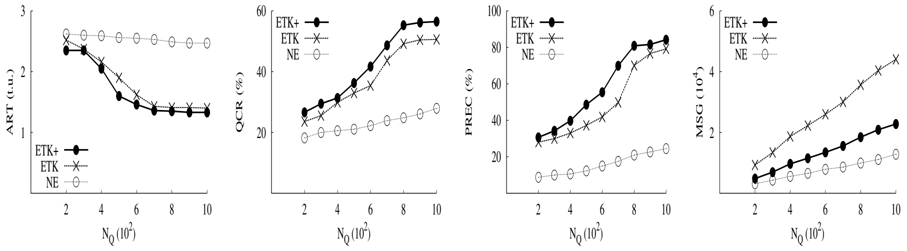
Parameter	Default Value	Variations
k	8	4, 12, 16, 20, 24
Number of MPs (N_{MP})	100	20, 40, 60, 80
Percentage of brokers	15%	
Queries/time unit	10	
Probability of MP availability	50% to 85%	
Initial energy of an MP	90000 to 100000 energy units	
Memory space of each MP	8 MB to 10 MB	
Speed of an MP	1 meters/s to 10 meters/s	
Size of a data item	50 Kb to 350Kb	

1000 metres × 1000 metres. MPs in the region move according to the Random Waypoint Model [1]. The Random Waypoint Model is appropriate for our application scenarios, which consider random movement of users. As a single instance, people looking for top-k restaurants generally move randomly i.e., they do not follow any specific mobility pattern. Communication range of all MPs is a circle of 50 metres radius. Each MP contains 20 to 25 data items.

For each top-k query Q, the query-issuer is selected randomly from among all MPs. 10 such top-k queries are issued in the network per time unit and we set the query deadline time τ_Q to 5 time units. Price ρ for each top-k query is chosen randomly between 100 to 500 currency units. Broker commission ρ_B is set to 10% of the total query price ρ. Relay commission ρ_{RL} is set to 1 currency unit. Initial energy of an MP is selected to be randomly in the range of 90000 to 100000 energy units. Sending and receiving a message require 1.5 and 1 energy units respectively. Incidentally, querying proceeds via the AODV protocol.

Recall that each ranker is associated with a risk profile δ. The number of MPs with the values of δ as 0.3 (high-risk), 0.5 (medium-risk) and 0.8 (low-risk) are 27, 43 and 30 respectively. For all our experiments, the economic parameters for ETK and ETK+ are set as follows: (a) weight coefficients w_1 and w_2 in γ (see Equation 2) are set to 0.2 and 0.8 respectively (b) penalty factor ψ (see Equation 6) is set to 1.3 (c) α_{up} and α_{down} (see Equations 7 and 11) are set to 0.1 and 0.3 respectively. Table 1 summarizes the parameters used in our performance evaluation. Notably, parameter values depend on initial experiments.

Performance metrics are average response time (ART), precision rate (PREC), query completeness rate (QCR) and communication traffic (MSG). We define a query as **completed** if the broker receives at least k items from the rankers within 70% of the query deadline time τ_Q. Notably, a broker may fail to receive at least k items due to reasons such as ranker unavailability and network partitioning. We compute ART only for the *completed* queries. $ART = \frac{1}{N_C} \sum_{q=1}^{N_C} (t_f - t_0)$, where t_0 is the query-issuing time, t_f is the time of the query result reaching the query-issuer, and N_C is the total number of *completed* queries. We compute ART in simulation time units (t.u.).



(a) ART (b) QCR (c) PREC (d) MSG

Fig. 4. Performance of ETK & ETK+

PREC is the average precision rate over all the queries. Suppose T_{Aq} is the top-k query result and T_{Gq} is the global top-k rank list of the respective broker for a query q. To obtain PREC for q, we measure the number of items in T_{Aq} which also occur in T_{Gq}; then we divide by the number of items in T_{Gq}. Notably, PREC is computed only for *completed* queries. Thus, $PREC = \left[\frac{1}{N_C} \sum_{q=1}^{N_C} \left(\frac{|T_{Gq} - T_{Aq}|}{|T_{Gq}|} \right) \times 100\right]$. QCR is the ratio of total number N_C of completed queries to the total number N_Q of queries. $QCR = (\ N_C/N_Q\ ) \times 100$. We define MSG as the total number of messages incurred for query processing during the course of the experiment. Thus, $MSG = \sum_{q=1}^{N_Q} M_q$, where M_q is the number of messages incurred for the q^{th} query.

As reference, we adapt a scheme **NE** (**Non_Economic**), which does not provide any economic incentives to MPs towards top-k query processing. Similar to ETK and ETK+, brokers facilitate top-k query processing in NE. In NE, query processing proceeds in exactly the same manner as in the case of ETK and ETK+. The only difference is that the broker sends feedback to the rankers without including any economic payoff. Thus, the broker's feedback only concerns which of the rankers-sent items contributed (or did not contribute) to the top-k result. Hence, each ranker re-evaluates the selection probability as follows. If a sent item contributed to the top-k result, it will increase the selection probability of the item by m such that $0 < m \leqslant 1$. Conversely, if an item did not contribute to the top-k result, it will decrease the selection probability of the item by m. For our experiments, we set m to 0.005. Observe that item re-ranking in NE is not linked to any economic payoff.

Performance of ETK and ETK+

We conducted an experiment using the default values of the parameters in Table 1. Figure 4 depicts the results. After the first 300 queries have been processed, ART decreases sharply for both ETK and ETK+, and QCR and PREC increase steadily due to incentives and effective item re-ranking. However, ART, QCR and PREC eventually plateau due to network partitioning and unavailability of some of the rankers. Notably, ETK+ performs better than ETK in terms of ART, QCR and PREC due to two reasons. First, ETK+'s rank-weighted payoff strategy provides better incentivization than the uniform incentivization provided by ETK. Second, ETK+ provides more effective re-evaluation of μ due to its payoff-driven score update strategy as opposed to ETK's rank-weighted score update strategy, in which the item re-rankings are not directly tied to the payoffs associated with the rankers' items.

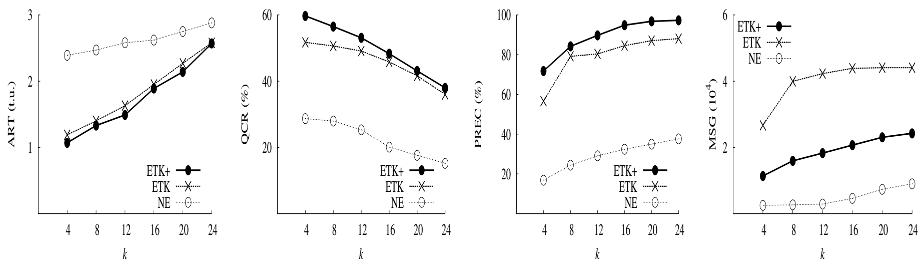
NE exhibits relatively constant ART due to its non-incentive nature, hence most queries are answered relatively close to the deadline time. QCR and PREC remain relatively low for NE because it does not provide any economic incentives for ranker participation. Hence, the broker does not always receive at least k items from rankers, thereby resulting in a significant number of incomplete queries. The increase in QCR and PREC occurs due to the re-evaluation of μ,but this increase is only slight due to the relatively less effective item re-ranking strategy, which is neither directly nor indirectly tied to any economic payoff.

Recall that MSG is the cumulative measure of the messages over the course of the experiment. Hence, MSG increases over time for all the three approaches as more queries are being processed. NE incurs lower MSG than both ETK and ETK+ because fewer rankers reply to queries in the absence of incentives. ETK+ assigns higher amount of penalties (as compared to ETK) to rankers that send irrelevant items, hence fewer rankers reply to the broker in case of ETK+. Thus, ETK+ incurs lower MSG than ETK.

Effect of Variations in *k*

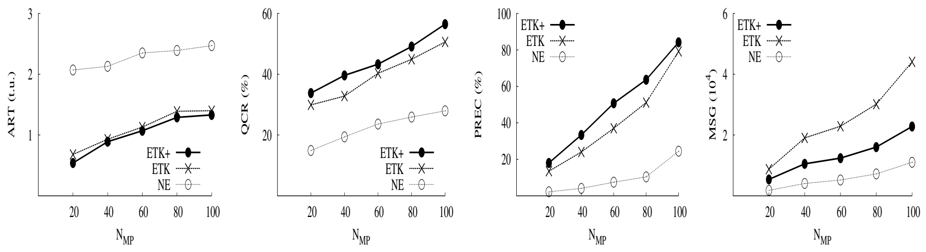
Figure 5 depicts the effect of variations in k. As k increases, ART increases and QCR decreases for all the approaches. This is because at higher values of k, fewer *nearby* rankers are able to provide enough relevant data items, thereby resulting in longer query paths. However, as k increases, PREC increases for all the approaches due to increasing probability of the items (sent by the rankers) in terms of contributing to the top-k result. For example, if $k = 4$, an item will contribute to the top-k *only* if it matches one of the four items in the broker's global top-k list T_G. However, if $k = 24$, T_G has 24 items, hence the ranker-sent item has a better chance of a 'match' with *any* one of the items in T_G.

Interestingly, even though *ranker participation* decreases with increase in k, *relay MP participation* increases because of the involvement of more MPs for retrieving a larger number of items. The (potential) rankers, which were not able to send data items, simply start forwarding queries, thereby increasing MSG for all the approaches. MSG eventually plateaus for ETK because its effective incentivization strategy *already involves* the participation of the majority of rankers at $k = 8$. Hence, relatively fewer additional rankers become involved in the query processing when $k > 8$. ETK incurs more number of messages than ETK+ and NE due to the reasons explained for the results in Figure 4d.



(a) ART (b) QCR (c) PREC (d) MSG

Fig. 5. Effect of variations in k



(a) ART (b) QCR (c) PREC (d) MSG

Fig. 6. Effect of variations in the number of MPs

Effect of Variations in the Number of MPs

We conducted an experiment to examine the scalability of ETK and ETK+. Figure 6 depicts the results. As the number N_{MP} of MPs increases, ART shows a generally increasing trend essentially due to larger network size. However, ART increases at a lower rate for ETK and ETK+ as compared to NE due to their effective economic incentive strategies, as discussed for the results in Figure 4a.

As N_{MP} increases, QCR and PREC increase for both ETK and ETK+ because larger network implies the presence of more rankers. NE exhibits a significantly lower rate of increase in QCR and PREC than both ETK and ETK+ essentially due to its non-economic approach, which does not effectively incentivize ranker participation. The explanation for the results in Figure 6d is essentially similar to that of the results in Figure 4d. Furthermore, MSG increases for all the approaches due to larger network size.

6 Conclusion

We have proposed EcoTop, which is an economic model for dynamic processing of top-k queries in M-P2P networks. EcoTop issues economic rewards to MPs, which send *relevant* data items, and penalizes MPs for sending *irrelevant* items, thereby incentivizing the optimization of communication traffic. EcoTop uses two schemes, namely ETK and ETK+, for assigning payoffs to peers and for enabling peers to re-evaluate the scores of their data items for item re-ranking purposes. ETK and ETK+ differ in that ETK performs equal distribution of payoffs to the relevant items, while ETK+ uses a weighted distribution. Our performance study shows that EcoTop is indeed effective in improving the performance of top-k queries, while minimizing the communication traffic. Our work has mainly considered empirical formulae based on important parameters affecting top-k queries. In the near future, we plan to extend EcoTop by incorporating game-theoretic techniques.

References

1. Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y.C., Jetcheva, J.: A performance comparison of multi-hop wireless ad hoc network routing protocol. In: Proc. MOBICOM (1998)
2. Buttyan, L., Hubaux, J.P.: Stimulating cooperation in self-organizing mobile ad hoc networks. ACM/Kluwer Mobile Networks and Applications 8(5) (2003)
3. Golle, P., Brown, K.L., Mironov, I.: Incentives for sharing in peer-to-peer networks. In: Proc. Electronic Commerce (2001)
4. Hagihara, R., Shinohara, M., Hara, T., Nishio, S.: A message processing method for top-k query for traffic reduction in ad hoc networks. In: Proc. MDM (2009)
5. Hwang, S., Chang, K.C.: Optimizing top-k queries for middleware access: A unified cost-based approach. ACM TODS 32(1) (2007)
6. Jung, H., Cho, B.K., Chung, Y.D., Liu, L.: On processing location based top-k queries in the wireless broadcasting system. In: Proc. ACM SAC (2010)
7. Kamvar, S., Schlosser, M., Garcia-Molina, H.: Incentives for combatting freeriding on P2P networks. In: Kosch, H., Böszörményi, L., Hellwagner, H. (eds.) Euro-Par 2003. LNCS, vol. 2790, pp. 1273–1279. Springer, Heidelberg (2003)
8. Kurose, J.F., Simha, R.: A microeconomic approach to optimal resource allocation in distributed computer systems. IEEE Trans. Computers 38(5) (1989)
9. Lee, W., Lee, J.J., Kim, Y., Leung, C.K.: Anchorwoman: Top-k structured mobile web search engine. In: Proc. CIKM (2009)
10. Liu, X., Xu, J., Lee, W.: A cross pruning framework for top-k data collection in wireless sensor networks. In: Proc. MDM (2010)
11. Soliman, M.A., Ilyas, I.F., Chang, K.C.: Probabilistic top-k and ranking-aggregate queries. ACM TODS 33(3) (2008)
12. Wolfson, O., Xu, B., Sistla, A.P.: An economic model for resource exchange in mobile Peer-to-Peer networks. In: Proc. SSDBM (2004)
13. Wu, M., Xu, J., Tang, X., Lee, W.: Monitoring top-k query in wireless sensor networks. In: Proc. ICDE (2006)
14. Xu, B., Wolfson, O., Rishe, N.: Benefit and pricing of spatio-temporal information in Mobile Peer-to-Peer networks. In: Proc. HICSS-39 (2006)
15. Zeinalipour-Yazti, D., Vagena, Z.: Distributed top-k query processing in wireless sensor networks. In: Proc. MDM (2008)

REQUEST: Region-Based Query Processing in Sensor Networks

Dong-Wan Choi and Chin-Wan Chung

Department of Computer Science,
Korea Advanced Institute of Science and Technology(KAIST)
335 Gwahangno, Yuseong-gu, Daejeon, Republic of Korea
dongwan@islab.kaist.ac.kr, chungcw@kaist.edu

Abstract. In wireless sensor networks, node failures occur frequently. The effects of these failures can be reduced by using aggregated values of small groups of sensors instead of the values of individual sensors. However, most existing works have focused on computing the aggregation of all the nodes without grouping. Only a few approaches proposed the processing of grouped aggregate queries. However, since groups in their approaches are disjoint, some relevant regions to the query can be missing. In this paper, we propose *REQUEST*, region-based query processing in sensor networks. A region in our approach is defined as a maximal set of nodes located within a circle having a diameter specified in the query. To efficiently construct a large number of regions covering the entire monitoring field, we build the *SEC* (Smallest Enclosing Circle) index. Moreover, in order to process a region-based query, we adapt a hierarchical aggregation method, in which there is a leader for each region. To reduce the communication cost, we formulate an optimal leader selection problem and transform the problem into the set-cover problem. Finally, we devise a query-initiated routing tree for the communication between the leader and non-leader nodes in a region. In the experimental results, we show that the result of our region-based query is more reliable than that of the query which is based on individual nodes, and our processing method is more energy-efficient than existing methods for processing grouped aggregate queries.

Keywords: Sensor networks, Regional query, Group-by aggregate query.

1 Introduction

Wireless sensor networks are widely used in various environmental monitoring applications. By using these applications, we can find some phenomena of the monitoring area, and detect some events corresponding to a given set of conditions. For example, if farmers could collect the information about the distribution of nutrients in the soil or locations colonized by many insects in real-time, the farmers could predict where and how much water, pesticide, and fertilizer are needed currently[1]. We can collect these information by gathering sensing values from the sensor nodes deployed in the monitoring area.

select node
from sensors
where t_l < temp < t_u
and h_l < humid < h_u

(a) Query based on individual nodes

select region
from sensors
group by region(10m)
having t_l < AVG(temp) < t_u
and h_l < AVG(humid)< h_u

(b) Region-based query

Fig. 1. Example queries

However, each sensing value can have some noises, as sensor nodes are prone to failure inherently. Moreover, managing a large number of individual sensor nodes is ineffective when only a macro view of the monitoring area is required.

To overcome these problems, we can construct small groups of sensor nodes, and use an aggregated value for each group. Previous works on grouping nodes [7,15] in the sensor networks address the problem by partitioning or clustering nodes with appropriate criteria such as the geographic location. In these works, there can be missing areas since they do not allow groups to overlap.

It is natural to group sensor nodes with regions of the same size. This is because, in the sensor network applications, we are not interested in a node itself, but a region in which the node is located.

Considering the above grouping method, we propose the region-based query processing method in sensor networks (hereafter called "*REQUEST*"). In REQUEST, the primitive processing unit is a region instead of a node. In addition, regions can overlap to cover every possible region generated by sensor nodes. Fig 1 shows example queries that find nodes or regions with certain temperature and humidity. Note the difference between our proposed region-based query(Fig 1(b)) and the query that is based on individual nodes(Fig 1(a)).

There are some challenging problems in REQUEST. First, since regions overlap and the number of regions is fairly large, it is not trivial to efficiently construct regions with a specified size in the query. A naive approach is to move a circle representing a region as a certain step size. However, this approach is too inefficient, and it is not easy to find appropriate step size. To solve this problem, we create the *SEC* (Smallest Enclosing Circle) index structure in the preprocessing phase, and construct regions by using the SEC index.

Second, the communication costs of forwarding sensing values and aggregated values can be considerably high due to a large number of regions. In order to process the region-based queries energy-efficiently, we use a hierarchical aggregation method [6] as a basic processing scheme. In the hierarchical aggregation method, we have a leader node and several non-leader nodes in each group, and the aggregation of each group is computed locally in a group. Since there are numerous regions in our environment, it is more beneficial to calculate an aggregated value for each region as early as possible. By doing so, we can reduce the size and the total number of messages to send to the basestation. Moreover, to minimize communication costs while gathering aggregated values, we need an algorithm that selects optimal leader nodes efficiently in terms of energy consumption. To determine optimal leader nodes, we consider some criteria such as the hop counts between nodes, the size of messages, and the number of regions covered. Based on these criteria, we formulate a leader selection problem, and design an algorithm that uses the idea of transformation into the set-cover problem.

Finally, we need a topology for communication between leader nodes and non-leader nodes. TAG-based global routing tree [10] is not appropriate for intra-region communication, since it is constructed in order for the basestation to collect the data of the entire network. For intra-region communication, it is required to construct a tree in order for the leader node to collect the data of non-leader nodes inside the region. Therefore, for each leader node, we build a new routing tree whose root node is the leader node, called query-initiated routing tree.

Our contributions in this paper are as follows:

- We propose region-based queries, a new type of queries which use a region as a primitive data unit. This type of queries are useful especially when individual sensor readings are not reliable or only a rough view of the monitoring environment is required. By adjusting the sizes of regions in the query, we can collect the data in various degrees of circumstantiality. Moreover, since regions can be overlapped in our approach, we can avoid that some important regions are missing. To the best of our knowledge, the region-based query is the first type of grouped aggregate query in which groups can be overlapped.
- We propose algorithms to efficiently process region-based queries. To construct regions efficiently in real time, we present an algorithm that utilize the Smallest Enclosing Circle index. In addition, we address the optimization problem for leader selection, and propose a corresponding algorithm to the problem by using the algorithm that solves the set-cover problem.
- Lastly, we present extensive experimental results that show the effectiveness and efficiency of our method.

The rest of the paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we propose REQUEST, and explain our processing method. In Section 4, we show experimental results, and we conclude our work in Section 5.

2 Related Work

Our region-based queries are similar to the spatial queries in sensor networks. These spatial queries in sensor networks have been actively reported [3,4,5,12]. For instance, in [12], they propose a distributed spatial index, called SPIX, over the sensor nodes to process spatial queries in a distributed manner. However, most of works about spatial queries in sensor networks have focused on using the predefined regions. In [12], spatial queries are used to answer questions such as “what is the average temperature in room-1?”. In contrast, our region-based queries ask questions such as “which regions with a fixed size have the average temperature lower than a certain threshold?". Thus, in REQUEST, regions are not predefined before the query is posed, but redefined whenever the size of region specified in the query is changed.

Our query processing scheme is also related to aggregate queries in sensor networks. Even if many works have been proposed to process aggregate queries, only a few works deal with grouped aggregate queries.

TAG [10] propose a grouped in-network aggregation method. In this method, as climbing up the global routing tree from the leaf nodes to the basestaiton, partial aggregated values for each group are computed and forwarded respectively at each node. However, this method has a problem if there are many groups to be maintained at each node. There have been other works [11,13] to improve the grouped in-network aggregation method, which are based on TAG. In these works, they focus on modifying the routing protocol, in order to reduce the size of messages. In [11], they propose group-aware network configuration algorithms. The key idea of these algorithms is selecting a node in the same group as a parent. By doing so, the number of partial aggregations that should be maintained at each node can be reduced. In [13], a multipath routing protocol is proposed in order for each node to send its data to different parents. These grouped in-network aggregation methods based on TAG consider only disjoint groups. However, in REQUEST, a large number of overlapped regions (groups) can be generated. The methods based on TAG are not directly applicable to our environment, since a node can belong to several groups simultaneously.

Zhuang et al. propose the max regional aggregate query [16] which is the most similar and related query with the proposed region-based query. The max regional aggregate query is for finding a region with the maximum aggregated value. To do that, they propose a sampling-based approach in which regions and nodes are sampled within a certain accuracy. However, they only consider regions at which individual nodes are centered. Therefore, some relevant regions to the user's interest can be missing. In addition, they only focus on reducing the data to process but do not deal with an efficient collection of the data.

Lee et al. in [9] present a framework to efficiently process group-by aggregate queries in sensor networks. They propose the compression scheme that uses the Haar wavelet in order to reduce the size of messages. However, they assume that groups and leader nodes are pre-determined, and only consider pre-clustered groups which are disjoint each other.

3 Region-Based Query Processing

In this section, we propose REQUEST, region-based query processing in sensor networks. The goal of REQUEST is gathering regional aggregated values energy-efficiently so that we can find some interest regions satisfying several conditions. The specification of the region-based query is as follows:

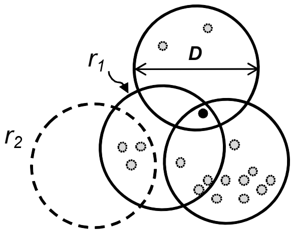


Fig. 2. 3 different regions in REQUEST

select	{region, aggfunc(attributes)}
from	sensors
group by	region(diameter D)
having	{having predicates}
sampling rate	{time of sampling interval}
duration	{maximum of sampling time}

We regard a region as a circle located in any places in the sensor network monitoring field. For the simplicity, we assume that the diameter of the circle is not larger than the communication range. We leave the problem which deals with regions with a larger size than the communication range for the future work. In fact, there are infinite number of regions in the monitoring field, even if every region has the same size. However, the number of regions can be limited since the number of sensor nodes is finite. In order to limit the number of regions reasonably, we formally define a region as follows:

Definition 1. *Let D be a diameter specified in the query. A region r is a maximal set of sensor nodes which are located within a circle having a diameter D. Since r is maximal, it is not contained in any other regions.*

Fig 2 shows that 3 different regions with the size D in REQUEST. We discard region r_2 since it is included in region r_1. Unlike existing grouped aggregate methods, regions can overlap in our approach. Through this, we can cover all areas which contain the sensor nodes deployed uniformly in the monitoring field. A node can belong to several different regions.

When there are a large number of groups or especially the selectivity of the having predicates in the query is relatively low, the earlier aggregation can have more benefits in terms of energy consumption. Therefore, we adapt a hierarchical aggregation method to process region-based queries more energy-efficiently. In this method, there is a leader node for each group, and non-leader nodes in the same group forward their sensing values, and finally aggregation of each group is computed in the leader node. Since regions overlap in our approach, a leader node can represent more than one region. In Fig 2, a dark-colored node is a leader node which covers 3 different regions.

The overall process of REQUEST comprises the following steps:

1. Regions and leader nodes are decided at the basestation.
2. The initial query message with the leader notification is sent to the entire network.
3. A query-initiated routing tree is constructed.
4. Every non-leader node sends its data to the leader node in the same region.
5. Leader nodes compute and forward the aggregation value for each region to the basestation.

3.1 Region Construction

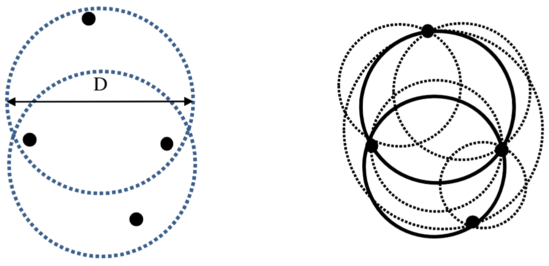
The region construction in REQUEST is to find every possible disjoint combination of sensor nodes located within a circle corresponding to the query. For a naive idea, we can find regions by moving the circle from the top left corner to the bottom right corner. However, it is difficult to determine the appropriate step size for covering the entire region since a region can be placed at an arbitrary position. Moreover, if the area of monitoring field is large, this naive method is extremely time-consuming.

Therefore, we propose an efficient region construction method that utilizes the SEC(Smallest Enclosing Circle) index. Our intuitive idea is that every region can be identified by a SEC which encloses every node inside the region. Moreover, every SEC can be defined by using at most 3 points [14].

For example, Fig 3 shows the relationship between regions and the corresponding SECs. Fig 3(b) shows every possible SECs given a set of nodes. To construct regions with a diameter D is identical to find maximal sets of nodes inside a circle having a diameter D by Definition 1. In Fig 3(a), there are 2 regions, which are sets of nodes. To generate these sets, we find the largest SECs among the SECs smaller than a circle with a diameter D. In Fig 3(b), the SECs in the solid line correspond to the regions in Fig 3(a).

In summary, first we build the SEC index of the sensor network, and then construct regions by using the SEC index.

To build the SEC index, we need an algorithm of finding a SEC that completely contains a given set of points. Finding SEC problem has been well-studied in the research area of mathematics. We use an algorithm from [14], which is



(a) Regions with a diameter D (b) Smallest Enclosing Circles

Fig. 3. Regions and the corresponding SECs

```
Algorithm BuildSecIndex
Input The set of all nodes $N = \{n_1, n_2, \ldots, n_m\}$
Output The SEC index $I$ which is sorted by a diameter
begin
1.  for each node $n_i$ in $N$
2.     for each node $n_j$ in $N$
3.         for each node $n_k$ in $N$
4.             $sec$ = FindSmallestEnclosingCircle($n_i$, $n_j$, $n_k$)
5.             $nodeSet$ = The set of nodes which are contained in $sec$
6.             Insert an index entry ($sec$.diameter, $sec$, $nodeSet$) to $I$
7.  return $I$
end
```

Fig. 4. Algorithm of Building SEC Index

simple to implement and has linear average time complexity. Fig 4 shows the algorithm of building the SEC index. Since every region can be defined by at most 3 nodes, the number of the SEC index entries is at most m^3, where m is the number of nodes. We manage the SEC index to be sorted by a diameter so as to construct regions efficiently in runtime. Each SEC index entry consists of a diameter, a SEC, and a set of nodes which are located inside SEC(Line 5). We assume that every node has a static position. Therefore, once the SEC index is built, we do not need further modification to the index.

By means of the SEC index, we can construct corresponding regions when the region-based query is posed. Fig 5 presents the algorithm of region construction. First, we find the largest SEC among the SEC index entries which have a smaller diameter than the diameter specified in the query(Line 1). From the largest SEC to the smallest SEC, we generate regions unless they are sub regions of already generated regions(Line 3∼5).

```
Algorithm ConstructRegions
Input The diameter of regions given in the query $D$, The SEC index $I$
Output The set of regions $R = \{r_1, r_2, \ldots\}$
begin
1.  $MaxSEC$ = A largest SEC among SECs in $I$ having a smaller diameter than $D$
2.  $MinSEC$ = A smallest SEC, thus, a node itself with a diameter 0
3.  for each SEC $sec$ from $MaxSEC$ to $MinSEC$ in $I$
4.     if $sec$ is not redundant then construct a region $r$ using $sec$
5.         Insert $r$ to $R$
6.  return $R$
end
```

Fig. 5. Algorithm of Region Construction

3.2 Leader Selection

To minimize the communication of REQUEST, it is important to select optimal leaders of each region. There are several requirements for optimal leader selection as follows:

– Leader nodes should be close enough to the basestation.
– Leader nodes should represent as many regions as possible.
– Distances between leader and non-leader nodes should be short enough.
– Leader nodes should cover all regions and sensor nodes.

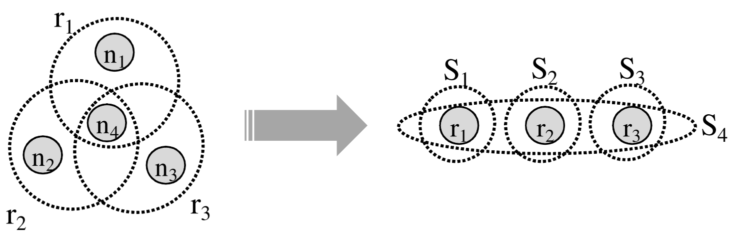


Fig. 6. Transformation from the leader selection problem to the set-cover problem

Based on these requirements, we formulate the leader selection problem as follows:

$$Minimize \quad \sum_{j \in L} (dist(root, j) \cdot |R_j| + \sum_{i \in N_j} dist(i, j))$$

$$Subject\ to: \quad \bigcup_{j \in L} R_j = \mathbb{R}, \ where\ \mathbb{R}\ is\ the\ set\ of\ entire\ regions.$$

$$\begin{aligned} L &: \ The\ set\ of\ leader\ nodes. \\ R_j &: \ The\ set\ of\ regions\ which\ include\ node\ j. \\ N_j &: \ The\ set\ of\ nodes\ which\ are\ in\ the\ same\ region\ with\ node\ j. \\ dist(i, j) &: \ The\ distance\ (hop\ counts)\ from\ node\ i\ to\ node\ j. \end{aligned}$$

In this formulation, we want to find the optimal L while minimizing the objective function at the same time. The objective function is the summation over j of the expected cost when we select a certain node j. Note that the size of messages between the basestation and the leader node j is $|R_j|$ times larger than that of messages between the leader node j and the non-leader nodes i. A unique constraint is for covering all regions and nodes.

To solve this problem, we adapt an idea that the facility location problem can be transformed into the weighted set-cover problem [8]. To transform the problem into the set-cover problem, it is required to define the set and the cost of the set. Intuitively, selecting a node as a leader in the leader selection problem is identical to choosing a set in the set-cover problem. Therefore, each node becomes a set and regions become elements of a set. Fig 6 shows that the sensor network in the left-side can be transformed into the instance of the set-cover problem in the right-side by means of our idea. If we select n_4 as a leader, we can cover 3 regions, which are r_1, r_2, and r_3. This is identical to choosing S_4 which can cover 3 elements in the set-cover problem.

The cost of each set is naturally derived from the objective function in our formulation.

$$C(S_j) = dist(root, j) \cdot |S_j| + \sum_{i \in N_j} dist(i, j)$$

```
Algorithm SelectLeaders
Input The set of all nodes $N$, The set of all regions $R$
Output The set of leader nodes $L$
begin
1.  $U = R$
2.  $X = \emptyset$
3.  for each node $n_i$ in $N$
4.      $S_i = \{r \mid r$ is a region which includes node $n_i\}$
5.      Insert $S_i$ to $X$
6.      $C(S_i) = \text{dist}(\text{root}, n_i) \cdot |S_i|$
7.      for each region $r_j$ in $S_i$
8.          for each node $n_k$ in $r_j$
9.              $C(S_i) = C(S_i) + \text{dist}(n_i, n_k)$
10. $L = \text{GreedySetCover}(U, X, C)$
11. return $L$
end
```

Fig. 7. Algorithm of leader selection

The set-cover problem is a well-known NP-complete problem, and has been actively studied in the algorithmic research fields. Among the several approximation algorithms that solve the set-cover problem in polynomial time, we use the set-greedy algorithm [2] which is the best known for the simplicity. In the set-greedy algorithm, we pick a set that covers the greatest number of elements not yet covered at each step. We skip the detail explanation of the set-greedy algorithm since it is beyond our work.

Fig 7 presents the algorithm of the leader selection, in which the set-greedy algorithm is called as a sub function.

3.3 Query-Initiated Routing Tree

In order for a leader node to communicate with non-leader nodes in the same region, it is required to build a routing tree for each leader node, called the query-initiated routing tree. When a new region-based query is posed, we construct a new routing tree for each leader node since the requested diameter can be changed. Basically, we apply to each leader node the routing method which is similar to the method used when the basestation constructs the global routing tree. The query-initiated routing tree construction performs the following steps:

1. Query messages with the leader nodes information are flooded in the entire network. (Fig. 8(a))
   – Each node is only aware whether itself is a leader or not.
2. The routing request messages (*leader_id*, *hop_count*) are broadcasted to the non-leader nodes within distance D from the leader nodes. (Fig. 8(b))
   – Receiver nodes should increase *hop_count* and broadcast to its neighbors.
   – We assume that every node can identify another node's location from the node id.
3. Each node designates the sender node as its parent node. (Fig. 8(c))
   – If multiple messages with the same leader node arrive at a node, the sender node of the message with the smallest *hop_count* is selected as the parent of the node.

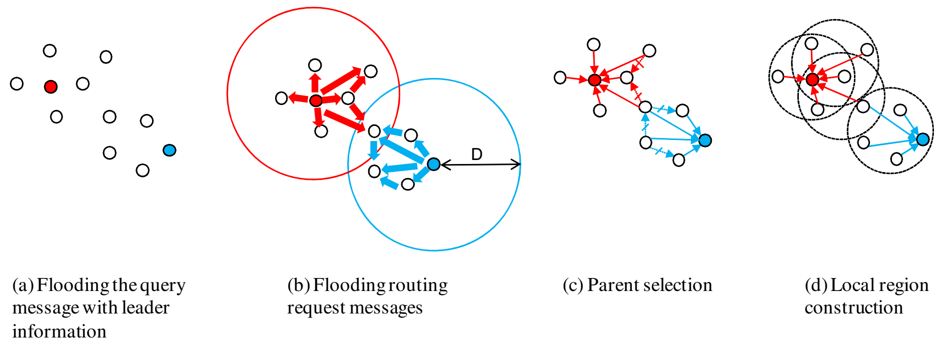

D
(a) Flooding the query message with leader information
(b) Flooding routing request messages
(c) Parent selection
(d) Local region construction

Fig. 8. The process of query-initiated routing tree construction

4. For each leader node, the local region construction is performed. (Fig. 8(d))
   – When the leader nodes received data message from the other nodes for the first time, they perform the local region construction based on the sender nodes of arrived messages.

Fig 8 shows these steps sequentially. Local region construction algorithm is almost same as the algorithm in Fig 5 except that we can only consider nodes inside the regions that the leader node belongs to and SECs are dynamically generated at each leader node.

4 Experiments

In order to investigate the effectiveness and efficiency of the proposed method in REQUEST, we conduct experimental evaluations.

4.1 Experimental Environment

We implement our method and comparison methods using our own simulator.

As a topology for experimental evaluations, we deploy total 100 sensor nodes in a grid environment. The area of each grid segment is $100m^2$, and 2 sensor nodes are randomly located in each segment. We set the communication range to $10m$.

Since our experiments are not affected by the spatial or temporal correlation in sensor networks, we use the synthetic data that is generated randomly. In our simulator, sensing values for each node at each sampling time are randomly changed in the range from 0 to 10.

For the convenience, we assume that a packet has a simple header information which comprises of a source address and destination addresses. Note that destination addresses can be more than one, if a node belongs to several regions simultaneously. A message consists of the region or node identifier and the corresponding sensing value(s). We set the node identifier and the region identifier to a sequential number and coordinates of the region center, respectively. A region

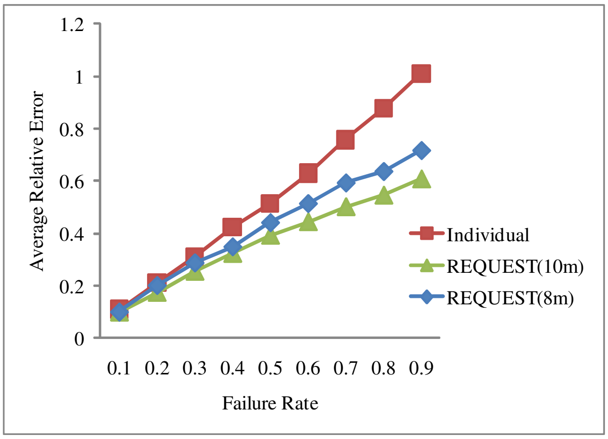


Fig. 9. Reliability

center can be decided from the corresponding SEC index entry. However, in our proposed method, we do not need the region identifier for a message, since the basestation can identify regions from the leader node id.

4.2 Reliability

We conduct experiments to evaluate the effectiveness of REQUEST. As a metric of the reliability, we use the average relative error rate. This metric is calculated as follows:

$$Average(relative\ error) = \frac{\sum_{i=1\ldots n} \frac{|v_i - v_i'|}{|v_i|}}{n}$$

In the above formula, n is the number of values, v is the original value, and v' is a value with noises due to the failure. Using the average relative error, we conduct experiments with various failure rates from 10% to 90%. We use the following region-based query for this experiment.

select region, AVG(temp) **from** sensors
group by region(D)
sampling rate 1 **duration** 100.

Fig 9 shows the results of experiments on the reliability. "Individual", "REQUEST(8m)", and "REQUEST(10m)" are the cases that D is 0m, 8m, and 10m, respectively. As the failure rate increases, the average relative error rates of all the cases also increase. However, REQUEST(8) and REQUEST(10) show better accuracy than Individual especially when the failure rate is high. This result shows that using the aggregated values in the regions is effective to reduce the effect of the node failures.

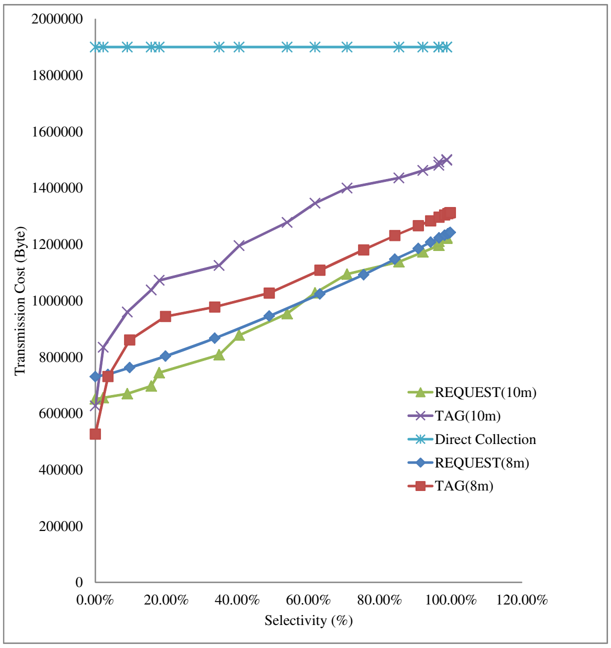


Fig. 10. Energy-efficiency

4.3 Energy-Efficiency

In sensor networks, consumed energy is the primary performance measure. Since the communication is the dominant factor in consuming energy, we use the amount of transmission as an efficiency metric. For the comparison system, we implement the grouped in-network aggregation method which is proposed in TAG [10]. In fact, in REQUEST, a node can belong to several groups at the same time, and each node can not know those groups before the query is posed. Groups can change according to the size of regions that is specified in the query. Additional communication to notify the group information to each node is needed. However, for the simplicity, we assume that every node knows its corresponding groups in advance for the comparison system, TAG.

The query used in this experiment is as follows:

select region, SUM(temp) **from** sensors
group by region(D)
having $0 \leqq \text{SUM(temp)} \leqq t$
sampling rate 1 **duration** 100

We conduct experiments with varying the selectivity of the having predicate. To do that, we change t in the range from the minimum of SUM(temp) to the maximum of SUM(temp). If t increases, the selectivity also increases. To apply effects of changing the selectivity to the comparison system equally, we implement TAG to exploit suppressing messages in the intermediate node. Like the experiments on the reliability, we test on two region sizes which are 8m and 10m.

Fig 10 shows that the cost of REQUEST is smaller than those of TAG and direct data collection in most cases. In the case that D is 8m, the difference between REQUEST and TAG is not large, because the number of regions that each node belongs to is smaller than that in the case that D is 10m. When the selectivity is close to zero, the communication cost of TAG is smaller than that of REQUEST. This is because in REQUEST, even if every aggregated values are filtered by the having predicate, the communication inside regions is still required. However, except for these cases, REQUEST is always better than TAG and direct data collection with regards to energy consumption.

5 Conclusion

In this work, we proposed a new type of query in sensor networks, called the region-based query. This type of queries are helpful to overcome noises in the sensor data, and to provide a macro view of a monitoring area. By permitting overlapping regions, we could deal with every possible regions which are generated by sensor nodes. In order to construct numerous regions efficiently, we used the SEC index that is built in the preprocessing phase. Moreover, to efficiently process the region-based query, we used a hierarchical aggregation method and addressed an optimization problem for leader selection with a solution algorithm by mapping the problem to the set-cover problem. Also, we built a new routing tree for each leader node, a query-initiated routing tree that enables intra-region communication. Finally, we showed that our proposed approach is effective and energy-efficient from the experimental results.

We plan to extend our work to deal with big size regions and aggregation of aggregation queries such as finding the average value among the maximum value of each region.

Acknowledgments. This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government (MEST) (No. 2010-0000863).

References

1. Precision agriculture, http://en.wikipedia.org/wiki/Precision_agriculture
2. Chvatal, V.: A greedy heuristic for the set-covering problem. Mathematics of Operations Research 4(3), 233–235 (1979)
3. Demirbas, M., Ferhatosmanoglu, H.: Peer-to-peer spatial queries in sensor networks. In: P2P 2003, pp. 32–39 (2003)
4. Dyo, V., Mascolo, C.: Adaptive distributed indexing for spatial queries in sensor networks. In: DEXA Workshops 2005, pp. 1103–1107 (2005)

5. Gupta, H., Zhou, Z., Das, S.R., Gu, Q.: Connected sensor cover: self-organization of sensor networks for efficient query execution. IEEE/ACM Transactions on Networking 14(1), 55–67 (2006)
6. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: HICSS 2000, pp. 8020–8029 (2000)
7. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: An application-specific protocol architecture for wireless microsensor networks. IEEE Transactions on Wireless Communications 1(4), 660–670 (2002)
8. Hochbaum, D.S.: Heuristics for the fixed cost median problem. Mathematical Programming 22(1), 148–162 (1982)
9. Lee, C.H., Chung, C.W., Chun, S.J.: Effective processing of continuous group-by aggregate queries in sensor networks. Journal of Systems and Software 83(12), 2627–2641 (2010)
10. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tag: a tiny aggregation service for ad-hoc sensor networks. SIGOPS Oper. Syst. Rev. 36(SI), 131–146 (2002)
11. Sharaf, A., Beaver, J., Labrinidis, A., Chrysanthis, K.: Balancing energy efficiency and quality of aggregate data in sensor networks. The VLDB Journal 13(4), 384–403 (2004)
12. Soheili, A., Kalogeraki, V., Gunopulos, D.: Spatial queries in sensor networks. In: GIS 2005, pp. 61–70 (2005)
13. Song, I., Roh, Y.J., Kim, M.H.: Content-based multipath routing for sensor networks. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5981, pp. 520–534. Springer, Heidelberg (2010)
14. Welzl, E.: Smallest enclosing disks (balls and ellipsoids). In: Maurer, H.A. (ed.) New Results and New Trends in Computer Science. LNCS, vol. 555, pp. 359–370. Springer, Heidelberg (1991)
15. Younis, O., Fahmy, S.: Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. In: INFOCOM 2004, pp. 629–640 (2004)
16. Zhuang, Y., Chen, L.: Max regional aggregate over sensor networks. In: ICDE 2009, pp. 1295–1298 (2009)

Efficient Distributed Top-k Query Processing with Caching

Norvald H. Ryeng, Akrivi Vlachou, Christos Doulkeridis, and Kjetil Nørvåg

Norwegian University of Science and Technology,
Department of Computer and Information Science,
Trondheim, Norway
{ryeng,vlachou,cdoulk,noervaag}@idi.ntnu.no

Abstract. Recently, there has been an increased interest in incorporating in database management systems rank-aware query operators, such as top-k queries, that allow users to retrieve only the most interesting data objects. In this paper, we propose a cache-based approach for efficiently supporting top-k queries in distributed database management systems. In large distributed systems, the query performance depends mainly on the network cost, measured as the number of tuples transmitted over the network. Ideally, only the k tuples that belong to the query result set should be transmitted. Nevertheless, a server cannot decide based only on its local data which tuples belong to the result set. Therefore, in this paper, we use caching of previous results to reduce the number of tuples that must be fetched over the network. To this end, our approach always delivers as many tuples as possible from cache and constructs a remainder query to fetch the remaining tuples. This is different from the existing distributed approaches that need to re-execute the entire top-k query when the cached entries are not sufficient to provide the result set. We demonstrate the feasibility and efficiency of our approach through implementation in a distributed database management system.

1 Introduction

Nowadays, due to the huge amount of available data, users are often overwhelmed by the variety of relevant data. Therefore, database management systems offer rank-aware query operators, such as top-k queries, that allow users to retrieve only a limited set of the most interesting tuples. Top-k queries [5,8,15] retrieve the k tuples that best match the individual user preferences based on a user-specified scoring function. Different scoring functions express the preferences of different users. Several applications benefit from top-k queries, including web search, digital libraries and e-commerce. Moreover, the high distribution of data raises the importance of supporting efficient top-k query processing in distributed systems.

In this paper, we propose a cache-based approach, called *ARTO*[1], for efficiently supporting top-k queries in distributed database management systems.

[1] Algorithm with Remainder TOp-k queries.

In large-scale distributed systems, the dominant factor in the performance of query processing is the communication cost, measured as the number of tuples transmitted over the network. Ideally, only the k tuples that belong to the result set should be fetched. Nevertheless, in the case of top-k queries, a server cannot individually decide which of its top-k local tuples belong to the global top-k result set of the query. In order to restrict the number of fetched tuples and reduce the communication costs, we employ caching of result sets of previously posed top-k queries. Each server autonomously maintains its own cache and only a summary description of the cache is available to any other server in the network.

In general, a top-k query is defined by a scoring function f and a desired number of results k, and these parameters differ between queries. Given a set of cached top-k queries in the system and a new query, the problem is to identify whether the results of cached queries are sufficient to answer the new query. To deal with this problem, we apply techniques similar to those of the view selection problem in the case of materialized views [15] in centralized database systems. Based on the cached queries, we need to decide whether the cached results *cover* the results of a new query. In this case, the query is answered from the cache and no tuples need to be transferred over the network. However, the major challenge arises when the query is not covered by the cached tuples.

Different from existing approaches [20] that require the servers to recompute the query from scratch, we do not evaluate the entire query, but we create a *remainder query* that provides the result tuples that are not found in cache. More detailed, we split the top-k query into a top-k' query ($k' < k$) that is answerable from cache, and a remainder next-$(k - k')$ query that provides the remaining tuples that were not retrieved from the top-k' query. To further optimize the query performance, we deliberately assign the top-k query to the server that is expected to induce the lowest network cost based on the locally cached tuples. To summarize, the contributions of this paper are:

- We propose a novel framework for distributed top-k queries that retrieves as many tuples k' ($k' < k$) as possible from the cache, and poses a *remainder query* that provides the remaining $k - k'$ tuples that are not found in cache.
- We present a novel method for efficiently computing remainder queries, without recomputing the entire top-k query.
- We propose a server selection mechanism that identifies the server that owns the cache with the most relevant entries for a given query.
- We evaluate our approach experimentally by integrating ARTO in an existing distributed database management system [14], and we show that our method significantly reduces communication costs.

The rest of this paper is organized as follows. In Section 2, we explain how this paper relates to previous work in this area. Section 3 presents preliminary concepts, and Section 4 presents our framework for distributed top-k query processing. Answering top-k queries from cache is outlined in Section 5. The remainder queries are described in Section 6, while Section 7 presents the server selection mechanism. Results from our experimental evaluation are presented in Section 8, and in Section 9 we conclude the paper.

2 Related Work

Centralized processing of top-k queries has received considerable attention recently [2,5,8,15]. For a comprehensive survey of top-k query processing we refer to [16]. Hristidis et al. [15] discuss how to answer top-k queries from a set of materialized ranked views of a relational table. Each view stores all tuples of the relation ranked according to different ranking functions. The idea is to materialize a set of views based on a requirement either on the maximum number of tuples that must be accessed to answer a query, or on the maximum number of views that may be created. When a query arrives, one of these views is selected to be used for answering the top-k query. In [11], the materialized views of previous top-k queries (not entire relations) are used to answer queries, as long as they contain enough tuples to satisfy the new query. For each incoming query, the view selection algorithm chooses a set of views that will give an optimal (in terms of cost) execution of the proposed LPTA algorithm. A theoretical background on view selection is given in [3], providing theoretical guarantees whether a view is able to answer a query or not. However, the algorithms that are presented only allow a query to be answered from views if the views are guaranteed to provide the answer.

In distributed top-k query processing, the proposed approaches can be categorized based on their operation on vertically [6,9,12,17,18] or horizontally [1,4,19,20] distributed data. In the case of vertically distributed data, any server maintains only a subset of the attributes of the complete relation. Then, each server is able to deliver tuples ranked according to any scoring function that is applied on one or more of its attributes [9,12,17]. The TPUT algorithm [6] focuses on limiting the number of communication round-trips, and this work has later been improved by KLEE [18].

In the case of horizontally distributed data, each server stores a subset of the tuples of the complete relation, but for each tuple all attributes are maintained. In [1], a broadcasting technique for answering top-k queries in unstructured peer-to-peer networks is presented. For super-peer topologies, Balke et al. [4] provides a method using indexing to reduce communication costs. This method requires all super-peers to process queries, unless exactly the same query reappears. SPEERTO [19] pre-computes and distributes skyline result sets of super-peers in order to contact only those super-peers that are necessary at query time. BRANCA [20] is a distributed system for answering top-k queries. Caching of previous intermediate and final results is used to avoid recomputing parts of the query. The cache is used much in the same way as the materialized views in [3,11,15], but on intermediate results of the query. This means that some servers in the system must process the query from scratch, while others may answer their part of the same query from cache. The main difference between ARTO and other caching approaches, such as BRANCA, becomes clear in the hard cases, when the query cannot be answered by the cache. ARTO still uses the part of the cache that partially answers the query and poses a remainder query for the remaining tuples, without the need to process the query from scratch, as in the case of BRANCA.

Finally, our techniques for answering top-k queries relate to stop-restart of query processing [7,10,13]. These methods assume that some of the result tuples are already produced and restart processing from where the original query stopped. Our remainder queries differ by not restarting an existing top-k query but a query that was partially answered by cached tuples.

3 Preliminaries

Top-k queries are defined based on a monotone function f that combines the individual scores into an overall scoring value, that in turn enables the ranking (ordering) of tuples. Given a relation R, which consists of n attributes a_i, the result set of a top-k query $Q = \langle R, f, k \rangle$ contains k tuples such that there exists no other tuple in R with better score than the k tuples in the result set. The relation R may be a base relation or the result of an algebra operator, i.e., the result of a join. The most commonly used scoring function is the weighted sum function, also called linear. Each attribute a_i is associated with query-dependent weight w_i indicating a_i's relative importance for the query. Furthermore, without loss of generality, we assume that for any tuple t and any attribute a_i the values $t(a_i)$ are scaled to $[0, 1]$. The aggregated score $f(t)$ for a tuple t is defined as a weighted sum of the individual scores: $f(t) = \sum_{i=1}^{n} w_i t(a_i)$, where $w_i \geq 0$ $(1 \leq i \leq n)$, and $\exists j$ such that $w_j > 0$. The weights represent the relative importance of different attributes, and without loss of generality we assume that $\sum_{i=1}^{n} w_i = 1$. Thus, a linear top-k query Q is defined by a vector $\boldsymbol{w_Q}$ and the parameter k. The ranked tuples can be delivered in either ascending or descending order, but for simplicity, we will only consider descending order in this paper. Our results are also valid in the ascending case.

A tuple t of R can be represented as a point in the n-dimensional Euclidean space. Furthermore, given a top-k query $Q = \langle R, f, k \rangle$ defined by a linear scoring function, there exists a one-to-one correspondence between the weighting vector $\boldsymbol{w_Q}$ and the hyperplane which is perpendicular to $\boldsymbol{w_Q}$. We refer to the (n-1)-dimensional hyperplane, which is perpendicular to vector $\boldsymbol{w_Q}$ and crosses the k-th result tuple, as the *query plane* of $\boldsymbol{w_Q}$, and denote it as L_Q. All points on the query plane L_Q have the same scoring value for $\boldsymbol{w_Q}$. A 2-dimensional example is depicted in Fig. 1. Processing the top-k query Q is equivalent to sweeping the line L_Q from the upper right corner towards the lower left corner. Each time L_Q meets a tuple t, this tuple is reported as the next result tuple. When L_Q meets the k-th tuple, the complete result set has been retrieved.

4 ARTO Framework

In this paper, we assume a distributed database system where the relations are horizontally fragmented over multiple servers. In more details, each relation R is fragmented into a set of fragments $R_1, R_2, \ldots, R_f$ and each fragment R_i consists of a subset of tuples of the relation R. Our approach is generic and imposes no further constraints on the way fragments are created or whether

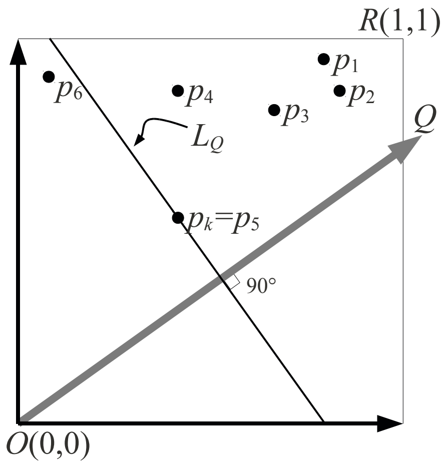


Fig. 1. 2D representation of query and data space

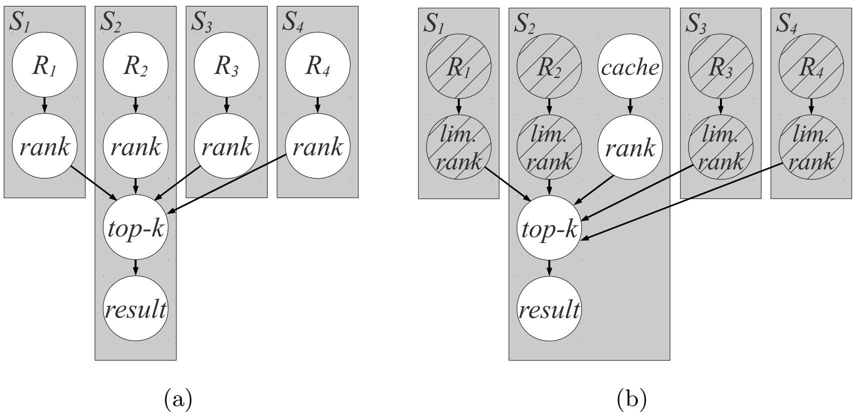


Fig. 2. (a) Query plan for distributed top-k query (b) Transformed query plan

they are overlapping or not. Furthermore, each server may store fragments of different relations. Any server can pose a top-k query and we refer to that server as *querying server*. During query processing, the querying server may connect to any other server. Thus, no routing paths are imposed on the system other than those of the physical network itself. The only assumption of ARTO is that there exists a distributed *catalog* accessible to all servers, which indexes the information about which server stores fragments of each relation R. Such a distributed catalog can be implemented using a distributed hash table (DHT).

To answer a top-k query over a relation R, the querying server first locates those servers that store fragments of R by using the catalog, and constructs a query plan such as the one in Fig. 2(a). In our example, S_2 is the querying server and the relation R is fragmented in four fragments $R_1, \ldots, R_4$ stored on servers

$S_1, \ldots, S_4$ respectively. Based on the query plan, each fragment R_i is scanned in ranked order (denoted in Fig. 2(a) as rank), and the top-k operator reads tuples one by one, until the k highest scoring tuples have been retrieved. In more details, the top-k operator maintains a sorted output queue and additionally a list containing the score of the last tuple from each server. Since the tuples read from R_i are in ranked order, whenever a tuple in the output queue has a higher score than all scores in the list, it can safely be output as a result tuple. Thus, the top-k tuples are returned incrementally. Moreover, the top-k operator reads the next tuple from the fragment R_i with the tuple with the highest score in the list. Therefore, the top-k operator reads as few input tuples as possible from the fragments R_i.

This is the basic approach of answering top-k queries in a distributed data management system. Since it is important to minimize the network cost of query processing, ARTO uses a caching mechanism to take advantage of previously answered top-k queries. Thus, ARTO avoids retrieving tuples from other servers, when the cached tuples are sufficient to answer the new query. To this end, each server maintains its own cache locally, and caches the queries (and their results sets) that were processed by itself. During query processing, the querying server first uses its cache to detect whether the cached tuples are sufficient to answer the given top-k query (see Section 5). Even if the cached tuples are not sufficient, ARTO minimizes the transferred data by using as many cached tuples as possible and retrieving only the missing tuples from the remote servers through the novel use of *remainder queries* (see Section 6). To this end, the query plan is rewritten in order to take advantage of the local cache. The result of such a query transformation is shown in Fig. 2(b). Compared to the initial query plan, the top-k operator additionally retrieves tuples from the cache and performs a limited scan from the relation fragments, thus transferring only tuples that are not cached.

The combination of cached tuples and remainder queries allows ARTO to reduce the number of transferred tuples. The exact number of transferred tuples depends on the similarity of cached queries to the new query. Thus, in order to improve further the query processing performance, we extend ARTO with a *server selection* mechanism, which assigns the new query to the server with the most similar cached query. In order to facilitate this mechanism, each server publishes descriptions of its cached queries in the distributed catalog. Then, the querying server first detects the server with the most similar cached query, and re-assigns the new query to this server (see Section 7).

In rest of this paper, we assume that data tuples are not updated, inserted or deleted during query processing. This means that the cache always will be up-to-date. Techniques that enforce cache consistency can be adopted in a straightforward way, as they are orthogonal to our work.

5 Answering Top-k Queries from Cache

In ARTO, each server autonomously maintains its own cache. More specifically, after answering a top-k query and retrieving the entire result set, the query

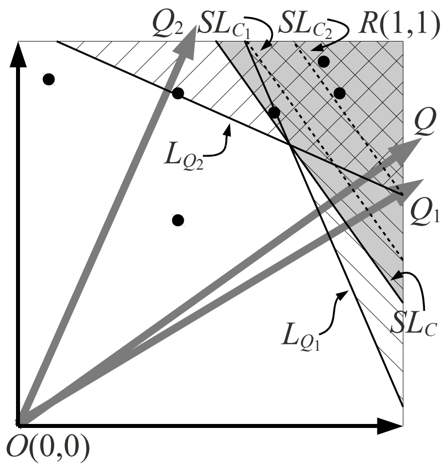


Fig. 3. Cache containing the cache entries of two queries

originator is able to cache the query result. The cache $\mathcal{C} = \{C_i\}$ maintains a set of m *cache entries* C_i. Each cache entry $C_i = \{Q_i, b_i, \{p_1, \ldots, p_{k_i}\}\}$ is defined by a query $Q_i = \{R, f_i, k_i\}$, the tuples $\{p_1, \ldots, p_{k_i}\}$ that belong to the result set of Q_i, and a *threshold* b_i which is the scoring value of point p_{k_i} with respect to f_i, i.e., $b_i = f_i(p_{k_i})$. Consequently, any tuple p of the cache entry C_i has score $f_i(p) \geq b_i$. Notice that the description of a cached entry C_i that is published in the catalog consists only of $\{Q_i, b_i\}$, without the individual result tuples. For the sake of simplicity, we assume that all cache entries refer to the same relation R. Obviously, given a query $Q = \{R, f, k\}$, only cache entries that refer to relation R are taken into account for answering Q.

Fig. 3 shows a server's cache $\mathcal{C}$ that contains two cache entries, C_1 and C_2. Query Q_1 corresponds to a top-3 query, while Q_2 is a top-4 query with different weights. Their corresponding lines, L_{Q_1} and L_{Q_2}, stop at the k-th point for each query respectively.

5.1 Basic Properties

In this section, we analyze when the query results of a cache $\mathcal{C}$ are sufficient to answer a top-k query Q. When this situation occurs, we say that the cache *covers* the query. Given a query $Q = \{R, f, k\}$, we identify three cases of covering: (1) a cache entry C_i covers a query defined by the same function ($f = f_i$), (2) a cache entry C_i covers a query defined by a different function ($f \neq f_i$), and (3) a set of cache entries $\{C_i\}$ cover a query defined by a different function ($f \neq f_i$, $\forall i$).

In the first case, if there exists a cache entry C_i such that the weighting vectors that define f and f_i are identical and $k \leq k_i$, then Q can be answered from the result of the cache entry C_i. More specifically, the first k data points of the cache entry C_i provide the answer to Q.

In the second case, we examine if a cache entry covers a query defined by a different function. To this end, we use the concept of *safe area* [3] SA_i of a cache entry C_i.

Definition 1. ***(Safe area)*** *The safe area SA_i of a cache entry C_i with respect to a query Q is the area defined by the right upper corner of the data space and the $(n-1)$-dimensional hyperplane SL_{C_i} that is perpendicular to the query vector, intersects the query plane L_{Q_i}, and has the largest scoring value for Q between all candidate hyperplanes.*

In Fig. 3, the lines that define the safe areas for C_1 and C_2 with respect to Q are shown as SL_{C_1} and SL_{C_2}, respectively. Given a query Q, a cache entry C_i is sufficient to answer a query Q, if it holds that the safe area SA_i of the cache entry C_i contains at least k data points. This means that there cannot be any other tuples in the result set of Q that have not been retrieved by the query Q_i, because the safe area has been scanned during the processing of Q_i. For example, in Fig. 3, both cache entries are sufficient for answering the query Q for $k = 2$, but none of those is sufficient to answer the query Q for $k = 3$.

The third case deals effectively with the previous situation. Several cache entries need to be combined to answer the top-k query, since a single cache entry is not sufficient. To determine whether a set of cache entries can be used to answer a top-k query, we define the concept of *cache horizon*.

Definition 2. ***(Cache horizon)*** *The cache horizon of a cache $\mathcal{C} = \{C_i\}$ is defined as the borderline of the area defined by the union of query planes L_{Q_i}.*

The cache horizon represents the border between the points that are cached and those that are not. Points behind the cache horizon (towards the right upper corner of the data space) are contained in at least one cached entry, while points beyond the cache horizon (near the origin of the data space) have to be retrieved from the relation R that is stored at different servers. In Fig. 3, the cache horizon is defined by the lines L_{Q_1} and L_{Q_2} and the enclosed area has been examined to answer queries Q_1 and Q_2. In order to determine if the result set of Q is behind the cache horizon and can be answered by combining more than one cache entry, we define the *limiting point* of the cache.

Definition 3. ***(Limiting point)*** *The limiting point of a cache $\mathcal{C}$ is the point, where the hyperplane $SL_{\mathcal{C}}$ perpendicular to the query vector intersects the cache horizon, when $SL_{\mathcal{C}}$ moves from the right upper corner of the data space towards the origin.*

The area defined by the hyperplane $SL_{\mathcal{C}}$ and the right upper corner of the data space is called *safe area of the cache horizon*. If this area contains more than k data points, then Q can be answered by combining more than one cache entry.

Given a cache $\mathcal{C}$ with m cache entries $C_1, C_2, \ldots, C_m$, the limiting point of the horizon with respect to a query Q can be identified using linear programming. We construct a constraint matrix $\boldsymbol{H}$ and right-hand-side values $\boldsymbol{b}$ from the weights and thresholds of the m cache entries:

$$\boldsymbol{H} = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & & & \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Given a query Q, the objective is to maximize $f = \boldsymbol{w_Q}^{\mathrm{T}}\boldsymbol{a}$, subject to the constraints $\boldsymbol{Ha} \leq \boldsymbol{b}$ and $0 < a_i < 1$, $\forall a_i$. The solution of this linear programming problem provides the coordinates of the limiting point. By applying the scoring function f defined by Q, we get the *cache score* $b = f(p)$ of the limiting point p. If at least k cached points p_i exist such that $f(p_i) \geq b$, then the entire result set of query Q is retrieved from the cache entries.

5.2 Cache Replacement Policy

A first observation regarding a cache entry $C_i \in \mathcal{C}$ is that it can become *redundant* due to other cache entries in $\mathcal{C}$. More formally, C_i becomes redundant if its query Q_i is covered by a set of other cache entries. Redundant cache entries can be evicted from the cache without affecting the cache's ability to answer queries. Identifying whether a cache entry C_i is redundant is achieved by solving a linear programming problem. More detailed, the objective is to maximize $\boldsymbol{w_{C_i}}^{\mathrm{T}}\boldsymbol{a}$, subject to $\boldsymbol{H'a} \leq \boldsymbol{b'}$ and $\forall a_j : 0 < a_j < 1$, where $\boldsymbol{H'}$ and $\boldsymbol{b'}$ describe the combined horizon of all cache entries except C_i. Thus, we find the limiting point p of the cache when C_i is ignored. If $b_i > f_i(p)$, the cache entry C_i is redundant and can be removed.

Applying a traditional cache replacement policy, such as LRU, is inappropriate due to the unique characteristics of our cache. The reason is that answering a top-k query from the cache may require combining tuples from more than one cache entry. Consequently, cache entries are utilized collectively, rendering any policy based on usage statistics of individual cache entries ineffective.

Motivated by this, we introduce a new cache replacement policy named *Least-Deviation Angle* (LDA), which is particularly tailored to our problem. After removing redundant entries, LDA determines the priority of a cache entry to be evicted based on deviation from the equal-weights vector $\boldsymbol{e}^{\mathrm{T}} = (1, 1, \ldots, 1)$. For each cache entry C_i, the angle $\theta_i = \arccos(\hat{\boldsymbol{w_{C_i}}} \cdot \hat{\boldsymbol{e}})$ between $\boldsymbol{e}$ and C_i is calculated and used as a measure of deviation. The entry C_i with the largest θ_i is replaced first. Intuitively, LDA penalizes cache entries that have low probability to be highly similar to other queries.

6 Remainder Queries

In the previous section, we described in which cases the entries of the cache are sufficient for retrieving the entire result set of a query Q. When this occurs, no networking cost exists for answering the query. In the case where only $k' < k$ tuples t are retrieved from the cache for which the inequality $f(t) \geq b$ holds (b is the cache score), the cache fails to return the complete result set. Then, instead of executing the entire query Q from scratch, ARTO executes a

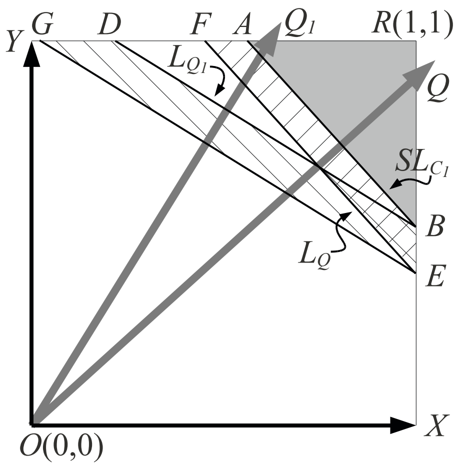


Fig. 4. Areas examined by the remainder query vs. restarting a query Q_1

remainder query that retrieves only the $k - k'$ missing tuples and transfers only the necessary tuples to provide the complete result set. We first provide a short discussion showing that is more beneficial to execute a remainder query, rather than restarting a cached query $Q_i = \{R, f_i, k_i\}$ and retrieving additional tuples, so that the k tuples of Q are retrieved. Then, we define the remainder query and explain how it will be processed in order to minimize the network consumption.

6.1 Discussion

In this section, we discuss the issue whether it is more beneficial to restart a query of a cache entry C_i than posing a remainder query. Fig. 4 depicts a cache containing one cache entry C_1 that covers the data space until the line L_{Q_1} (line DB). A query Q is posed, and the points in the cache entry until the line SL_{C_1} (line AB) are used for answering the query Q. If fewer than k tuples are enclosed in ABR, additional uncached tuples must be retrieved from remote servers. We consider two options for retrieving the remaining tuples. The first alternative is to pose a remainder query that would scan the part $FEBA$ of the data space. Since the query is executed based on the given weighting vector of Q, we can stop after retrieving k tuples exactly, i.e., at the query line L_Q (FE). The other alternative is to restart the cached query Q_1. In this case, we can take advantage of all k_1 data points of the cache entry C_1 (i.e., we save the cost of scanning DBA). On the other hand, in order to be sure that we have retrieved all tuples of the result set of Q we have to scan a larger area at least until the line GE.

If data is uniformly distributed, the number of tuples retrieved is proportional to the area of the data space that is scanned. For the sake of simplicity, we assume that the query line of any query lies in the the upper right triangle of the data space. This means that we have scanned less than half the data space, in order to retrieve the result set of any query, which is an acceptable assumption since usually the values of k are small. In our example, the area of $FEBA$ is smaller than the area of $GEBD$, and the retrieved tuples are expected to be fewer when the remainder query is used. In the following, we prove that this always holds for the 2-dimensional case, when the query line does not cross the diagonal line XY. Similar conclusions can been drawn for arbitrary dimensionality.

Theorem 1. *Given a 2-dimensional data space, if all query lines do not cross the diagonal line XY, a smaller area is scanned if the remainder query is executed than if continuing a cached query.*

Proof. Using the areas of Fig. 4, it suffices to show that the area of trapezoid $FEBA$ is smaller than the area of trapezoid $GEBD$. The two trapezoids share one common side, namely EB. Furthermore, it is always the case that $BD > BA$ and $GE > FE$. Based on Thales' theorem about the ratios of line segments that are created if two intersecting lines are intercepted by a pair of parallels, we derive that $\frac{FA}{AR} = \frac{EB}{BR}$ (1) and $\frac{GD}{DR} = \frac{EB}{BR}$ (2). From (1) and (2) we conclude that $\frac{FA}{AR} = \frac{GD}{DR}$. Since $DR > AR$, we derive that $GD > FA$. Therefore, three sides of $FEBA$ are smaller than the corresponding three sides of $GEBD$ and the remaining fourth side BE is common. Hence, the area of $FEBA$ is smaller than the area of $GEBD$.

6.2 Processing of Remainder Queries

Given a query Q and a cache score b, a remainder query is defined as $Q' = \langle R, f, k - k', b \rangle$, where k' is the number of cached tuples p such that $f(p) \geq b$. Any server S_i that stores a fragment R_i of the relation R receives the remainder query Q'. Independently from the implementation of the top-k operator at S_i, the server S_i transfers to the querying server only tuples p such that $f(p) \leq b$. Thus, it avoids transferring tuples that are already cached and lie in the safe area of the querying server.

To further limit the number of transferred tuples to the querying server, S_i filters out some of the locally retrieved tuples by using the cache horizon before transferring them. Even though some tuples lie outside the safe area, they are available at the querying server in some cache entry. For example, in Fig. 4, the remainder query has to start scanning the data space from the line SL_{C_1} until k tuples are retrieved, i.e., the remainder query fetches new tuples until the query line L_Q. Nevertheless, the points that fall in the triangle DBA are already available at the querying server in the cache entry C_1. These tuples do not need to be transferred, thus minimizing the number of transferred data. In order for S_i to be able to filter out tuples based on the cache horizon, S_i retrieves the descriptions of all cache entries from the querying server. Then, all tuples p such that there exists a cache entry C_i such that $f_i(p) > b_i$ are not transferred to the querying server, since these tuples are stored locally in the cache. The querying server combines the tuples received from the servers S_i with the tuples in the cache and produces the final result set of the query Q. To summarize, the cache horizon is used to limit the remainder query, which means that the whole cache is exploited and a minimal number of tuples is fetched from other servers.

Algorithm 1. Server selection

```
1: Input: Query $Q = \{R, f, k\}$, Servers $\mathcal{S}$
2: Output: Server $S^*$ that will process $Q$
3: $S^* \leftarrow null$, $minScore \leftarrow \infty$
4: for $(\forall S_i \in \mathcal{S})$ do
5:    $\{(Q_j, b_j)\} \leftarrow catalog$.getCacheDesc($S_i$)
6:    $score(S_i) \leftarrow$ computeLimitingPoint($\{(Q_j, b_j)\}$)
7:    if $(score(S_i) < minScore)$ then
8:       $S^* \leftarrow S_i$
9:       $minScore \leftarrow score(S_i)$
10:   end if
11: end for
12: return $S^*$
```

7 Server Selection

The problem of server selection is to identify the best server for executing the top-k operator. While the rank scan operators must be located at the servers that store the relation fragments, the top-k operator can be placed on any server. Our server selection algorithm assigns the top-k operator to the server that results in the most cost-efficient query execution in terms of network cost.

Intuitively, the best server S^* to process the top-k query $Q = \{R, f, k\}$ is the one that can return as many as possible from the k result tuples from its local cache, thereby reducing the amount of the remaining result tuples that need to be fetched. To identify S^*, we need to inspect the cache entries for each server. This operation is efficiently performed using the distributed catalog. In more detail, the catalog can report the descriptions of cache entries $\mathcal{C} = \{C_i\}$ of any server, where a description of C_i consists of $\{Q_i, b_i\}$. Based on this information, the limiting point of the server is calculated, as described in Section 5. Based on the limiting point, we compute the score of each server by applying the function f of the query Q. The server S^* with the smallest score is selected because this server has the largest safe area and therefore is the best candidate to process the top-k query. Algorithm 1 provides the pseudocode for the server selection mechanism.

8 Experiments

In this section, we present an experimental evaluation of ARTO. We have implemented ARTO into the DASCOSA-DB [14] distributed database management system and use this implementation to investigate the effect of different parameters, query workloads and datasets.

Experimental setup. DASCOSA-DB provides a global distributed catalog based on a distributed hash table, and this catalog was used to implement publishing and lookup of cache entries' descriptions. Experiments were performed for varying a) number of servers, b) values of k, and c) cache size. We used three

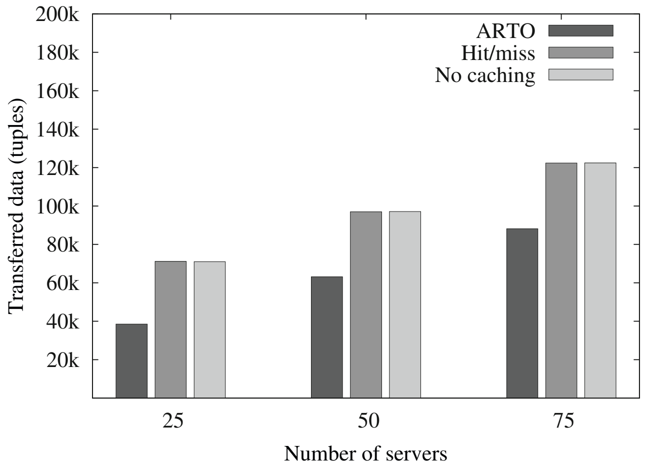


Fig. 5. Transferred data for 1,000 queries and uniform data distribution

datasets, with uniform, correlated and anti-correlated distributions. Each dataset consisted of 1,000,000 5-dimensional tuples. The datasets were distributed horizontally and uniformly over all servers. A separate querying server issued queries to the system and received the results. The weights of the queries were uniformly distributed.

Each experiment was performed both without caching and with ARTO enabled. In addition, we did experiments with a hit/miss implementation where the cache was used only if it were sufficient to answer the complete query. This is conceptually similar to previously proposed methods, e.g., BRANCA [20]. We measured the number of tuples accessed a) locally on the querying server using its cache, and b) from remote servers.

Varying number of servers. In this experiment, ARTO was evaluated with uniformly distributed, correlated and anti-correlated datasets. Each dataset was distributed over 25, 50 and 75 servers. A workload of 1,000 queries with uniformly distributed weights and $k = 50$ were issued. Each server had 10,000 bytes cache, which allows for 4 complete top-50 results to be cached at each server.

Fig. 5 shows the total number of tuples that are transferred over the network for the query workload using a uniform dataset. We observed similar results for correlated and anti-correlated datasets, which hints that the performance of our approach is stable across different data distributions. The combination of server selection with remainder queries causes a major improvement in network communication costs, even with such a small cache size (4 cache entries). The advantage of ARTO is clearly demonstrated when comparing to the hit/miss strategy, which performs poorly, as it requires k tuples in the safe area to use the cache. Since cache misses are frequent, the complete top-k query has to be executed. The results of hit/miss are just barely better than without caching, while ARTO achieves significant improvements.

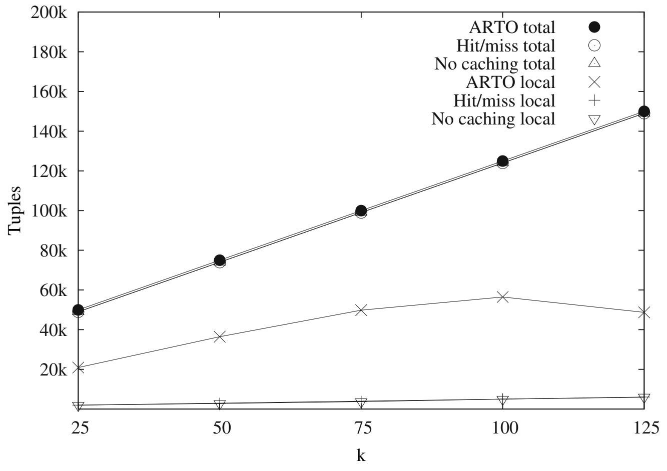


Fig. 6. Results of queries with varying k

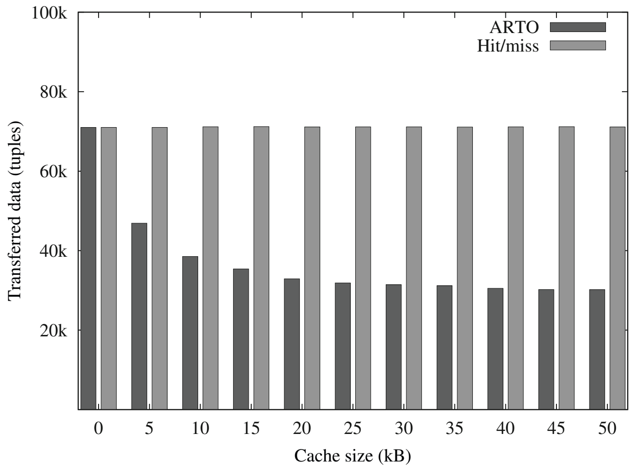


Fig. 7. Results of queries with varying cache size

Varying *k*. The size of k affects the gain that can be obtained from caching. If k is very small, there are not that many remote accesses that can be replaced by local accesses. In this experiment, the caching method was tested with varying values for k. A uniform dataset of 1,000,000 tuples on 25 servers was used. Each site had 10,000 bytes cache. The results displayed in Fig. 6 show how the number of total and local accesses increases with increasing k. ARTO always accesses significantly more local tuples compared to the competitor approaches. Around $k = 100$, the number of local tuples accessed starts to decrease. This is because the cache is of a limited size. With $k = 100$, only two complete top-k results fit in cache. Even in this extreme case, ARTO still manages to access a high percentage of the total tuples from the local cache, thereby saving communication costs.

Cache size. In order to study the effect of cache size in more detail, we performed an experiment where we gradually increased the cache size up to 50,000 bytes, i.e., more than 20 complete results. We fixed $k = 50$ and used a uniform dataset of 1,000,000 tuples on 25 servers. The results are shown in Fig. 7. As the cache size increases, more top-k queries can be cached, thus enlarging the safe area. Consequently, ARTO reduces the number of transferred data (remote tuples accessed). In contrast, the hit/miss strategy always results in cache misses and cannot reduce the amount of transferred data.

9 Conclusion

In this paper, we present ARTO, a novel framework for efficient distributed top-k query processing. ARTO relies on a caching mechanism that reduces the network communication costs significantly by retrieving as many tuples as possible from the local cache. In order to retrieve the missing tuples, we define the remainder query that transfers only the tuples that are not stored in the cache by filtering out tuples based on the cache horizon. Moreover, ARTO provides a server selection mechanism that assigns a new top-k query to the most promising server. We have implemented our framework in the DASCOSA-DB database management system. The results of the experiments show considerable improvements in network communication costs.

Acknowledgments

The authors would like to express their thanks to Jon Olav Hauglid for help with the implementation in DASCOSA-DB and João B. Rocha-Junior for providing the dataset generator.

References

1. Akbarinia, R., Pacitti, E., Valduriez, P.: Reducing network traffic in unstructured P2P systems using top-k queries. Distributed and Parallel Databases 19(2-3), 67–86 (2006)
2. Akbarinia, R., Pacitti, E., Valduriez, P.: Best position algorithms for top-k queries. In: Proceedings of VLDB (2007)
3. Baikousi, E., Vassiliadis, P.: View usability and safety for the answering of top-k queries via materialized views. In: Proceedings of DOLAP (2009)
4. Balke, W.T., Nejdl, W., Siberski, W., Thaden, U.: Progressive distributed top-k retrieval in peer-to-peer networks. In: Proceedings of ICDE (2005)
5. Bruno, N., Chaudhuri, S., Gravano, L.: Top-k selection queries over relational databases: Mapping strategies and performance evaluation. ACM Trans. Database Syst. 27(2), 153–187 (2002)
6. Cao, P., Wang, Z.: Efficient top-k query calculation in distributed networks. In: Proceedings of PODC (2004)
7. Chandramouli, B., Bond, C.N., Babu, S., Yang, J.: Query suspend and resume. In: Proceedings of SIGMOD (2007)

8. Chaudhuri, S., Gravano, L.: Evaluating top-k selection queries. In: Proceedings of VLDB (1999)
9. Chaudhuri, S., Gravano, L., Marian, A.: Optimizing top-k selection queries over multimedia repositories. IEEE Trans. on Knowledge and Data Engineering 16(8), 992–1009 (2004)
10. Chaudhuri, S., Kaushik, R., Ramamurthy, R., Pol, A.: Stop-and-restart style execution for long running decision support queries. In: Proceedings of VLDB (2007)
11. Das, G., Gunopulos, D., Koudas, N., Tsirogiannis, D.: Answering top-k queries using views. In: Proceedings of VLDB (2006)
12. Güntzer, U., Balke, W.T., Kießling, W.: Optimizing multi-feature queries for image databases. In: Proceedings of VLDB (2000)
13. Hauglid, J.O., Nørvåg, K.: PROQID: Partial restarts of queries in distributed databases. In: Proceedings of CIKM (2008)
14. Hauglid, J.O., Nørvåg, K., Ryeng, N.H.: Efficient and robust database support for data-intensive applications in dynamic environments. In: Proceedings of ICDE (2009)
15. Hristidis, V., Koudas, N., Papakonstantinou, Y.: PREFER: A system for the efficient execution of multi-parametric ranked queries. In: Proceedings of SIGMOD (2001)
16. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing techniques in relational database systems. ACM Comput. Surv. 40(4) (2008)
17. Marian, A., Bruno, N., Gravano, L.: Evaluating top-k queries over web-accessible databases. ACM Trans. Database Syst. 29(2), 319–362 (2004)
18. Michel, S., Triantafillou, P., Weikum, G.: KLEE: A framework for distributed top-k query algorithms. In: Proceedings of VLDB (2005)
19. Vlachou, A., Doulkeridis, C., Nørvåg, K., Vazirgiannis, M.: On efficient top-k query processing in highly distributed environments. In: Proceedings of SIGMOD (2008)
20. Zhao, K., Tao, Y., Zhou, S.: Efficient top-k processing in large-scaled distributed environments. Data and Knowledge Engineering 63(2), 315–335 (2007)

Exploiting Correlation to Rank Database Query Results*

Jaehui Park and Sang-goo Lee

Seoul National University, 599 Gwanak-ro, Seoul, Korea
`{jaehui,sglee}@europa.snu.ac.kr`

Abstract. In recent years, effective ranking strategies for relational databases have been extensively studied. Existing approaches have adopted empirical term-weighting strategies called *tf*×*idf* (term frequency times inverse document frequency) schemes from the field of information retrieval (IR) without careful consideration of relational model. This paper proposes a novel ranking scheme that exploits the statistical correlations, which represent the underlying semantics of the relational model. We extend Bayesian network models to provide dependence structure in relational databases. Furthermore, a limited assumption of value independence is defined to relax the unrealistic execution cost of the probabilistic model. Experimental results show that our model is competitive in terms of efficiency without losing the quality of query results.

Keywords: Ranking, Keyword search over structured data, Correlation, Attribute value, Bayesian networks.

1 Introduction

In recent years, there has been significant interest in developing effective retrieval models for structured data (e.g., relational databases) because a large portion of available data is structured and stored in database systems. Although relational database systems are designed to manage structured data effectively, they currently only support a simple Boolean retrieval model. Recent studies [6, 2, 1, 7, 4] have proposed many useful ranking functions for their own purposes. The most commonly used ranking schemes are the adaptation of *tf*×*idf* schemes from the area of IR. However, we note that *tf*×*idf* schemes often ignore the fundamental aspects of relational model. Many of them ignore the relationship between values in practice. Although they work well for several pragmatic cases such as in text databases, they are not appropriate for general relational databases. For example, repeating groups of terms hardly ever appear in a tuple because a tuple is a set of attribute values; it cannot contain multiple occurrences of an atomic value. For example of car sales databases, there is no car that has two car body styles at the same time. A car instance is represented as a set of atomic properties. Therefore, the frequency of a term (which is represented as a set of attribute values) in a tuple is not appropriate measure for significance of it. Therefore, the direct adaptation of *tf*×*idf* schemes in structured data cannot effectively work in general cases.

* This research was supported by Seoul R&BD Program (WR080951).

In this paper, we propose a novel ranking method by analyzing the rich dependency structures explicitly represented in relational databases. In an explicit feature space, we can easily find the relationship between values and use it more effectively. Rather than using the frequencies of a given term (as a query), we model the *correlations* of the term with other terms in the same tuple to estimate the significance of the term in the tuple. Although computing the dependency structured between terms may incur high cost, we can reduce the cost by assuming a limited independence on term sets selected by user's intention. The Bayesian network model is used to describe the correlations between terms with the assumption.

We evaluated and validated our ranking method through experimentation using real datasets extracted from commercial web databases. We compared our method with a state-of-the-art approach [4]. The experimental results show that our method is competitive in terms of efficiency without losing the quality of query results.

The rest of the paper is organized as follows: In Section 2, we review related work and compare it with our work. In Section 3, we describe a problem with a correlation-based weighting scheme and statistical measures for quantifying the attribute value correlations. In Section 4, we introduce a probabilistic ranking model based on attribute value correlation with a limited assumption of value independence. In Section 5, we present experimental results for evaluating our ranking method on real datasets. Finally, we present concluding remarks in Section 6.

2 Related Work

The problem of ranking database query results has been actively investigated in recent years. Several works such as [3, 9] have employed relevance feedback based on machine learning techniques. In [9], the authors proposed SQL extensions in which users can specify ranking functions via preference constraints. In [4], authors adopt the principles of the probabilistic model from the IR field, namely, the Probabilistic Information Retrieval (PIR) model, to rank database tuples. The aspect that distinguishes our work from the above is that we do not need any prior knowledge from users (e.g., query workloads or user feedback) for ranking query results.

Several existing works have supported keyword searches on relational databases, e.g., [2, 7, 12]. They utilize state-of-the-art IR ranking strategies such as *tf*×*idf* schemes or *PageRank* style schemes, which perform well in practice. Because they mainly focus on the strategies to join tuples for keyword matches across multiple tables, the size of the joined tuples is considered as an important measure of the ranking. Recent work, such as [12], has focused on modifying ranking functions from the approaches in IR and their related query processing methods. Top-k query algorithms are considered as an efficient online query processing technique for score aggregation. Fagin et al. [5] introduced a set of novel algorithms, assuming that sorted access and random access to the objects are available for each attribute. A number of improvements have been suggested by several studies, such as [8]. However, such approaches have primarily focused on the scores of an individual object based on its own attributes independently. While the dependency among attributes comprises a major part in many real-world datasets, ranking over correlated attribute value has not been properly supported by the classic algorithms for top-k queries.

The works most closely related to ours are found in [4]. In particular, Chaudhuri et al. [4] employed the inverse document frequency of attribute values to estimate odds of relevance, given queries in the context of querying relational databases. They exploit the query workload (user query logs) to discover term significance for a query. Although it is a usual technique in IR to exploit the knowledge of relevance at query time, it can produce potentially unintuitive results if the query workload is unavailable or is built up by unrelated terms in a query. On the other hand, we do not use any prior knowledge to extract user preferences, but rather we proactively analyze given database statistics. Our work is more intuitive when there is no proper information related to previous uses (such as query workloads); this context is similar to cold start problem in the fields of recommendation.

3 Attribute Value Correlation

3.1 Basic Concepts

In this section, we introduce the basic concepts for modeling the correlation in relational databases. We present two types of attribute values: specified attribute values and unspecified attribute values.

Definition 1. Let $K = \{k_1, k_2, ..., k_n\}$ be the *terms* of a collection R of tuples, that is, the set of n unique terms that appear in all tuples in R.

Definition 2. Let $V = \{v_1, v_2, ..., v_o\}$ be the *attribute values* of a collection R of tuples, that is, the set of o unique attribute values that appear in all tuples in R. We assume that a term k_i in K is arbitrary mapping to an attribute value v_i; there exists a mapping function of k_i to v_i in R.

Definition 3. A *query* Q, $Q \subseteq V$, is a set of l attribute values. $|Q|$ denotes the number of query terms. A *tuple* T, $T \subseteq V$, is a set of m attribute values. $|T|$ denotes the number of attribute. An attribute value av of a tuple t binding to an attribute a is denoted as $av=t[a]$. R is a set of tuples, $R=\{T_Q^1, T_Q^2, ..., T_Q^{|R|}\}$ and $Q \subseteq T_Q^i$, which contains all of the query terms.

Definition 4. Given query Q, let $SV_i = \{sv_{i1}, sv_{i2}, ..., sv_{i|Q|}\}$, $SV_i \subseteq T_Q^i$, be the set of unique attribute values that appear in the query Q and the tuples T_Q^i. We define this set as the *specified attribute values*. Let $UV_i = \{uv_{i1}, uv_{i2}, ..., uv_{i|T|-|Q|}\}$, $UV_i \subseteq T_Q^i$, be the set of unique attribute values that appear in the tuples T_Q^i but not in the query Q. We define this set as the *unspecified attribute values*. Two sets, $SV_i \cap UV_i = \emptyset$, are disjoint, and the union of SV_i and UV_i forms a tuple T_Q^i.

3.2 Illustrative Examples

To quantify the usefulness of terms in characterizing the semantics of the tuple in which they appear, each attribute value matching the terms in a tuple is assigned a weight based on its significance for the tuple, given a query. We suggest using the *correlations* between the specified attribute values and the unspecified attributes to derive the weight of the attribute values specified by the queries. A key feature of our

Table 1. A part of a table with correlated values

	WD	Brakes	Doors	Color
t1	AWD	Power Brakes	3 Doors	Silver
t2	AWD	Power Brakes	3 Doors	Black
t3	AWD	Power Brakes	3 Doors	Black
t4	AWD	Power Brakes	3 Doors	Black
t5	AWD	Power Brakes	3 Doors	Green
t6	AWD	Power Brakes	2 Doors	Grey

approach is that we can easily extract the correlations from the statistics of relational databases.

The basic intuition in this example is that semantically related terms often occur close to each other. Suppose that we have a set of tuples and wish to determine the weight of an attribute value for a tuple. Let us consider Table 1 as the Boolean query results for a query term "AWD" for a car sales database. The value *AWD* is considered as a specified attribute value for a car instance *ti*. We estimate the weight of the specified attribute value based on the correlations with unspecified attribute values. Table 1 illustrates an extreme case of statistical closeness, such as the correlations between *AWD* and *Power Brakes*. These values are heavily correlated and can be exploited to weight the specified attribute value *AWD*: the weight of the attribute value *AWD* is estimated as high for a tuple that contains *Power Brakes*. The weight of a specified attribute value for a given tuple is increased by the correlations of unspecified attribute values. To compute the correlation, we naively count the co-occurrences of common tuples to estimate correlations. The correlations of the unspecified attribute values *Power Brakes*, *3 Doors* and *2 Doors* are (approximately) proportional to *6*, *5* and *1*, respectively. Based on this intuition, the weight of AWD for t_1~t_5 is estimated higher than that for t_6.

3.3 Measures of the Correlations

Originally, correlations represent statistical relationships between random variables or observed data values. We consider relational databases as collections of random variables at a specific point in a search space.

Definition 5. Let V be a discrete random variable on a finite set $V = \{v_1, v_2, ..., v_n\}$, with probability distribution function $p(v_i) = Pr(V=v_i)$. If the observations of $V=v_i$ are dependent to the observations of $V=v_j$, then we say that the two attribute values v_i and v_j, have a correlation. Although correlation does not imply causation, we can establish a causal relationship in the correlation by considering the prior condition, the observation of the query Q. $Pr(V=v_j|Q)$ describes the extent of the correlation of the unspecified attribute value v_j with the specified attribute value v_i given by the query Q.

To test the correlations (or dependencies) between categorical values, theoretical frequencies are calculated using a distribution from values binding an attribute of a column. We measure all possible outcomes pointwise from the expected value of the covariance. The function of the correlation test for the attribute values x and y is:

$$C(x,y) = \frac{p(x,y) - p(x)p(y)}{\sqrt{p(x)\big(1-p(x)\big)p(y)(1-p(y))}}. \tag{1}$$

4 Probabilistic Ranking Model

4.1 Bayesian Network Model for Probabilistic Ranking

Bayesian networks provide a graphical formalism (a directed acyclic graph) for explicitly representing dependencies among the variables of a domain, thus providing a concise specification of a joint probability distribution. In our model, the dependencies can be interpreted as correlations, given a specific query. Let *P* be a joint probability distribution defined over the sample space *U*. As in [11], the probability *P*(*c*), associated with a generic concept *c* (which represents a tuple or a query by a subset of *U*) in the sample space *U*, is defined by the following formula:

$$P(c) = \sum_{u \in U} P(c|u)P(u). \tag{2}$$

The ranking computation is based on interpreting the relevance between a tuple *t* and the query *q* as the intersection between the concepts *t* and *q*. To quantify the degree of relevance of tuple *t* given the query *q*, we use the probability *P*(*t*|*q*), the probability of relevance, as follows (for details, refer to [11]):

$$P(t|q) = \frac{1}{P(q)} \sum_{u \in U} P(t|u)P(q|u)P(u). \tag{3}$$

4.2 Problem Formulation

Consider a database table *R* with tuples $T = \{t_1, t_2, \ldots, t_{|R|}\}$ with attributes $A = \{a_1, a_2, \ldots, a_m\}$ and a query $Q = \{k_1, k_2, \ldots, k_{|Q|}\}$ over *R* with a parameterized query of the form "SELECT * FROM R WHERE $a_1 = k_1$ AND $a_2 = k_2$ … AND $a_{|Q|} = k_{|Q|}$", where each a_i is the attribute from *A* corresponding to the specified attribute value k_i specified by the query terms ($|Q|<m$). The set of attributes $A_s = \{sa_1, sa_2, \ldots, sa_{|Q|}\} \subseteq A$ is called the set of specified attributes by the query terms. Analogously, the set of attributes $A_u = \{ua_1, ua_2, \ldots, ua_{m-|Q|}\} \subseteq A$ is called the set of unspecified attributes. We assume that the set of attributes A_s can be simply specified by a structured query or previous work on candidate networks as in [6, 7] (topics related to querying on multiple tables). To compute the score, we mine the attribute value correlations *C*(*sv*,*uv*) for the given query, which is an specified attribute value $sv=t[sa_i]$ ($uv=t[ua_i]$ denotes an unspecified attribute value). Now a simple scoring function can be defined by the following formula:

$$S(Q,t) = \prod_{\forall sa_i \in A_s, \forall ua_i \in A_u} C(t[sa_i], t[ua_i]). \tag{4}$$

4.3 Extending the Bayesian Network Model with the Limited Assumption of Value Dependency

Figure 1 illustrates the extended Bayesian network model for our purposes. A database is represented by the set of the tuples t_i. The nodes sv_i and uv_j represent two types of attribute values: the specified attribute values and the unspecified attribute values. Each node k_i represents a term in the sample space U. The universe of discourse U, which we take as our sample space, is a set of elementary terms u in the space U. The node q represents the query that corresponds to terms in u.

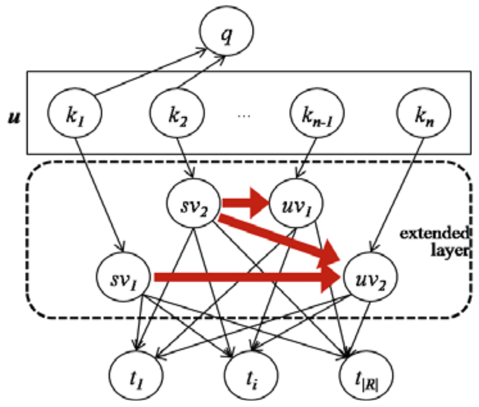


Fig. 1. Correlation modeling in the Bayesian network

First of all, we define an extended layer for attribute values to allow dependencies between them. The specification of the probability $P(t|q)$ needs to consider the dependency. Traditionally, most practical retrieval models assume the following: given a query Q and a tuple t, dependencies between the terms are not allowed. Although this assumption is unrealistic in many cases, it reduces computational costs. For example, given the sample space U, we can have 2^t concepts (subsets of U). If all concepts are considered to be equally likely a priori, each prior probability $P(u)$ is set to $P(u)=(1/2)^t$, which is a constant. Without this assumption, the dependencies among elementary nodes have to be specified by expanding $P(u)$ to $P(u|u')P(u'|u'')$ … . Our preliminary implementations suggested that such approaches have unacceptable preprocessing and query processing costs to calculate the probabilities. Consequently, we define a limited independence assumption only on the extended layer.

Limited Value Independence Assumption. Given a query q and a tuple t, the specified attribute values are assumed to be mutually independent. Analogously, the unspecified attribute values are assumed to be mutually independent. We allow dependencies between a specified attribute value and an unspecified attribute value. Therefore, $p(sv,uv)$ is not equal to $p(sv)p(uv)$ if there exists a correlation between them.

By allowing dependencies (thick arrows in Figure 1) in only a limited set of nodes, we can avoid the high cost of calculating the prior probabilities. Still, the dependencies approximate the probability of relevance that is proved to be significant for the quality of ranking. Based on *Definition 5*, the probability of the relevance between the query Q and the tuple t_i can be derived as follows:

$$\begin{aligned} P(t_i|q) &= \frac{1}{P(q)} \sum_{u \in U} P(t_i|u)P(q|u)P(u) \\ &= \frac{1}{P(q)} \sum_{u \in U} P(SV_i, UV_i|u)P(q|u)P(u) \\ &= \frac{1}{P(q)} \sum_{u \in U} P(SV_i|u)P(UV_i|SV_i, u)P(q|u)P(u). \end{aligned} \tag{5}$$

Based on *Definition 2*, edges from the term k_i only exist if k_i is observed for a given query and given attribute values. This implies that the term k_i is conditionally independent, given the query q. Therefore, u can be divided into two disjoint concepts u_q and u_x=u-u_q corresponding to the attribute values sv and uv, respectively. The conditionally independent concepts u_x (and u_q) for corresponding sv_i (and uv_i) is removed because the value is reducible. Additionally, $P(q)$ and $P(u)$ can be considered as constants. The final ranking function is as follows:

$$P(t_i|q) \propto \sum_{u \in U} P(SV_i|u_q)P(UV_i|SV_i, u_x)P(q|u_q). \tag{6}$$

Therefore, to infer the probability of a tuple t_i given a query q, we need to specify the following probabilities: $P(SV_i|u_q)$, $P(UV_i|SV_i,u_x)$ and $P(q|u_q)$. The specifications of the probabilities determine the final ranking function for our method.

$$P(SV_i|u_q) = \prod_{\forall sv_{il} \in SV_i} P(sv_{il}|u_q). \tag{7}$$

$$P(UV_i|SV_i, u_x) = \prod_{\forall sv_{il} \in SV_i, \forall uv_{iw} \in UV_i} C(sv_{il}, uv_{iw}). \tag{8}$$

$$P(q|u_q) = \begin{cases} 1 & if\ \forall k_i, \quad I_q(k_i) = I_{u_q}(k_i) \\ 0 & otherwise \end{cases}. \tag{9}$$

$P(SV_i|u_q)$ is specified based on the mutual independence among the specified attribute values. In our experiment, we set this probability to unity, which corresponds to Boolean matching. $P(UV_i|SV_i,u_x)$ is defined as the aggregation of the attribute value correlations C. We use $P(q|uq)$ to select the concept u_q that matches the query. The function $I_q(k)$ is 1 if $k \in q$ and 0 otherwise. This equation guarantees that the states where the active values u_q of the query q are taken into consideration. By applying Equations 7, 8 and 9 to Equation 6, we obtain a ranking equivalent to the one found with Equation 4. The score for tuple t, given the query Q, is defined by the probability $p(t|q)$. To define the probability of observing the unspecified values of the tuples t_i, we use the correlation measure, as described in Section 3.

5 Experimental Evaluation

5.1 Experimental Setup

In our evaluation, we use a used car database *CarDB* (*Make, Model, Year, Color, Style, Price, Mileage, Location*) containing 158351 tuples extracted from a web database, *Yahoo! Autos*[1]. This database is used in the related works such as [4, 10]. In this database, each tuple represents a car for sale, and each column represents a set of car attributes. Because we focus on categorical data, the numerical attributes (e.g., *Price* and *Mileage*) are discretized into five buckets. A MySQL Server 5.0 RDBMS is used on an AMD Athlon 64 processor 3.2 GHz PC with 2GB of RAM for the environment of the experiment. All of the algorithms are implemented in JAVA, connected to the RDBMS through JDBC. Two other ranking methods (RANDOM, PIR[4]) are implemented for comparison with our method (CORR-#).

5.2 Retrieval Effectiveness

While IR relies on extensive user studies and available benchmarks (such as the TREC collection), such infrastructure is not currently available for evaluating database rankings [1]. Nonetheless, we conducted user studies (similar to [1, 4, 12]) to evaluate the search quality using limited resources in a nascent stage of research. We requested 14 subjects who are graduate students from our respective universities and institutions to behave like used-car buyers. We solicited 10 test queries that represent a heterogeneous mix of different profiles of positional car buyers as we did not wish our queries to be biased. Because it is not practical to ask participants to select all of the query results for a specific query, we conducted a comparative study for the top-10 results for average precision (cut-off: 10). To compare the performance of the different ranking methods for the top-10 query results, first, top 10 tuples are collected from each ranking method, obtaining a total of 30 tuples. If there is overlap among the tuples resulting from the different methods, more tuples are extracted using the RANDOM method so that 30 unique tuples are collected in total. Next, for each of the 10 queries, each participant was asked to list the top-10 tuples in the order of preference.

To evaluate the Kendall tau distance for the top-k results, we asked the participants to order only the top-30 tuples for each query to evaluate the rank-specific effectiveness, because requiring users to rank all of the results (over hundreds of tuples) for each query would be tedious. In varying the number of k, $1 \leq k \leq 30$, the Kendall tau distance is measured by comparing the counts of swaps required to place the list of one ranking method in the same order as the answer list ordered by the candidates. In this experiment, a primitive correlation measure, *co-occurrence* (CORR-2), is considered as a baseline because the RANDOM method yields unreasonable effectiveness in the top-k lists. The Pearson product-moment correlation coefficient-based method is denoted by CORR-1.

Figure 2 shows the average precision of the different ranking methods for each query. The evaluation results show that CORR generally produces rankings of higher

[1] `http://autos.yahoo.com`

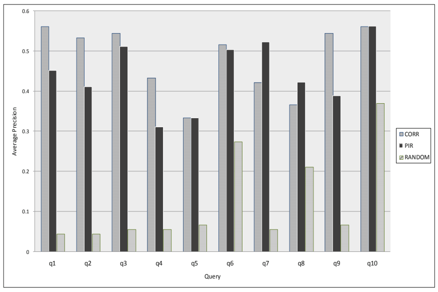


Fig. 2. Precision at 10

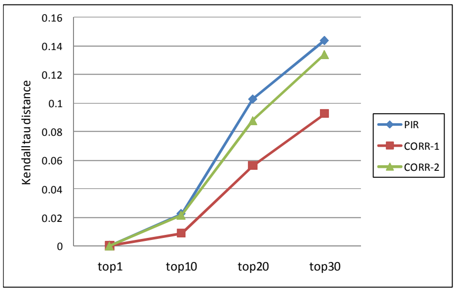


Fig. 3. Kendall tau distance

quality compared to the RANDOM method for every query. For most of the queries, there are some overlaps between the results of CORR and that of PIR. The precision of CORR is 0.1 higher than that of the PIR method on average. However, the query results for q7 and q8 have lower in precision compared to the PIR method. The reason for this phenomenon was that some terms that were very commonly surfaced in the query workloads, but these terms were not popular terms for car sales databases. Because our method only focuses on the given database statistics, the vocabulary not appearing in the database cannot be used as evidences. For eight of ten queries, the precision of CORR is 0.15 higher than that of the PIR method on average. Figure 3 shows the Kendall tau distance of the different ranking methods for each top-*k* query. It shows that CORR-1 generates the most relevant tuple ranking for any *k* threshold. We emphasize that even the primitive correlation metrics-based method (e.g., co-occurrences) performs better (or almost equal to) than the *tf*×*idf* scheme-based method, PIR. While the results of these user studies appear promising, we caution that it would be premature to interpret the results as conclusive evidence at all cases. From the above results, we find that our basic premise, that the statistical relationships between attribute values are effective for ranking tuples, is reasonable.

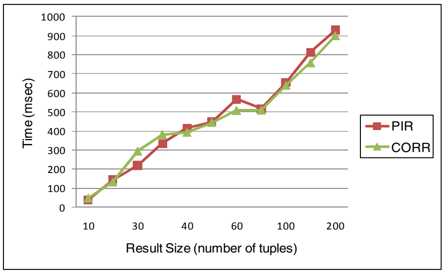


Fig. 4. Execution time

5.3 Computational Efficiency

In this section, we show how the performance of our methods changes according to the size of the data processed. We measure the execution time while varying the number of tuples in the result set. Figure 4 shows the online execution time as a function of the number of tuples in the query results. The online query processing includes the attribute value weight selection and aggregation for the tuple scores. The execution time grows almost linearly with the number of tuples in the query results because the number of categorical attributes and the query size are relatively smaller than the number of tuples (l, m<<n). Our method is competitive compared to related work in terms of efficient top-*k* computation in random access-based approaches.

6 Conclusion

In this paper, we introduce a probabilistic method to enable an effective retrieval over relational databases with categorical attributes by analyzing the attribute value correlation. Our ranking function is based on the Bayesian network model, explicitly introducing the limited independence assumption on the set of attribute values to avoid high computational costs. We present a set of results from experiments conducted on a real dataset. Our experiments show that our ranking strategy is effective without a priori knowledge and provides a reasonable efficiency.

Further research is required to develop the efficient top-*k* computation for multiple relational data structure.

References

1. Agrawal, S., Chaudhuri, S., Das, G., Gionis, A.: Automated Ranking of Database Query Results. In: CIDR, pp. 262–268. ACM Press, New York (2003)
2. Hulgeri, A., Nakhe, C.: Keyword Searching and Browsing in Databases using BANKS. In: ICDE, pp. 431–440. IEEE Press, Washington DC (2002)
3. Chakrabarti, K., Porkaew, K., Mehrotra, S.: Efficient Query Refinement in Multimedia Databases. In: ICDE, pp. 196–204. IEEE Press, Washington DC (2000)
4. Chaudhuri, S., Das, G., Hristidis, V., Weikum, G.: Probabilistic ranking of database query results. In: VLDB, pp. 888–899, VLDB Endowment (2000)
5. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: PODS, pp. 102–113. ACM Press, New York (2001)
6. Hristidis, V., Papakonstantinou, Y.: Discover: keyword search in relational databases. In: VLDB, pp. 670–681, VLDB Endowment (2002)
7. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient IR-style keyword search over relational databases. In: VLDB, pp. 850–861, VLDB Endowment (2003)
8. Ilyas, F., Aref, G., Elmagarmid, K.: Supporting top-*k join* queries in relational databases. The VLDB J. 13(3), 207–221 (2004)
9. Ortega-Binderberger, M., Chakrabarti, K., Mehrotra, S.: An Approach to Integrating Query Refinement in SQL. In: EDBT, pp. 15–33. ACM Press, New York (2002)
10. Nambiar, U., Kambhampati, S.: Supporting queries with imprecise constraints. In: AAAI, pp. 1654–1657. AAAI Press, New York (2006)
11. Ribeiro, B.A., Muntz, R.: A belief network model for IR. In: SIGIR, pp. 253–260. ACM Press, New York (1996)
12. Meng, X., Ma, Z.M., Yan, L.: Answering approximate queries over autonomous web databases. In: WWW, pp. 1021–1030. ACM Press, New York (2009)

LinearDB: A Relational Approach to Make Data Warehouse Scale Like MapReduce*

Huiju Wang[1,2], Xiongpai Qin[1,2], Yansong Zhang[1,2], Shan Wang[1,2], and Zhanwei Wang[1,2]

[1] Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University of China), MOE, Beijing 100872, P.R. China
[2] School of Information, Renmin University of China, Beijing 100872, P.R. China
wanghuiju.cn@hotmail.com, qxp1990@sina.com, {zhangys_ruc,swang}@ruc.edu.cn, mayjojo@yeah.net

Abstract. Operating on computer clusters, parallel databases enjoy enhanced performance. However, the scalability of a parallel database is limited by a number of factors. Although MapReduce-based systems are highly scalable, their performance is not satisfactory for data intensive applications. In this paper, we explore the feasibility of building a data warehouse that incorporates the best features from both technologies – the efficiency of parallel database and the scalability and fault tolerance of MapReduce. Towards this target, we design a prototype system called LinearDB. LinearDB organizes data in a decomposed snowflake schema and adopts three operations – transform, reduce and merge – to accomplish query processing. All these techniques are specially designed for the cluster environment. Our experimental results show that its scalability matches MapReduce and its performance is up to 3 times as good as that of PostgreSQL.

Keywords: hierarchical encoding, data warehouse, scalability, star join.

1 Introduction

The scale of data warehouse continues exploding [4]. More and more enterprises have started building their "private clouds", shifting away from high-end machines, and moving to computer clusters composed of low-cost machines.

Parallel database does not adapt to the computer cluster architecture. Most parallel databases are designed for the platforms consisting of no more than several hundreds of high-end servers, where queries take no more than a few hours to finish. Failures are relatively rare in such an environment. Once a failure happens, a parallel

* This work is partly supported by the Important National Science & Technology Specific Projects of China ("HGJ" Projects, Grant No.2010ZX01042-001-002), the National Natural Science Foundation of China (Grant No.61070054), the Fundamental Research Funds for the Central Universities (the Research Funds of Renmin University of China, Grant No.10XNI018), the Renmin University of China (Grant No.10XNB053), and the Graduate Science Foundation of Renmin University of China (Grant No.10XNH096 and No.11XNH120).

database can simply re-execute the query. In contrast, the machines in a computer cluster tend to be cheaper, less reliable, and more numerous. Their failures are much more common. When the system scales to a large number of nodes in a cluster, the cost of system failures will increase quickly – a single node's failure will cause the whole query to be redone.

Star schema and snowflake schema are two of the most frequently used data models in data warehouse [7]. Traditionally, they are implemented on RDBMS, which heavily depends on join operations to handle analytical queries. This solution prevents data warehouse from scaling across a cluster. For example:

1. If we distribute both the dimension tables and the fact table evenly over the computing nodes, we will introduce excessive network transmission to the cluster, because star (snowflake) queries requires a significant number of join operations to associate the dimension tables with the fact table;

2. If we copy all the dimension tables to each node and distribute only the fact table, the storage space consumption may become unaffordable. For instance, to distribute 5GB dimension tables and 1TB fact table over one hundred nodes, the dimension tables will take 5GB*100=500GB, which is almost half of the fact table. If there are more nodes, more additional space will be consumed.

MapReduce-based system proves to be inefficient. MapReduce-based system is superior to RDBMS in scalability and fault tolerance, but inferior to RDBMS in performance [12]. In particular, MapReduce-based systems are very inefficient at join operations, let alone star (snowflake) joins.

It is possible to make data warehouse scale as MapReduce. Our aim is to divide a query processing task into multiple sub-tasks, such that each sub-task can be distributed to a data node that works independently without communicating with the other nodes. Then, the work of a failed node can be done by its backup node, without the necessity of redoing the whole query. However, star (snowflake) schemas require joins to correlate their fact tables and dimension tables, making it difficult to divide the process into independent tasks. To handle this issue, we adopt a denormalization method – putting dimension hierarchy information into fact tables. In this way, most queries can be executed on the fact table without accessing the dimension tables. We achieve such de-normalization through hierarchy encoding [1, 3, 14], as the vast majority of predicates of data warehouse queries operate on hierarchies (such as the queries on MOLAP cubes).

Based on the de-normalized schema, we design a query processing model called TRM. It consists of three operators: 1) **T**ransform: probe dimension tables and translate predicates on dimension tables into predicates on fact tables; 2) **R**educe: scan, aggregate and sort fact tables in a distributed manner; 3) **M**erge: merge all results generated by the data nodes, perform residuary joins and ordering, and finally output the results. The three operations can be executed independently. Most importantly, our approach liberates the fact table from the dimension tables, so that we can distribute the fact table across a large cluster and scan it in parallel. Network transmission is also minimized in our TRM execution model.

The main contributions of this work include:

- We re-evaluate the traditional star (snowflake) schema in the context of the cluster environment, and propose a scalable data warehouse schema – decomposed snowflake schema for distributed data-warehouse query processing. The improved schema eliminates the reliance of the fact table on the dimension tables, such that the segments of the fact table can be stored and processed independently. It also transforms the expensive star (snowflake) join operation into a series of record matching based on the dimensional hierarchy encoding and a scan of the fact table.
- We present a complete query processing model – TRM execution model, which answers all data warehouse-style queries in three operations: Transform, Reduce and Merge. To handle the optimization challenges posed by the new execution model, we propose a number of optimization techniques, such as multi-column scan, data merging, etc.
- We have performed extensive experiments to evaluate the scalability and performance of our proposal.

The rest of the paper is organized as follows. Section 2 gives an overview of the related work. In Section 3, we describe the data organization in LinearDB. Section 4 presents the Transform–Scan–Reduce execution model, and Section 5 deals with its implementation and optimization issues. Our experiment results are presented in Section 6. Finally, Section 7 summarizes our approach and discusses directions of our future work.

2 Related Work

Large scaled data analysis has proliferated recently with the introduction of the Map- Reduce model [8] (represented by the open-source Hadoop [5]), and its related soft-ware, such as Hive [9], Pig Latin [13], HBase [15], etc. However, MapReduce is desi- gnnnned initially for unstructured data processing based on files. The applications are mainly focused on language and interface issues. There have been some proposals to bridge the MapReduce-based systems and parallel DBMSs, such as HadoopDB [10], Greenplum and Aster Data. HadoopDB is a hybrid of MapReduce and DBMS techno- logies. It uses MapReduce as the communication layer and conduct data processing within DBMS. Greenplum and Aster Data provide the ability of writing MapReduce jobs over data stored in the databases. These systems are important steps towards the direction of making DBMS scalable [17]. Different from all of them, we stem from the relational theory. In other words, we store all data as relations and process queries based on relations.

Processing and optimizing star join queries based on hierarchically encoded star schema are first introduced in [1, 3]. In [1, 3], surrogate keys based on the dimension hierarchies are used to link dimension tables and fact tables. In their approaches, a fact table is hierarchically clustered and star joins are transformed to multi- dimensional range queries based on UB-Tree [16]. In [14], the authors proposed a CSB star schema based on composite surrogate key, which is similar to our dimension surrogate key. We use hierarchical encoding too, but more thoroughly: first, each hierarchy is viewed as a dimension and associated with a hierarchy surrogate key; second, the original dimension table's key are replaced by a path-based dimension key; third, all foreign keys in the fact table are replaced by a code which contains all the dimensions' hierarchical information; finally, each query is processed by a scan on fact table, instead of a range query using an index.

Processing star queries based on table scans is a new trend for data warehouse. Blink [2] pre-joins data, transforms star schema into a universal relation, and then encodes the relation into a bit stream. It processes all queries using table scans. This method eliminates expensive joins, but introduces heavy workload to preprocessing and high cost to storage space. Our approach processes queries by table scans and a small number of residuary joins. Our system organizes data in a more efficient way – the information we store in the fact table is just the surrogate code of the dimension hierarchy, rather than the complete dimension table. Our batch predicate evaluation is similar to the parallel predicate evaluation in Blink. The difference is in the implementation, that is, we do not distinguish between odd-numbered fields or even-numbered fields.

3 Data Organization

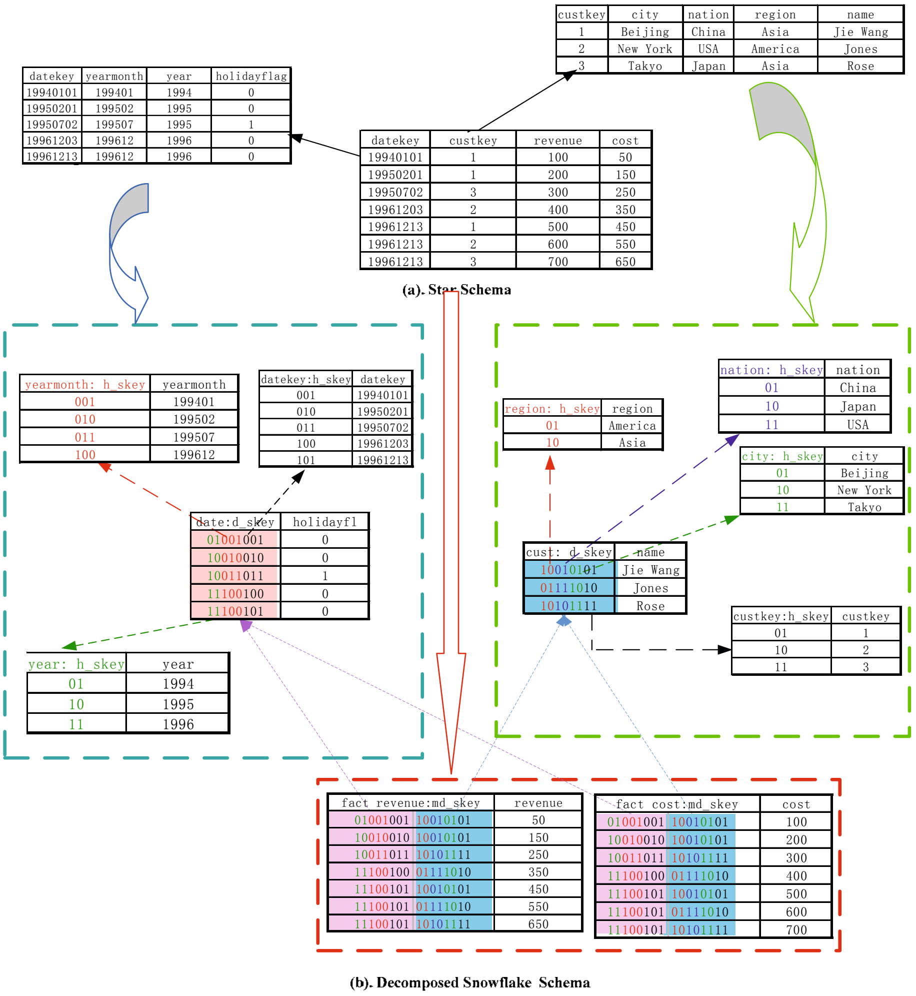
The query processing in LinearDB is centered around the decomposed snowflake schema, an evolution of snowflake schema based on hierarchical encoding. In this section, we discuss the key idea of LinearDB and its deployment strategy.

3.1 Decomposed Snowflake Schema

To eliminate the star (snowflake) joins, we modified a star (snowflake) schema into a decomposed snowflake schema. This solves the scalability issue from the schema level. The new schema puts the dimension hierarchy information into the fact table. To achieve this, we use a code to represent the hierarchies of each dimension, and replace the foreign keys in the fact table with the codes. Our encoding method is a little more complex than that of the existing work:

Hierarchy Surrogate. A dimension table consists of several hierarchical attributes and other descriptive information. Each hierarchy has a value domain, whose values can be ordered. Hierarchical encoding uses a code to represent a value of each hierarchy level. Let L be a hierarchy level of a dimension with C values, $<$ be the "less than" comparison operator, and *member (L)* be the possible values of L. We can define a one-to-one function S: member (L) $\rightarrow$ [0, C-1], such that, for every $u, u' \in$ member (L), $u < u'$ implies $S(u) < S(u')$. Then, $S(u)$ is called the surrogate of u. Note that if $<$ is not defined for the domain of L, S is simply defined as an arbitrary one-to-one function from the value domain to [0, C-1]. In this work, we express surrogate as a bit string. Then, at least [$\log_2 C$] bits are reserved in the binary representation of the surrogate of the level L. We denote the surrogate of a hierarchy as *h_skey*.

Dimension Surrogate. As suggested previously, each dimension of a snowflake schema corresponds to a hierarchy tree. The dimension surrogate of a tuple v, denoted by *d_skey (v)*, is the path of v in the hierarchy tree, where the concatenated values are replaced by their *h_skey* keys. If there are two or more paths for a level in the hierarchy tree, a *d_skey* is produced for each path, and then these *d_skey* keys *are*



datekey	yearmonth	year	holidayflag
19940101	199401	1994	0
19950201	199502	1995	0
19950702	199507	1995	1
19961203	199612	1996	0
19961213	199612	1996	0

custkey	city	nation	region	name
1	Beijing	China	Asia	Jie Wang
2	New York	USA	America	Jones
3	Takyo	Japan	Asia	Rose

datekey	custkey	revenue	cost
19940101	1	100	50
19950201	1	200	150
19950702	3	300	250
19961203	2	400	350
19961213	1	500	450
19961213	2	600	550
19961213	3	700	650

yearmonth: h_skey	yearmonth
001	199401
010	199502
011	199507
100	199612

datekey:h_skey	datekey
001	19940101
010	19950201
011	19950702
100	19961203
101	19961213

date:d_skey	holidayfl
01001001	0
10010010	0
10011011	1
11100100	0
11100101	0

year: h_skey	year
01	1994
10	1995
11	1996

region: h_skey	region
01	America
10	Asia

nation: h_skey	nation
01	China
10	Japan
11	USA

city: h_skey	city
01	Beijing
10	New York
11	Takyo

cust: d_skey	name
10010101	Jie Wang
01111010	Jones
10101111	Rose

custkey:h_skey	custkey
01	1
10	2
11	3

fact revenue:md_skey		revenue
01001001	10010101	50
10010010	10010101	150
10011011	10101111	250
11100100	01111010	350
11100101	10010101	450
11100101	01111010	550
11100101	10101111	650

fact cost:md_skey		cost
01001001	10010101	100
10010010	10010101	200
10011011	10101111	300
11100100	01111010	400
11100101	10010101	500
11100101	01111010	600
11100101	10101111	700

Fig. 1. A Star Schema and its transformational Decomposed Snowflake Schema. Each hierarchy is transformed to a dimension table with an *h_skey*, and the original dimension table is transformed to a new one with a *d_skey*. The two tables at the bottom correspond to the original fact table.

concatenated to form a new *d_skey*. As a result, each *d_skey* contains all hierarchical information of the corresponding dimension.

Examples for *h_skey* and *d_skey* are given in Figure 1(b). Based on the hierarchy surrogate and the dimension surrogate, we propose the decomposed snowflake schema, whose structures are described as follows.

The Dimension table. Every dimension is normalized and encoded using the hierarchical encoding. Each hierarchy level of a dimension corresponds to a dimension table:

1. If the hierarchy level is the lowest level, its dimension table consists of:
   - Dimension surrogate key: *d_skey*;
   - Fields of the original dimension table that depend on this hierarchy attribute.
2. If the hierarchy level is not the lowest level, its dimension table consists of:
   - Hierarchy surrogate key: *h_skey*;
   - Fields of the original dimension table that depend on this hierarchy attribute.

The Fact table. To put the dimensional hierarchical information into the fact table, we use a surrogate attribute to represent the hierarchical information of every dimension in the fact table. This is achieved by replacing each foreign key of the fact table with its corresponding *d_skey*, and concatenating all *d_skey* keys to form a new surrogate key, called *md_skey*. Let *t* be a tuple, and $d\_skey_i(t)$ be a dimension surrogate of the *ith* dimension of *t*. Then, *md_skey(t)* is the multi-dimensional surrogate key of the tuple *t*. In other words, $md\_skey(t) = d\_skey_1(t).d\_skey_2(t) \ldots d\_skey_k(t)$, where *k* is the dimension number of the fact table. To achieve good performance, we can drop the former foreign keys from the fact table.

A typical analytical query usually needs to inspect a large number of tuples of the fact table, but only a small subset of columns. For instance, in the SSB benchmark, more than half of the queries only need to access one measure. To improve I/O performance, we partition the fact table vertically by column. Each decomposed fact table has only two columns:

- Multidimension surrogate key: *md_skey* key;
- The fields of a non-hierarchy field of the original fact table.

In a running system, all surrogates are system assigned and maintained, and typically are made transparent to the user. A comparison between our decomposed snowflake schema and the star schema is shown in figure 1.

3.2 Data Distribution Strategy

As described above, in a decomposed snowflake schema, the fact table and the dimension tables can be processed independently. Thus, we can design data distribution strategies for the fact table and the dimension tables separately.

Our data distribution strategy is based on the following considerations: 1) The dimension tables are very small (normally less than 10GB); 2) The capacity of today's main memory is very large (some high-end servers can provide 1-2TB of memory space), and can hold all dimension tables' data; 3) The operations on the fact table are most time-consuming. Thus, we adopt a master-slave architecture. We store the dimension tables on the master node to simplify the management and processing of data, and distribute the fact table across data nodes to achieve good scalability. All the other metadata, such as the information of tables and data nodes and the data backup information, are stored on the master node.

To populate data into a decomposed snowflake schema, we partition the fact table in two directions. Firstly, we partition the fact table horizontally across data nodes in a *Round-Robin* fashion. Secondly, we repartition the data on each data node vertically according to the decomposed snowflake schema. For fault tolerance, we backup each node's data on one of the other nodes.

4 The Transform – Reduce – Merge Execution Model

The core of LinearDB's query processing consists of three operations: transform, reduce and merge. In this section, we show how to process queries using these operations.

4.1 An Overview of TRM Execution Model

Our approach handles all analytical queries in a unified model, called TRM (Transform–Reduce–Merge), which is illustrated in Figure 2. A query is first transformed by the transformer module into a reduce operation (filter, aggregation, sort) on the local decomposed fact tables. Then the reduce operation is passed to a coordinator which coordinates the related data nodes in the cluster to execute the reduce operation in parallel. After finishing the reduce operation, the data nodes transfer the locally reduced data back to the coordinator, which invokes the merger to merge the data and produces the final results. Following the idea of "Moving Computation is Cheaper than Moving Data", we scatter reduce operations across data nodes when processing queries. And other modules, including transformer module, coordinator module and merge module are deployed on the master node.

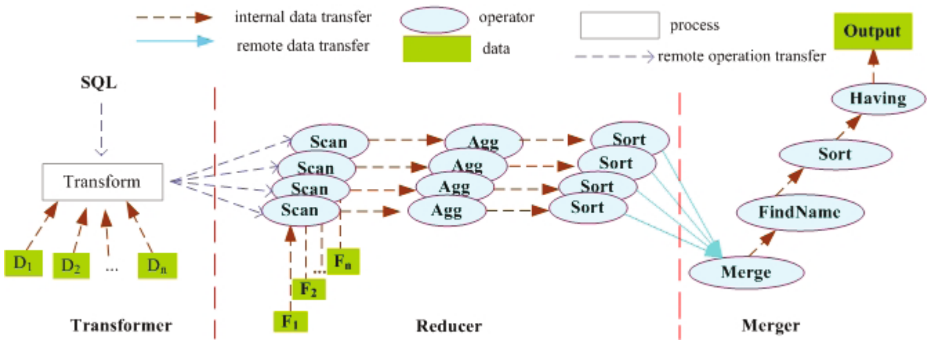


Fig. 2. An Overview of TRM execution model. A SQL is transformed first, and the produced reducer is scattered to each data node which scans fact table, and aggregates, sorts data in a pipeline, lastly data converge at master node and are merged.

As we can see from Figure 2, TRM is a single-direction execution flow, and the communication among the nodes occur only when the coordinator scatters the reduce operations to the data nodes or each data node transfers the result data to the master node. All the data nodes work in parallel without communicating with one another. Thus, the network transmission load is minimized.

4.2 Transformer

In the first stage of the TRM execution model, a SQL is transformed to a reduce operation. The transformer extracts predicates on the dimension tables from the SQL query and translates these predicates into predicates on the *md_skey* by checking the hierarchical encoding functions. The predicates on the fact table remain unchangeable.

We restrict ourselves to those queries whose selection predicates can be expressed as a conjunction of predicates in the form *attrib op const*, where *attrib* is an attribute of the relation, *op* is a comparison operator, and *const* is a constant value. Queries containing disjunctions can always be expressed as a union of such queries. Thus, this is not a limitation. Our transformation rules are as follows:

A. Predicates on Hierarchy

Equation conjunctive predicate transformation. Predicates in an equation conjunctive are all equation predicates. To transform this type of predicates, we create two bit strings. One is a mask, *extract_mask_equal*, for exacting the required hierarchies by a bitwise AND operation on the *md_skey*. The other is the expected result which contains corresponding hierarchies' *h_skey* at an appropriate offset. We denote this bit string by *md_skey_e*. As a result, each conjunctive becomes a pair of *extract_mask_equal* and *md_skey_e*. Then, for each tuple *t*, we evaluate the equation *extract_mask_equal & t = md_skey_e*, which only needs one comparison to evaluate all the predicates.

Range conjunctive predicates transformation. Range conjunctive predicate transformation is more complex, because there may be both range predicates and equation predicates in the conjunctive. For this type of transformation, we produce a new conjunctive predicate containing two predicates. One is an equation predicate which contains all the equation predicates in the conjunctive. We get the equation predicate in the same way as the equation conjunctive transformation described above. The other is a range predicate which evaluates all range predicates in this conjunctive. We transform all range predicates in a conjunctive into two bit strings that represent the expected bound for all predicates to be true. One bit string represents the lower bound (denoted by *md_skey_l*), and the other the upper bound (denoted by *md_skey_u*). The *md_skey_l* is combined with the *h_skey* of all the hierarchies' lower bound in the conjunctive. If a hierarchy has no lower bound, its corresponding bits in the *md_skey* are set to false (representing the minimal value). In the same way, the upper bound *md_skey_u* are combined with the *h_skey* of all the hierarchies' upper bound in the conjunctive. If a hierarchy has no upper bound, its corresponding bits in the *md_skey* are set to true (representing the maximum value). Bits that do not correspond to a range predicates are set to false in both *md_skey_l* and *md_skey_u*. We also create two masks – 1) *extract_mask_range*, for extracting the appropriate surrogates for the range predicate from *md_skey*; 2) *group_mask*, for extracting the code of the hierarchies which appear in the group-by clause for aggregation. When evaluating a tuple *t*, we test such a conjunctive: *extract_mask_equal & t = md_skey_e and* (*extract_mask_range & t*) *between md_skey_l and md_skey_u*. If a range predicate is in the form of *attrib* < *const* (or *attrib* > *const*), we translate it into a form of *attrib* ≤ *const'* (or *attrib* ≥ *const'*), where *const'* is the biggest sibling that is smaller than *const* (or the smallest sibling greater than *const*) in the hierarchy tree. For example, *year<1995* can be translated into *year≤1994*.

Consider a conjunctive predicate: $dim_1.hierarcy_1=l_1$ and $dim_2.hierarcy_2\leq l_2$ and $dim_3.hierarcy_1\geq l_3$ and $dim_3.hierarcy_1\leq l_4$ and $dim_4.hierarcy_1=l_5$., where $dim_i.hierarcy_i$ represents the *jth* hierarchy of the *ith* dimension and l_i are literals. This is a range conjunctive of predicates. To transform it, we first divide the predicates into two conjunctive predicates. One is an equation conjunctive: $dim_1.hierarcy_1=l_1$ and

$dim_4.hierarcy_1=l_5$, the other is a range conjunctive: *$dim_2.hierarcy_2 \le l_2$* and *$dim_3.hierarcy_1 \ge l_3$* and *$dim_3.hierarcy_1 \le l_4$.*

We suppose that the start and end offsets of the bits for *$dim_i.hierarcy_j$* is $[b_{ij}, e_{ij}]$, and the *h_skey* key of hierarchy member *l* that belongs to the *jth* hierarchy of the *ith* dimension is *$h\_skey_{ij}(l)$*. Then, the transformation works as follows:

//extract_mask_equal: exact the needed fields for equation predicates

$$extract\_mask\_\ equal \leftarrow 1^{e_{11}-b_{11}} 0^{e_{12}-b_{12}} \ldots 1^{e_{41}-b_{41}} \ldots.$$

//extract_mask_range: exact the needed fields for range predicates

$$extract\_mask\_range \leftarrow 0^{e_{11}-b_{11}} 0^{e_{12}-b_{12}} \ldots 1^{e_{22}-b_{22}} \ldots 1^{e_{31}-b_{31}} \ldots.$$

// md_skey_e: mask with expected results for equation predicates

$$md\_skey\_e \leftarrow h\_skey_{11}(l_1) 0^{e_{12}-b_{12}} \ldots 0^{e_{22}-b_{22}} \ldots h\_skey_{41}(l_5) \ldots.$$

//md_skey_l: mask with expected lower bound for range predicates

$$md\_skey\_l \leftarrow 0^{e_{11}-b_{11}} 0^{e_{12}-b_{12}} \ldots 0^{e_{22}-b_{22}} \ldots h\_skey_{31}(l_3) \ldots.$$

//md_skey_u: mask with expected upper bound for range predicates

$$md\_skey\_u \leftarrow 0^{e_{11}-b_{11}} 0^{e_{12}-b_{12}} \ldots h\_skey_{22}(l_2) \ldots h\_skey_{31}(l_4) \ldots.$$

Finally, for each tuple t, we do the following test:

extract_mask_equal & t = md_skey_e and (extract_mask_range & t) between md_skey_l and md_skey_u.

Like and IN conjunctive predicates transformation. We handle these types of predicates by transforming each one into an IN predicate. The value list is a set of *md_skey* which are produced by evaluating the predicates on the dimension tables.

B. Predicates on other fields

There are two ways to transform the predicates on non-dimensional attributes. One way is to transform the predicates into an IN predicate, as discussed above. For some hot fields, we can use the second way – encoding these fields into *d_skey* and *md_skey*, such that the predicates on these fields can be evaluated while the fact table is being scanned, similar to the predicates on dimensional attributes.

4.3 Reducer

After the transformation, a reduce operation with new predicates is scattered to all the data nodes which executes the operation in parallel. On each data node, the reduce operation is executed in a pipeline: scan the fact table → group result data→ sort result data, which is similar to the pipeline parallelization used by traditional RDBMS.

It is worth mentioning that during the table scan, the transformed predicates are evaluated in a batch job [2], which is more efficient than the traditional step-wise method. For example, for predicates col_1=2 and col_2=4, the traditional method needs to evaluate col_1=2 and col_2=4 one by one. In contrast, our method needs only a single comparison. We describe the detailed implementation of the scan algorithm in Section 5.1.

Finally, the grouping is executed based on the conjunctive result (called *groupcode*) of *group_mask* and *md_skey*. To improve the merge performance, we sort the aggregated data on *groupcode*.

4.4 Merger

Each data node transfers aggregated and sorted results to the master node, and the master node merges the data. As each data node operates on bit strings and the outputs are <*key, value*> pairs, residuary joins are needed to parse the keys (bit strings) and get their corresponding names according to those in the group-by clause. After the merging operation, we execute the having-clause and sort the data if necessary. The detailed implementation and the optimization of the merge operation are discussed in Section 5.2.

5 Implementation and Optimization Issues

In this section, we focus on the most crucial parts of the TRM execution model, the fact table scan step and the merge step.

5.1 Scan-Index: A Runtime Index

Partitioning the fact table vertically can improve the I/O performance. However, if the query involves multiple measures, this method is inefficient, because TRM has to scan each local decomposed fact table and then join the results. After the fact able is partitioned, the attributes of a tuple are distributed to different decomposed fact tables. However, they are logically a whole and are highly correlated. For instance, the *ith* tuple in the original fact table is the *ith* tuple in each decomposed fact table. If the tuple in the first decomposed fact table does not satisfy a predicate, the complete logical tuple does not satisfy the predicate. Therefore, there is no need to process the remaining tuples in the other decomposed fact tables that correspond to the same primary key.

Based on the above observation, we propose an algorithm to accelerate the processing of queries involving multiple measures. During the scan, we use an index (denoted as scan-index) to record the offsets of the tuples that have passed the predicate evaluation. We implement the scan-index as a bitmap, in which the *ith* bit indicates whether the *ith* tuple satisfies the predicates. The processing of the successive fact tables is conducted based on the scan-index. Let *TupNumInBlk* be the tuple number in a block, SI_{offset} be the offset of an indexed tuple in scan-index bitmap. We can calculate the block offset of an indexed tuple by $SI_{offset} / TupNumInBlk$, and the offset in block by $SI_{offset} \bmod TupNumInBlk$. In the following, we discuss different scan-index generation algorithms.

1. All the predicates in the conjunctive are on hierarchies. As mentioned earlier, most predicates operate on hierarchies. For this type of conjunctives, we need only one scan to produce the scan-index. The non-dimensional columns can be accessed by index lookups.

For instance, given the reduce operation (π: projection, σ: selection):

$\pi_{\text{measure1, measure2}}(\sigma_{\text{md\_skey \& "11100000111"="10001000101"}}$ (subjectName), its execution flowchart is that in figure 3. We need a full scan of the first table Fact_mearure1, and the second table Fact_measure2 can be accessed through the index.

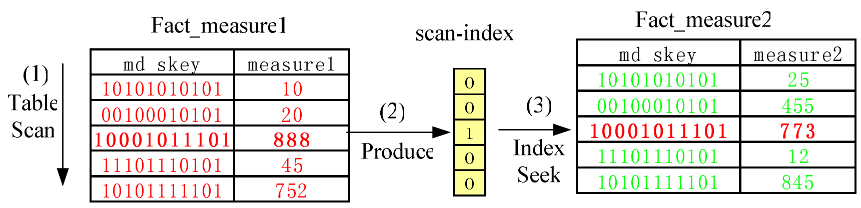


Fig. 3. Two-column access based on scan-index

2. Some predicates are on multiple measures or other descriptive information. We create the scan-index for such a query in two steps. First, we scan the first fact table, evaluate the predicates on this table and produce an initial scan-index. Second, we execute the other tables in a loop until all predicates are evaluated. The scan of each subsequent fact table is based on the scan-index generated by the previous scan. During each scan, we apply the predicates related to the current decomposed fact table and update the corresponding bits in the bitmap.

5.2 Parallelizing Merge Operation on Modern Hardware

Thus far, we have focused on the parallel scan operations. In fact, given thousands of data nodes, the merge operation can be highly time consuming too, as it is performed on only one node. Our design of LinearDB's merge operation are based on the following considerations: 1) Modern machines have large main memories, which are in general in GB level (a typical high-end server can own memory of TB level), and the intermediate results generated by the data nodes are usually in MB or KB level; 2) Modern processor has two or more cores which provide parallel processing capability; 3) Data have been sorted on each data node in our TRM model.

These characteristics inspire us to design a parallel one-way merge algorithm which operates only in memory. We partition the data horizontally and assign each thread a partition. Then, each thread executes the merge operation and writes data to the shared data structures in parallel. Although there can be small overlaps between the adjacent partitions, they will not incur waiting among the threads. This is because all the threads process data in the same order, such that when the previous thread begins to process the overlapping elements, the latter thread most likely has finished with those elements.

6 Experiment

We have built a prototype of LinearDB and implemented all features described in this paper. We used the pthread library and MPICH2 library (version 1.2.1p1) in our prototype, and implemented our algorithm in C++. In this section, we present some experimental results of LinearDB. We first analyze the scalability of LinearDB, and proceed to evaluate its overall performance.

Our datasets is from the SSB benchmark. We adopted a decomposed snowflake schema (described in Section 3), and populated each local decomposed fact table with 120M rows (30GB in SSB). We performed experiments on a cluster consisting of five PCs (detailed configurations are given in Table 1). The network bandwidth is 1Gbps.

Table 1. Cluster Configuration

Computer	CPU	Memory	Disk Capacity	OS
C1, C2	Intel Core2 Duo 1.87GHZ processor (1)	2G	130GB	32-bit Windows Vista
C3	Intel P4 3GHZ processor (1)	1G	80G	32-bit Windows XP
C4	Intel P4 3GHZ processor (1)	3 G	320G	32-bit Windows XP
C5	Intel Xeon E5310 1.6GHZ processors(2)	2G	1TB	32-bit Windows 2008

6.1 Scalability Analysis

As described in section 4, we divide the total cost of TRM execution model into four parts: the transformation cost (TC), the reduce cost (RC), the merge cost MC and other cost OC. The total cost expression is: TotalCost= TC+ RC + MC + OC.

The main cost of transformation (TC) is the lookup operation to get surrogate keys of the hierarchies in the predicates. We store each hierarchy of a dimension as an individual relation in memory, such that each lookup can be finished efficiently through a hash. As the processing on each node is limited to table scan, the reduce cost (RC) should vary monotonically with the size of the fact table (as shown in figure 4). Suppose the size of the fact table is F and there are N nodes with the same configuration in the cluster. Then the fragment of each data node is F/N. The cost can be denoted as RC=f (F/N). The merging cost MC is determined by the result size. Other costs are mainly network transmission cost which is related to the maximum size of the results generated by each data node. As there is only a small amount of aggregated data that need to be transferred on network, OC is usually very low.

Based on the analysis above, we carried out an experiment to analyze the time distribution of TRM execution model. The experiment was performed on C1 and C5, where C1 acted as a master node to perform the transform-ation, merge and other operations, and C5 acted as data nodes performing the reduce operation. We ran SSB Query 3.2 over a dataset of 30GB. This query is supposed to incur big TC+MC+OC, because it produces more results (six hundred records) and scans less data (just one column) compared with other SSB queries. Nevertheless, our experiments show that the proportion of TC+MC+OC to TotalCost is less than 0.12‰ for Query 3.2. Hence, we can conclude that the system performance is dominated by the reduce cost. (This is also confirmed in Section 6.3). As shown in Figure 4, the reduce cost is almost linear with the data size, thus we may deduce that TotalCost ≈ RC = f (F/N), which indicates that the system performance is almost linear to the data size on each data node.

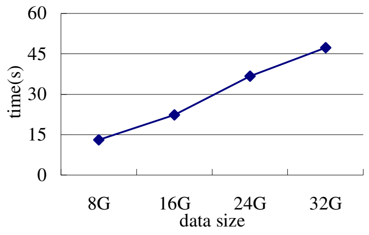


Fig. 4. Processing time vs data size

6.2 Fault Tolerance Analysis

Conventional databases on computer clusters usually experience high rates of failure or slowdown. LinearDB achieves fault tolerance by re-executing the reduce operation of failed nodes. If the coordinator does not receive response from a data node for a preset period of time, it will restart the reduce operation of this node on its backup node. To test the fault tolerance of LinearDB in the context of computer cluster, we executed the SSB 3.2 query on a 5-node cluster. To simulate failure, we terminated the slowest node before the completion of the query. As a result, LinearDB experienced an increase of about 10% in its query processing time. As the execution time of the terminated node was almost 2 times as much as that of its backup node, when we restarted the query on the backup node, the overall execution time was only slightly affected. This experiment also reveals a shortcoming of LinearDB: Query processing time is determined by the time the slowest node takes to complete its task. The current *Round-Robin* data distribution method does not consider different I/O speeds of different data nodes. Hence, we plan to design a new distribution algorithm according to the local disk I/O throughput in our future work.

6.3 Performance Evaluation

We compared the performance of LinearDB against that of PostgreSQL 8.3 on a dataset of 30GB. To minimize the influence of the hardware configuration, we conducted this experiment on one machine C5 and optimized PostgreSQL with 200MB *work_mem*. As shown in Figure 5, LinearDB exhibits a much more stable performance as compared with PostgreSQL, and outperforms PostgreSQL in most of the queries. In average, LinearDB is 1.73 times faster than PostgreSQL. In the best case, its performance is almost three times as fast as that of PostgreSQL.

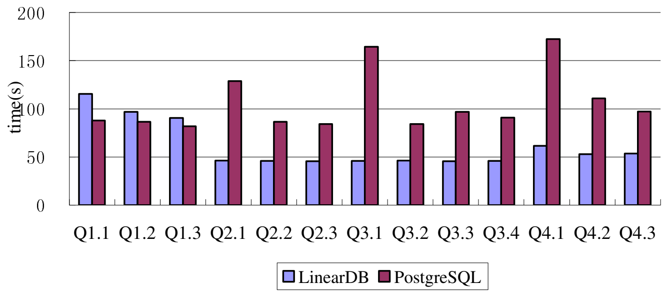


Fig. 5. PostgreSQL vs. LinearDB

We also observed that LinearDB is slower than PostgreSQL when running the first group of the SSB queries. This is due to two reasons: 1) These queries need three scans and there are too many tuples generated by the first scan. This suggests that scan-index should take into account of the selectivity of different predicates: the more selective the predicates, the earlier it should be executed. 2) The first group queries involve three measures and need to be evaluated on the detailed data. For instance, a production of two measures and a subsequent addition need to access two data files at the same time and cause high I/O cost. However, different from other methods, even in the worst case, the scan-index approach is equivalent to a full table scan operation because this index is memory resident and requires no random I/O.

6.4 Simulation of Merge

In the final set of experiments, we analyzed the characteristics of the merge operation. Due to the restriction of experimental condition, we adopted a simulation method. We populated a series of arrays with random data and then sorted the data to simulate data returned from the data nodes. The network transmission time can be calculated by the expression: *(tupleSize*tupleNumber) / NetworkBandwidth*. We supposed that *tuple-Size*=24byte (17 bytes for *md_skey*, 8 bytes for measure), and ran the experi- ments in the extreme case, where the cluster contains ten thousand nodes and each node produces 1000 tuples. We executed four merge threads in parallel. The results show that even in this extreme case, the merge can be finished in less than 80ms on C5 and the network transmission time is less than 0.2ms (we assumed that the data generated by each data node arrive at the master node at the same time). Thus the total merge time is less than 81ms, which is a very small compared to the time required by massive data scan. The execution will be faster when running on a high-end server.

7 Summary and Future Work

In this paper, we investigated the scalability and fault tolerance problem for analytical RDBMS in the context of cluster environment. Our design considers two levels: the schema level and the execution level. On the schema level, we improve the star (snowflake) schema and propose the decomposed snowflake schema, which eliminates the star (snowflake) join in query processing. On the execution level, we propose TRM execution model, which handles all data warehouse queries by three operations: transform, reduce, and merge. Based on these techniques, we divide a data warehouse-style query into many independent tasks, which can be distributed and executed independently across a cluster. We also employs new techniques such as multiple-column queries optimization, batch predicates evaluation, etc, to improve the performance of our new query processing model. We implemented our approach in a prototype called LinearDB. Our experiment results have illustrated that LinearDB can achieve the linear scale-out in the cluster environment, and it is 1.73 times faster than PostgreSQL in average. Our research also reveals that scan operation is more suitable for cluster computing than join operation.

Many challenges remain as future work. To address unpredictable workload, we need to use scan sharing techniques to reduce contention and amortize I/O cost between queries. LinearDB answers all queries by table scan and some residuary joins with an almost constant time. This is not suitable for ad hoc queries, which can be handled more efficiently by leveraging indexes.

References

1. Karayannidis, N., Tsois, A., Sellis, T.K., Pieringer, R., Markl, V., Ramsak, F., Fenk, R., Elhardt, K., Bayer, R.: Processing Star Queries on Hierarchically-Clustered Fact Tables. In: Proceedings of the 28th VLDB Conference, pp. 730–741 (2002)
2. Raman, V., Swart, G., Qiao, L., Reiss, F., Dialani, V., Kossmann, D., Narang, I., Sidle, R.: Constant-Time Query Processing. In: Proceedings of the 24th ICDE Conference, pp. 60–69 (2008)
3. Markl, V., Ramsak, F., Bayer, R.: Improving OLAP Performance by Multidimensional Hierarchical Clustering. In: Proceedings of the IDEAS 1999, pp. 165–177 (1999)
4. WinterCorp: 2005 TopTen Program Summary, `http://www.wintercorp.com/WhitePapers/WC_TopTenWP.pdf`
5. `http://hadoop.apache.org`
6. Korth, H.F., Kuper, G.M., Feigenbaum, J., Gelder, A.V., Ullman, J.D.: SYSTEM/U: adatabase system based on the universal relation assumption. TODS 9(3), 331–347 (1984)
7. Chaudhuri, S., Dayal, U.: An Overview of Data Warehousing and OLAP Technology. SIGMOD Record 26(1), 65–74 (1997)
8. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: Proceeding of the 6th Symposium on Operating System Design and Implementation (OSDI 2004), pp. 137–150 (2004)
9. `http://hive.apache.org/`
10. Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D.J., Rasin, A., Silberschatz, A.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2(1), 922–933 (2009)
11. Largest Commercial Database in Winter Corp. TopTen? Survey Tops One Hundred Terabytes, `http://test.wintercorp.com/PressReleases/ttp2005_pressrelease_091405.htm`
12. Pavlo, A., Paulson, E., Rasin, A., Abadi, D.J., DeWitt, D.J., Madden, S., Stonebraker, M.: A comparison of approaches to large-scale data analysis. In: Proceedings of the 35th SIGMOD Conference, pp. 165–178 (2009)
13. Olston, C., Reed, B., Srivastava, U., et al.: Pig latin: a not-so-foreign language for data. In: Proceedings of the 34th SIGMOD Conference, pp. 1099–1110 (2008)
14. Theodoratos, D., Tsois, A.: Heuristic Optimization of OLAP Queries inMultidimensionally Hierarchically ClusteredDatabases. In: DOLAP 2001(2001)
15. `http://hbase.apache.org`
16. Bayer, R.: The universal B-Tree for multi-dimensional Indexing: General Concepts. In: Masuda, T., Tsukamoto, M., Masunaga, Y. (eds.) WWCA 1997. LNCS, vol. 1274. Springer, Heidelberg (1997)
17. Abadi, D.J.: Data Management in the Cloud: limitations and Opportunities. IEEE Bulletin of the Technical Committee on Data Engineering 32(1), 3–12 (2009)

Genetic Algorithm Based QoS-Aware Service Compositions in Cloud Computing

Zhen Ye[1,2], Xiaofang Zhou[1,3], and Athman Bouguettaya[2]

[1] School of Information Technology and Electrical Engineering
The University of Queensland, Australia
[2] CSIRO ICT Centre, Australia
[3] School of Information, Renmin University of China, China
Key Lab of Data Engineering and Knowledge Engineering,
Ministry of Education, China
zhenye@itee.uq.edu.au, zxf@uq.edu.au, Athman.Bouguettaya@csiro.au

Abstract. Services in cloud computing can be categorized into two groups: *Application services* and *Utility Computing Services*. Compositions in the application level are similar to the Web service compositions in SOC (Service-Oriented Computing). Compositions in the utility level are similar to the task matching and scheduling in grid computing. Contributions of this paper include: 1) An extensible QoS model is proposed to calculate the QoS values of services in cloud computing. 2) A genetic-algorithm-based approach is proposed to compose services in cloud computing. 3) A comparison is presented between the proposed approach and other algorithms, i.e., exhaustive search algorithms and random selection algorithms.

1 Introduction

Cloud computing is emerging as the new paradigm for the next-generation distributed computing. It has attracted a lot of attention in both academia and industry. An increasing number of organizations have started to migrate their IT infrastructure to the cloud. Several companies such as Amazon, Microsoft and IBM are already offering cloud solutions in the market. The vision of cloud computing is that computing will not be conducted on local computers in the future, but on distributed facilities operated by third-party computing utilities [1]. Cloud computing aims at uniformly exposing hardware and software as a service that can be rented at will [2]. In this new framework, organizations no longer require the large capital outlays in hardware to deploy their services. They need not be concerned about overprovisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or underprovisioning for one that becomes popular, thus missing potential customers and revenue.

Services in cloud computing can be categorized into application services and utility computing services [2]. Almost all the software/applications that are available through the Internet are application services, e.g., flight booking services, hotel booking services. Utility computing services are software or virtualized hardware that support application services, e.g., virtual machines, CPU services, and storage services. Service compositions in cloud computing therefore include compositions of application services and utility computing services. Compositions in the application level are similar to the Web service compositions in SOC. Compositions in the utility level are similar to the task matching and scheduling in grid computing. A composite application service fulfills several tasks (i.e. *abstract services*). Each task is implemented by several substitute application services (i.e. *concrete services*). The choice among these substitute services is based on their non-functional properties, which are also referred to as Quality of Service (QoS). QoS values of these substitute application services are further dependent on the choices of utility computing services. In a word, once a concrete application service is selected for each abstract service, the following decisions have to be made: *matching*, i.e. assigning concrete application services to utility computing services, and *scheduling*, i.e. ordering execution sequence of application services.

Several approaches and systems are proposed to solve Web service composition problems in SOC. Most of them [3] [4] only consider the compositions in the application level. Composition approaches in cloud computing need to consider compositions both in the application level and utility computing level. Besides, most existing composition approaches in SOC [3] [4] use integer programming to find the global optimized solution. Although this is useful for small-scale compositions, it incurs a significant performance penalty if applied to large-scale composition problems such as compositions in cloud computing [6]. Contrasts to these existing approaches, Genetic Algorithms (GAs) are heuristic approaches to iteratively find near-optimal solutions in large search spaces. There is ample evidence regarding the applicability of GAs for large-scale optimization problems [5] [6]. Whereas, no GA based approach is available to compose services in cloud computing.

In this paper, a genetic-algorithm-based service composition approach is proposed for cloud computing. In particular, a coherent way to calculate the QoS values of services in cloud computing is presented. At last, comparisons between the proposed approach and other approaches show the effectiveness and efficiency of the proposed approach. The rest of the paper is structured as follows: Section 2 illustrates the background and preliminaries of service composition in cloud computing. Section 3 elaborates the details of the proposed approach. Section 4 evaluates the approach and shows the experiment results. Section 5 presents the related work to the proposed approach. Section 6 concludes this paper and highlights some future work.

2 Preliminaries

This section presents preliminary knowledge about cloud computing, service compositions in cloud computing. Genetic algorithms are also introduced at the end of this section. Services in a cloud, refers to both the applications delivered as services over the Internet and the hardware and system software in the data centers that provide those services [2]. Cloud computing provides easy access to *Application Services* (i.e. SaaS) and *Utility Computing Services* (UCS) (Fig. 1).

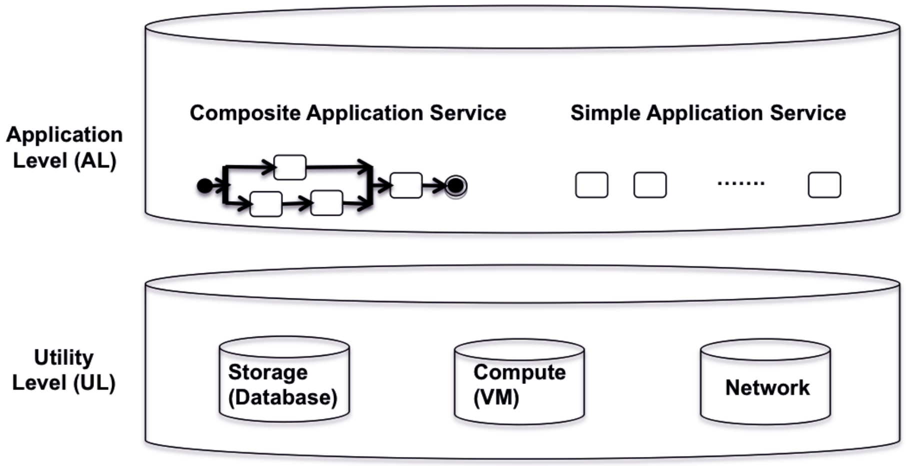


Fig. 1. Cloud System

– Application Services are the most visible services to the end users. Examples of application services include: Salesforce's CRM applications, Google Apps etc. Application services that contain other component application services are *Composite Application Services*. *Simple Application Services* do not contain other component application services. *Application Users* can be end users or other application services. *Application Providers* are providers of application services.
– Utility Computing Services. Some vendors use terms such as PaaS (Platform as a Service) or IaaS (Infrastructure as a Service) to describe their products. In this paper, PaaS and IaaS are considered together as UCSs. PaaS are platforms that are used to develop, test, deploy and monitor application services. For example, Google has Google App Engine works as the platform to develop, deploy and maintain Google Apps. Microsoft Azure and Force.com are also examples of PaaS. IaaS services provide fundamental computing resources, which can be used to construct new platform services or application services. UCSs can be categorized into computation services, i.e., *Virtual Machines* (VMs); storage services, i.e., *Databases*; and network services. *UCS Users* are these application providers or other utility computing services etc. *UCS Vendors* are these companies or organizations that make their computing resources available to the public.

2.1 Service Compositions in Cloud Computing

A composite service is specified as a collection of abstract application services according to a combination of control-flow and data-flow. Control-flow graphs are represented using UML activity diagrams. Each node in the graph is an abstract application service. There are four control-flow patterns. For example, Fig. 2 shows a composite service consists of four patterns of control-flows. S_1 and

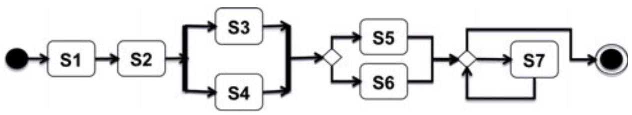


Fig. 2. Control Flows

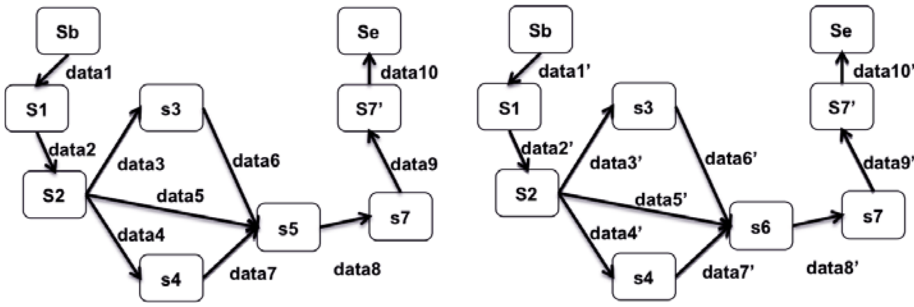
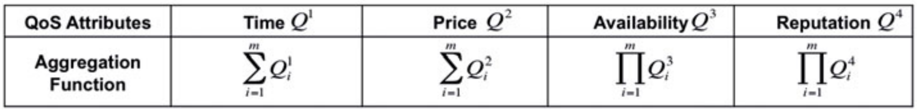


Fig. 3. Data Flow Graphs

S_2 run in a sequence pattern. S_3 runs in parallel with S_4 (parallel pattern). After that, either S_5 or S_6 is selected to run (conditional pattern). Finally, S_7 cycles for a certain times (loop pattern). There are several data-flow graphs for the same control-flow graph, if the control-flow graph contains conditional patterns. Fig. 3 shows the two data-flow graphs corresponding to the control-flow shown in Fig. 2. Directed acyclic graphs (DAGs) are used to represent data-flow graphs. The start node of an edge is denoted as *source service*, the node where the edge ends is denoted as *destination service*. Source services must be executed before the destination services. The destination service can only be executed after all its source services are finished. Node S_b represents the start point of the composite service. S_e represents the end point. The data items transferred between these abstract application services form a set $D = \{data_i, 1 \leq i \leq d\}$.

A set of k_n concrete application services $\{s_{n1}, s_{n2}, \ldots, s_{nk_n}\}$ is available to execute the abstract service S_n. A concrete application service can be executed on several virtual machines, databases and network services. After mapping each abstract service to a concrete application service, VM UCSs and Database UCSs need to be selected for each application service. Network UCSs need to be selected for each data transfer in the data-flow graph. Assume each VM can only execute one application service at a time. A late application service can only execute on the VM after the former application services finish their executions. To sum up, any solution to a composition problem in cloud computing includes: 1) Map the abstract application services to concrete application services and corresponding UCSs (VM, database and network services). 2) Schedule the execution order of the application services. This execution order is a topological sort [7] of the data-flow graph, i.e. a total ordering of the nodes in the DAG that obeys the precedence constraints.



QoS Attributes	Time Q^1	Price Q^2	Availability Q^3	Reputation Q^4
Aggregation Function	$\sum_{i=1}^{m} Q_i^1$	$\sum_{i=1}^{m} Q_i^2$	$\prod_{i=1}^{m} Q_i^3$	$\prod_{i=1}^{m} Q_i^4$

Fig. 4. Aggregation Functions for each QoS Attribute

2.2 QoS Model

QoS attributes contains (1) ascending QoS attributes, i.e. a higher value is better; (2) descending QoS attributes, i.e. a smaller value is better; (3) equal QoS attributes, i.e. no ordering but only equality, e.g. security protocol should be X.509. Four QoS attributes are considered in this work: response time, price, availability and reputation. Among them, time and price belong to the descending attributes while availability and reputation belong to the ascending attributes. Vector $\boldsymbol{Q} = Q^1, Q^2, Q^3, Q^4$ denotes all the available QoS attributes. $Q^i, 1 \leq i \leq 4$ represents time, price, availability and reputation.

QoS values of an application service consist of three parts: execution, network and storage QoSs. Existing QoS models in SOC [3] only consider the execution QoSs. *Execution QoS* refers to the QoS value for executing an application service in a specified VM. Same application service has different execution QoS in different VMs. *Network QoS* refers to the QoS for transferring data from one application service to another using a specified network UCS. Data transfers are determined by the source services and the destination services. Each data will be transferred as soon as the source service produces them. Hence, network QoS values are only calculated at the destination services. *Storage QoS* refers to the QoS for storing certain amount of data for a certain time using specified database service. Assume no data will be stored during the execution of an application service. Therefore, the only data needs to be stored are the input data. For example, a destination service has two input data. One input data arrives early, the other arrives later. The earlier arrived data need to be stored when waiting the second input data to arrive. The QoS value for a service therefore equals to the sum of execution QoS, network QoS and storage QoS. Fig. 4 shows the aggregation functions for calculating the overall QoS for composite services. m is the number of component services in the composite service. QoS values are normalized using Simple Additive Weighting (SAW), which is also used in [3]. The best QoS values are normalized to 0, the worst QoS values are normalized to 1. Thus, higher normalized values indicate worse quality.

QoS constraints (denoted as QC) for composite services have two types: *Global Constraints* and *Local Constraints*. Global Constraints are the QoS constraints for the overall composite service, while Local Constraints apply to component services within the composition. A global constraint (GC) for a given QoS attribute Q^l is denoted as GC^l. Local constraints are denoted as LC^l. Constraints on different QoS attributes are transformed into inequality constraints [8]. QC^1 (time) and QC^2 (price) can be transformed by subtract the threshold to the constraints, e.g. $QC^1 \leq 1$ *minute* is transformed to $QC^1 \Leftarrow QC^1 - 1 \leq 0$;

$QC^2 \leq 5\ USdollars$ is transformed to $QC^2 \Leftarrow QC^2 - 5 \leq 0$. QC^3 (availability) and QC^4 (reputation) can be transformed by subtracting the QoS value from the threshold, e.g. $QC^3 \geq 0.9$ is transformed to $QC^3 \Leftarrow 0.9 - QC^3 \leq 0$. Constraints on equal QoS attributes can be transformed using this function: $QC \Leftarrow |QC| - \epsilon \leq 0$, where ϵ is the tolerance allowed range (a very small value).

2.3 Genetic Algorithms

Genetic Algorithms (GAs) are heuristic approaches to iteratively find near-optimal solutions in large search spaces. Any possible solution to the optimization problem is encoded as a *Chromosome* (normally a string). A set of chromosomes is referred to as a *Population.* The first step of a GA is to derive an initial population. A random set of chromosomes is often used as the initial population. This initial population is the first generation from which the evolution starts. The second step is *selection.* Each chromosome is eliminated or duplicated (one or more times) based on its relative quality. The population size is typically kept constant. The next step is *Crossover.* Some pairs of chromosomes are selected from the current population and some of their corresponding components are exchanged to form two valid chromosome. After crossover, each chromosome in the population may be *mutated* with some probability. The mutation process transforms a chromosome into another valid one. The new population is then *evaluated.* Each chromosome is associated with a *fitness value*, which is a value obtained from the objective function (details will be discussed in section 3). The objective of the evaluation is to find a chromosome that has the optimal fitness value. If the stopping criterion is not met, the new population goes through another cycle (iteration) of selection, crossover, mutation, and evaluation. These cycles continue until the stopping criterion is met.

3 QoS-Aware Service Composition in Cloud Computing

Assume there are m VM UCSs $(vm_1, vm_2, \ldots, vm_m)$, p database UCSs (db_1, db_2, $\ldots$, db_p) and q network UCSs $(net_1, net_2, \ldots, net_q)$ in different cloud systems. Each composition solution (chromosome) consists of two parts, the matching string (ms) and the scheduling string (ss). ms is a vector of length n, such that $ms(i) = s_j vm_x db_y net_z$, where $1 \leq i \leq n$, $1 \leq j \leq k_n$, $1 \leq x \leq m$, $1 \leq y \leq p$ and $1 \leq z \leq q$. A matching string means that abstract service S_i is assigned to concrete service s_{ij} which is lodged on virtual machine vm_x and has database service db_y, network service net_z. The scheduling string is a topological sort of the data-flow graph. $ss(k) = i$, where $1 \leq i, k \leq n$; i.e. service S_i is the kth running service in the scheduling string. Thus, a chromosome represents the mapping from each abstract service to concrete service and UCSs, together with the execution order of the application services. Fig. 5 shows a solution to the composite problem that has the control-flow shown in Fig. 2, and the data-flow shown in Fig. 3 (left DAG). In this solution, ms represents the mapping string, e.g., abstract service S_1 is mapped to application service S_{11}, S_{11} is further deployed on virtual machine vm_1 and database db_1. The network service for

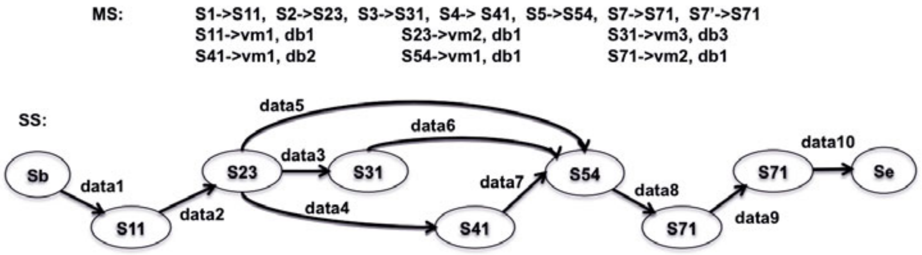


Fig. 5. Composition Solution

a transferred data is determined when the source service and the destination service are mapped to the corresponding virtual machine and database services. *ss* represents the scheduling string of the solution, e.g., the execution order of this solution in Fig. 5 is $S_{11}, S_{23}, S_{31}, S_{41}, S_{54}, S_{71}, S_{71}$.

3.1 Genetic Algorithm Based Approach

In the first step, a predefined number of chromosomes are generated to form the initial generation. The chromosomes in a generation are first ordered by their fitness values (explained later) from the best to worst. These having the same fitness value are ranked arbitrarily among themselves. Then a rank-based roulette wheel selection schema is used to implement the selection step [9]. There is a higher probability that one or more copies of the better solution will be included in the next generation, since a better solution has a larger sector angle than that of a worse solution. In this way, the chromosomes formed the next generation are determined. Notice that the population size of each generation is always P.

The crossover operator for a matching string randomly chooses some pairs of the matching strings. For each pair, it randomly generates a cut-off point to divide both matching strings into two parts. Then the bottom parts are exchanged. The crossover operator for a scheduling string randomly chooses some pairs of the scheduling strings. For each pair, it randomly generates a cut-off point, which divides the scheduling strings into top and bottom parts. The abstract application services in each bottom part are reordered. The new ordering of the services in one bottom part is the relative positions of these services in the other original scheduling string in the pair. This guarantees that the newly generated scheduling strings are valid schedules. Fig. 6(a) demonstrates the crossover operator for a scheduling string.

The mutation operator for a matching string randomly selects an abstract service and randomly replaces the corresponding concrete service and other utility computing services. The mutation operator for a scheduling string randomly chooses some scheduling strings. It then randomly selects a target service. The *valid range* of this target service is the set of the positions in the scheduling string at which the target service can be placed without violating any data

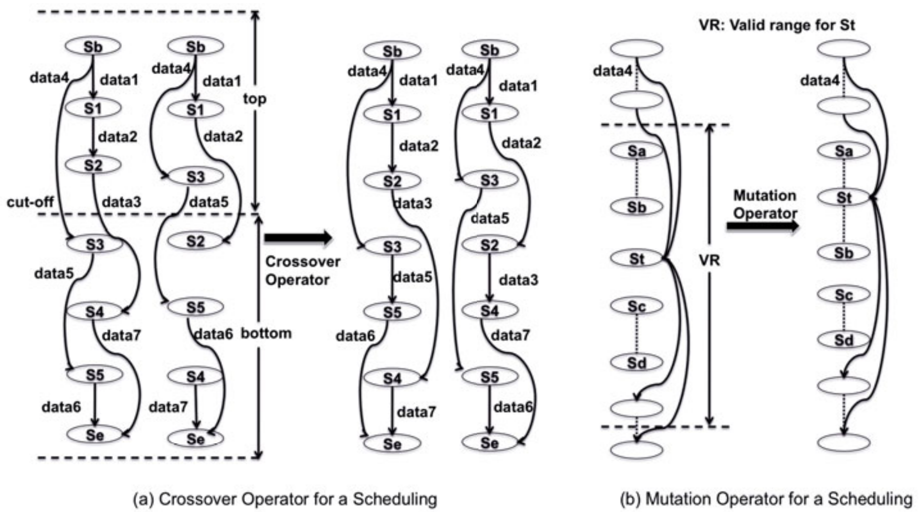


Fig. 6. Crossover and Mutation Operators

dependency constraints. The valid range is after all source services of the target service and before any destination service of the target service. The mutation operator can move this target service randomly to another position in the scheduling string within its valid range. Fig. 6(b) demonstrates the mutation operator for a scheduling string. s_v is between s_b and s_c before the mutation, it is between s_a and s_b after the mutation operator.

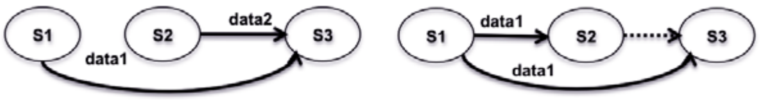
After crossover and mutation operators, GA will evaluate the chromosomes using *fitness function*. The fitness function needs to maximize some QoS attributes (i.e. ascending attributes), minimize some other attributes (i.e. descending attributes) and satisfy other QoS attributes (i.e. equal QoS attributes). In addition, the fitness function must penalize solutions that do not meet the QoS constraints and drive the evolution towards satisfaction. The distance from constraint satisfaction for a solution c is defined as:

$$D(c) = \Sigma_{i=1}^{l} QC^i(c) \times e_i \times weight^i, e_i = \begin{cases} 0 & QC^i(c) \leq 0 \\ 1 & QC^i(c) > 0 \end{cases} \tag{1}$$

where $weight^i$ indicates the weight of the QoS constraint. Notice that this distance function for constraints include both local and global constraints specified. The fitness function for a chromosome c is then defined as follows:

$$F(c) = \Sigma_{i=1}^{4} w^i * Q^i(c) + weight_p * D(c) \tag{2}$$

w^i are the weights for each QoS attribute. $weight_p$ is the *penalty factor*. Several features are highlighted when calculating the fitness function based on the match string and the scheduling string:



(a) Example 1 (b) Example 2 for Data Forwarding

Fig. 7. Example of Scheduling String

1. Services are executed exactly in the order specified by the scheduling string. For example, Fig. 7(a) shows a scheduling string for a composition. Assume there are two different match strings for this ss. a) ms_1: Let S_1 and S_2 be assigned to the same VM vm_1, and S_3 be assigned to another VM vm_2. In this chromosome, because S_1 is to be executed before S_2, $data_1$ is available before $data_2$. Thus, $data_1$ will be transferred to S_3 before $data_2$. And $data_1$ will be stored in S_3's database service till $data_2$ has been transferred to S_3. b) ms_2: Let the three services S_1, S_2, and S_3 be assigned to three different VMs vm_1, vm_2 and vm_3. S_2 starts to execute just after S_1 starts, S_1 and S_2 can be considered to start their execution at the same time. If $data_2$ is available (S_2 executes faster) before $data_1$, $data_2$ will be stored in S_2's database service till $data_1$ has been transferred to S_3.

2. Another important feature is *data forwarding* [5]. For an input data, the source service can be chosen among the services that produce or consume this input data. All the consumers of this input data can be forwarders. For example, Fig. 7(b) shows a scheduling string. S_2 and S_3 both have the input data from S_1. S_2 may forward $data_1$ from S_1 to S_3, i.e. shown as the dashed line in Fig. 7(b). This kind of data forwarding is not allowed in our work. Data must be only transferred from the original data producer to its consumers.

Stop criterions for the proposed approach are: 1) Iterate until the constraints are met (i.e. $D(c) = 0$). 2) If this does not happen within $MAXGEN$ generations, then iterate until the best fitness value remains unchanged for a given number ($MAXGEN$) of generations. 3) If neither 1) nor 2) happens within $MAXGEN$ generations, then no solution will be returned.

3.2 Handling Multiple Data Flow Graphs

Assume the composite service (e.g. shown in Fig. 2) has multiple data-flow graphs (shown in Fig. 3). For each data-flow graph, an optimal composition solution can be generated using the proposed GA-based approach. Since each of the optimal solution only covers a subset of the composite service, further actions are needed to aggregate these partial composition solutions into an overall solution. Assume the composite service has f data-flow graphs (i.e. $dfg_1, dfg_2, \ldots, dfg_f$). The approach adopts the following strategies to aggregate multiple solutions into an overall solution:

- Given an abstract service S_i, if S_i only belongs to one data-flow graph (e.g. dfg_j), then the proposed approach selects dfg_j's solution $chromosome_j$ to execute abstract service S_i.

– Given an abstract service S_i, if S_i belongs to more than one data-flow graphs, then there are many solutions can be used to execute S_i. The proposed approach will select the most frequently used solution (from execution history), or ask end users to select a preferable solution.

4 Experiment and Evaluation

Our experiments consist of two parts. First, comparisons are conducted between the proposed approach and other approaches in small-scale scenarios. Second, comparisons are conducted in large-scale scenarios. All the experiments are conducted on computers with Intel Core 2 Duo 6400 CPU (2.13GHz and 2GB RAM).

4.1 Creation of Experimental Scenarios

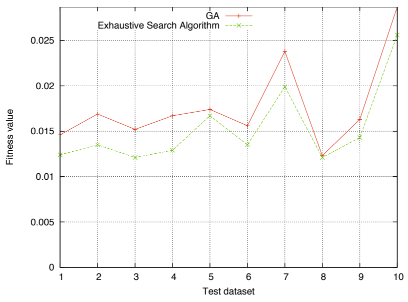
Randomly generated scenarios are used for the experiments. Each scenario contains a control-flow graph and a data-flow graph. QoS values of different concrete services, virtual machines, database services and network services for each abstract service are generated randomly with uniform probability. A scenario generation system is designed to generate the scenarios for experiments. The system first determines a root pattern (i.e. sequence, conditional, parallel, loop patterns) with uniform probability for the control-flow. Within this root, the system chooses with equal probability to either place an abstract services into it or to choose another composition pattern as substructure. This procedure ends until the generation system has spent the predefined number (n) of abstract services. All the conditional patterns have 2 possible options, either of them has the probability of 0.5. Each loop pattern will run for twice. There are k candidate concrete services to implement each abstract service. The number of data transferred between each abstract services in the flow graph is d. Each concrete service can be lodged in m virtual machines, p database services and q network services. These variables are predefined and used as input (denoted as $\{n, k, d, m, p, q\}$) to the generation system. Small-scale scenarios have the input $\{5, 2, 6, 3, 3, 3\}$. Large-scale scenarios have 100 abstract services. Each abstract service can be executed by 30 concrete services. 120 data items are transferred between services and each concrete service is suitable to run in 20 different VMs, 20 different database services and 20 network services. The four QoS attributes and the four QoS constraints have same weight equals 1. The execution QoS, network QoS and storage QoS were randomly generated with uniform distribution from the following intervals: $Q^1(Time) \in [100, 2000]$, $Q^2(Price) \in [200, 1000]$, $Q^3(Availability) \in [0.9750, 0.9999]$ and $Q^4(Reputation) \in [1, 100]$.

Every approach runs 50 times for each scenario. All the results shown below are the average values from these experiments. Each experiment for the GA-based approach starts from a different initial population each time. The probability of crossover $p_{cross} = 0.4$ is the same for the matching string and scheduling string. The probability of mutation $p_{mut} = 0.1$ is also the same for the matching string and scheduling string. The approach uses rank-based roulette wheel schema for selection. The angle ratio of the sectors on the roulette wheel for two adjacently ranked chromosomes, i.e. R, was chosen to be $1 + 1/P$, where P is the population size. By using this simple formula, the angle ratio between the slots of the best and median chromosomes for $P = 50$ (and also for $P = 200$ for large-scale scenarios) is very closely to the optimal empirical ratio value of 1.5 in [10]. $MAXFIT$ equals to 150. $MAXGEN$ equals to 1000. *Exhaustive search approach* would traverse all the possible solutions to the composition problem and find the optimized solution that has the smallest fitness value. Although this approach would always find the most optimal composition solution, the execution time is extremely high. *Random selection approach* is also a GA-based approach. This approach would randomly select chromosomes to form a new generation. Comparisons with these approaches show the effectiveness and efficiency of the proposed approach. Integer Programming (IP) approaches have been proposed to solve QoS-aware service composition in SOC. The IP approach is implemented using LPSolve [11], which is an open source integer programming system. Comparisons with IP approach show the scalability of the proposed approach.

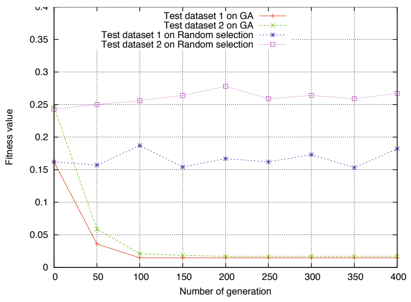
4.2 Experiments Results

Small-scale experiments are conducted on 10 different test datasets. We only show two of them in Fig. 8(b) to make the graph much easier to read. Fig. 8(a) shows the results between the proposed approach and the exhaustive search approach. Proposed GA-based approach would always find near-optimal solution compared to exhaustive search algorithms. Fig. 8(b) shows the comparisons between the proposed approach and the random selection solution. As shown in this figure, proposed approach will always reach an optimized fitness value while random selection seldom converges. To sum up, the proposed GA based approach will always reach an optimal fitness value and the converged point is very close to the actual optimal point. Fig. 9 shows the efficiency of the proposed approach. These experiments are conducted on small-scale scenarios. Each test dataset has the same configuration, except for the number of concrete services for each abstract service. As shown in Fig. 9, the execution time increases quickly at the beginning, but keeps stable when the number of concrete services for each abstract service is larger than 200.

As shown in Fig. 10(a), IP approach performs as good as the GA based approach at the beginning. Notice that, when the number of the abstract services becomes more than 40, IP approaches would cost exponential growing time to solve the composition problems. Fig. 10(b) shows the fitness value's trend corresponding to the increment of the number of the abstract services. Both IP approach and GA based approach behave well when the number of abstract services is relatively small. When the number of abstract services increases, the optimal fitness value obtained from GA based approach also increases. This is because population size and other related variables stay the same when the number of the abstract services varies. Hence, GA based approach are more scalable and efficient than IP approaches.



(a) Fitness VS. Dataset



(b) GA VS. Random selection

Fig. 8. Experiment Result 1

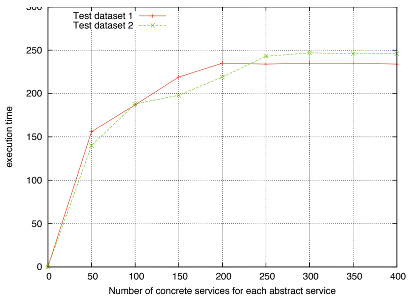
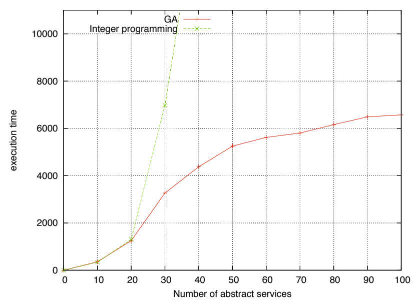


Fig. 9. Time VS. Concrete services

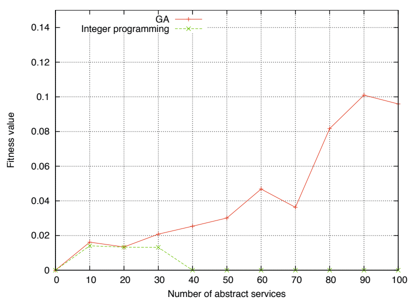
5 Related Work

Most composition approaches in SOC use linear programming methods. [3] presents two approaches: one focuses on local optimization, the other on global optimization. They use integer programming to solve the global optimization problem. The limit of this approach is that all QoS attributes need to be linearized as integer programming is a linear programming approach. [12] proposes an improved approach based on [3], using Mixed Linear Programming (MILP) approach. They also introduce several concepts such as loop peeling and negotiation mechanisms to address situation where no feasible solution can be found. [13] proposes an approach to decompose global QoS constraints into local constraints with conservative upper and lower bounds. These local constraints are resolved by using an efficient distributed local selection strategy.

All of the aforementioned approaches only consider the service composition problems in small-scale scenarios. These linear programming approaches are not suitable to handle large-scale scenarios problems, e.g. service composition in cloud computing. [14] was the first to use GA for optimization of QoS-aware compositions in SOC. The results show that their GA implementation scales



(a) GA VS. Integer Programming on Time



(b) GA VS. Integer Programming on Fitness

Fig. 10. GA VS. Integer Programming Approach

better than linear programming. [15] presents a GA and a Culture Algorithm (CA) for Web service compositions. The first algorithm is similar to [14], the latter uses a global belief space and an influence function that accelerate the convergence of the population. [6] presents a mutation operator which consider both the local and global constraints to accelerate the converge of the population.

Existing GA-based approaches are solely focus on service composition in application level, which do not consider the computing resources composition. Service composition in cloud computing involves application service composition and computing resources matching and scheduling. In this paper, a genetic algorithm based approach is proposed to compose services in cloud computing, by combining QoS-aware service composition approaches and resources matching and scheduling approaches.

6 Conclusion

A genetic algorithm based approach is presented for service compositions in cloud computing. Service compositions in cloud computing involve the selections of application services and utility computing services. The chromosome size is bound to the number of n of abstract services. The number of possible application services and utility computing services only augments the search space. For small-scale scenarios, the proposed approach finds optimal solutions. For larger-scale problems, it outperforms the integer programming approach. This is a beginning to propose robust service composition approaches in cloud computing. Future work may focus to eliminate several assumptions: 1) QoS values for each component are known in this research. Calculating the QoS values at runtime is one direction; 2) penalty factor in the fitness function is static. More dynamic fitness functions can be used to improve the performance of the approach. 3) novel crossover and mutation operators may accelerate the converge.

References

1. Buyya, R., Yeo, C., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems 25(6), 599–616 (2009)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A berkeley view of cloud computing. Tech. rep. (February 2009)
3. Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)
4. Rosenberg, F., Celikovic, P., Michlmayr, A., Leitner, P., Dustdar, S.: An end-to-end approach for QoS-aware service composition. In: Proceedings of 13th IEEE International EDOC Conference, pp. 1–4 (September 2009)
5. Wang, L., Siegel, H., Roychowdhury, V., Maciejewski, A.: Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. Journal of Parallel and Distributed Computing 47(1), 8–22 (1997)
6. Rosenberg, F., Muller, M., Leitner, P., Michlmayr, A., Bouguettaya, A., Dustdar, S.: Metaheuristic Optimization of Large-Scale QoS-aware Service Compositions. In: 2010 IEEE International Conference on Services Computing, pp. 97–104. IEEE, Los Alamitos (2010)
7. Cormen, T.: Introduction to algorithms. The MIT press, Cambridge (2001)
8. Coello, C., Carlos, A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)
9. Srinivas, M., Patnaik, L.: Genetic algorithms: A survey. Computer 27(6), 17–26 (1994)
10. Whitley, D., et al.: The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In: Proceedings of the Third International Conference on Genetic Algorithms, vol. 1, pp. 116–121, Citeseer (1989)
11. Berkelaar, M., Eikland, K., Notebaert, P., et al.: lpsolve: Open source (mixed-integer) linear programming system. Eindhoven U. of Technology
12. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Transactions on Software Engineering, 369–384 (2007)
13. Alrifai, M., Risse, T.: Combining global optimization with local selection for efficient QoS-aware service composition. In: Proceedings of the 18th International Conference on World Wide Web, pp. 881–890. ACM, New York (2009)
14. Canfora, G., Di Penta, M., Esposito, R., Villani, M.: An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1069–1075. ACM, New York (2005)
15. Gaber, J., Bakhouya, M.: An affinity-driven clustering approach for service discovery and composition for pervasive computing. In: ACS/IEEE International Conference on Pervasive Services, pp. 277–280. IEEE, Los Alamitos (2006)

Energy-Efficient Tree-Based Indexing Schemes for Information Retrieval in Wireless Data Broadcast*

Jiaofei Zhong[1], Weili Wu[1], Yan Shi[1], and Xiaofeng Gao[2]

[1] The University of Texas at Dallas, Department of Computer Science, 800 West Campbell Road, Richardson, TX 75080-3021
{fayzhong,weiliwu,yanshi}@utdallas.edu
[2] Georgia Gwinnett College, 1000 University Center Lane, Lawrenceville, GA 30043
xgao@ggc.edu

Abstract. Mobile computing can be equipped with wireless devices to allow users retrieving information from anywhere at anytime. Recently, wireless data broadcast becomes a popular data dissemination method, especially for broadcasting public information to a large number of mobile subscribers at the same time. *Access Latency* and *Tuning Time* are two main criteria to evaluate the performance of a data broadcast system. Indexing can dramatically reduce tuning time by guiding clients to turn into doze mode while waiting for the data to arrive. *B^+-Tree Distributed Index Scheme (BTD)* is a popular index scheme for wireless data broadcast, which has been extended by many research works. Among traditional index structures, alphabetic Huffman tree is another important tree-based index technique with the advantage of taking data's access frequency into consideration. In this paper, we are trying to answer one question: can alphabetic Huffman tree replace B^+-tree and provide better performance? To answer this question, we propose a novel *Huffman-Tree based Distributed Index Scheme (HTD)* and compare its performance with *BTD* based on a uniform communication environment. The performances of *HTD* and *BTD* are analyzed both theoretically and empirically. With the analysis result, we conclude that *HTD* outperforms *BTD* and can replace *BTD* in most existing wireless data broadcast system.

1 Introduction

Wireless Data Broadcast has attracted great attention recently in wireless computing area because of its scalability and flexibility to broadcast public information to a large number of mobile subscribers. In a typical data broadcast system, base stations broadcast a set of data periodically within its region. Mobile clients within the region could tune into broadcast channel, search for the data it needs, wait till target data items are broadcasted and download them. Considering that mobile devices has limited battery power and restricted lifetime, *access latency* and *tuning time* becomes two main criteria to evaluate the performance of a data broadcast system. According to the architectural enhancements, each mobile device has two modes: *active mode* and *doze mode*. The energy consumed

* This work is supported by NSF grant CCF-0829993 and CCF-0514796.

in active mode can be up to 100 times higher than that in doze mode. Based on this, access latency is defined to denote the whole time interval from the moment when a client initiates a query, till the moment it finishes downloading the data item, which evaluates the query time efficiency of a system; while tuning time is defined as the sum of time when a client keeps "active" during the process, which evaluates the energy efficiency of the system.

Researchers apply index technologies to reduce tuning time for a data broadcast system. An index is a specific data structure storing the location information of data items. Due to the nature of data broadcast scheme, indices in data broadcast system store the "time offset" of target data items. Once a client gets this offset, it is aware of the waiting time for the target data item to arrive on the broadcast channel. The client turns into doze mode to save some energy and tunes back to the broadcast channel right before the data item appears. Different index technologies have different searching efficiency. If we insert indices between data items, then the whole size of a program will increase, resulting a longer access latency. Therefore, discussing about an index technology, researchers will always consider the balance between tuning time and access latency.

B^+-Tree Distributed Index Scheme (BTD) is a popular index scheme for wireless data broadcast. Many other research works [3,4,5,18] have extended *BTD* with respect to different system configurations. Since the idea of distributed index can be generally adopted on tree-based search index methods, we naturally wonder what impact the choice of different search tree structures will have on the performance of broadcast system. Alphabetic Huffman tree is another primary tree-based index techniques. Compared with B^+-tree, it can not only guide searching of target data item, but also take into consideration the access frequencies of different data items. The higher access frequency a data item has, the closer it is to the root in an alphabetic Huffman tree. This can be a very beneficial feature for wireless data broadcast because it may reduce the time needed to search for more frequently requested data items and consequently reduce the average access latency. The purpose of this paper is to construct an Alphabetical *Huffman-tree based Distributed Index Scheme (HTD)* and evaluate the performance of distributed index schemes *BTD* and *HTD*.

For fair comparison, we build up a uniform environment with same communication model and data set. We assume each data item can have different size and different access probability, such that our mathematical model can be more practical and more accurate. Since system performance in skew broadcast heavily relies on data schedule algorithm/design, and we just want to compare the performance of indices, flat broadcast is adopted in this paper. We choose single channel data broadcast model to eliminate the impact of index and data allocation algorithms on the performances so that the difference in performances can be purely from different tree-based index structures. In order to perform the evaluation, we first develop a novel *Huffman-Tree based Distributed index scheme (HTD)*. We also adjust the packet design of *BTD* to fit our communication model. Based on the uniform system setup, a detailed theoretical analysis is performed on the expected access latency and tuning time of *BTD* and *HTD*.

Finally, we simulate the broadcast environment and provide mass numerical simulation. The theoretical and empirical analysis proves the superiority of Alphabetic Huffman tree based distributed index. We are the first work to construct a Huffman-tree based Distributed Index Scheme for wireless data broadcast problem.

The rest of this paper is organized as follows: in Section 2 we study previous literatures for wireless data broadcast problem, including various index technologies in different communication environments. In section 3 we illustrate our system model, and discuss broadcast environment, data type, and data schedule in detail. In Section 4 and 5 we construct and evaluate distributed index and Huffman tree correspondingly. Next, in Section 6 we illuminate the process of simulation and discuss index performance based on our numerical experiments. Finally, Section 7 gives conclusion and the plan of our next stage work.

2 Related Works

In wireless data broadcast area, the main research topics always focus on how to design index structures and how to allocate data on channels. Their purpose is to reduce access latency and tuning time, in order to improve the energy efficiency of the system. Many research works deal with data scheduling problem so as to decrease access latency. Acharya et al. [1] proposed "broadcast disk", which allocates data with similar access frequencies onto different disks and broadcast data of these disks repeatedly according to their frequencies, in order to cope with nonuniform access distribution. Vaidya et al. [16] discussed optimization issue with respect to the average access latency when data access distribution is nonuniform. Vlajic et al. [17] presented an optimized data broadcast strategy in hierarchical cellular organization system. However, none of these works implements indexing technique. Moreover, without doze mode, the tuning time is as long as access latency, which leads to high power consumption of mobile devices.

There are also many works converting traditional disk-based indexing approaches to air indexing by converting physical address into time offset. One paper [12] discussed a signature based approach for information filtering in wireless data broadcast. Another work by Xu et al. [19] gave an idea of exponential index that shares links in different search trees and allows clients to start searching at any index node. However, their approach may not perform well under nonuniform access probabilities. Yao et al. [22] proposed *MHash* to facilitate skewed data access probabilities and reduce access latency. Imielinski et al. [8] presented the flexible index and hash based index. Furthermore, they customized B$^+$-tree index and proposed (1, m) index as well as distributed index (BTD) [9]. BTD was extended by many other researchers to fit different system requirements. Hu et al. [5] designed a hybrid index scheme combining BTD and signature-based index. [18] proposed an index allocation method named TMBT for multi-channel data broadcast, which creates a virtual BTD for each data channel and multiplexes them on the index channel. Hsu et al. [4] modified *BTD* to deal with data with nonuniform access frequencies. In [3], Gao et al. built a complete multi-channel broadcast system to broadcast a data set with nonuniform access probability and data sizes, which used *BTD* as their index scheme.

Huffman-tree is a skewed index tree which takes into account the access probability of each data item, that the popular data has a shorter path from the root of the tree, thus it minimizes the average tuning time [2,14]. In [2], the proposed algorithms for constructing the skewed Huffman tree have a problem that the clients may fail to find the desired data item by traversing that Huffman tree. In [14], it discussed the construction of Huffman tree, which is similar to that of Huffman code, but the constructed Huffman tree has the same problem. There is another kind of Huffman tree, Alphabetic Huffman tree, proposed in [6], which serves as a binary search tree. It is further extended to k-ary search tree in [14], so that a tree node will fit in a wireless packet of any size by adjusting the fanout of the tree. However, all the above works discussed Huffman-tree on multi-channel environment. When it comes to the multi-channel data broadcast, how to allocate index and data will produce heavy impact on the performance of each index technique. A certain allocation method could be helpful to specific index structure, but at the same time it might reduce the efficiency of another index method. In this paper, we aim at comparing two commonly used index approaches under the same conditions, as well as minimizing both average access latency and average tuning time, so we adopt single channel data broadcast environment to avoid all kinds of influences introduced by a multiplicity of different multi-channel allocation methods. Unfortunately, there is no existing research applying alphabetic Huffman-tree onto single channel data broadcast systems.

3 System Symbols and Bucket Design

Now we present the system model of a wireless data broadcast communication environment for future comparison of tree-based distributed index schemes.

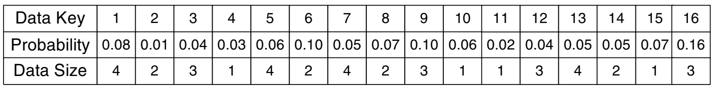
Table 1. Symbol Description

Sym	Description	Sym	Description
D	Data set $D = \{d_1, \cdots, d_t\}$	D_i	Data block on $\mathbb{B}_i$
t	Number of data items	P_i	Probability for block $\mathbb{B}_i$
P	Probability set $P = \{p_1, \cdots, p_t\}$	$\triangle_i$	The i^{th} subtree at level l+1 on T
S	Length set $S = \{s_1, \cdots, s_t\}$	V_i	Distributed path for $\triangle_i$
T	An index tree	u_i	Length of index on $\mathbb{B}_i$ with average u
L	Level of T	v_i	Length of V_i on $\mathbb{B}_i$ with average v
k	Maximum branch number for T	x_i	Length of D_i on $\mathbb{B}_i$ with average x
l	Threshold to cut T	R	Total number of $\triangle_i$ on T
$\mathbb{B}$	One broadcast sequence on a channel	B_i^j	The j^{th} index at i^{th} level of T
$\mathbb{B}_i$	The i^{th} block on $\mathbb{B}$	d_i^j	The j^{th} bucket of data item d_i
$\lvert\cdot\rvert$	Cardinality of one set	$\lVert\cdot\rVert$	Length measured in data bucket unit

3.1 System Symbols

The Base Station broadcasts data set D on a single wireless broadcast channel, where the total number of data items is t, and $D = \{d_1, d_2, \cdots, d_t\}$. Without loss of generality, assume data items in D are arranged in a consecutively increasing order of their primary key values. The access probability for each data item d_i is p_i, where $\sum_{i=1}^{t} p_i = 1$, and P indicates the probability set of D. Data items may have different sizes due to various applications, Let s_i denote the size of d_i, and S denote the length set of D. Fig. 1 is an example data set with 16 data items, which will be continuously used as our data sample throughout the paper.

In order to reduce tuning time for mobile clients, some tree-based index strategies, for instance the B^+*-tree Index* scheme, are applied to the wireless data broadcasting system. We use T to denote the index tree for tree-based index strategies, and define k as the maximum number of branches for each node in T. L is the depth or height of T. In *distributed index* [9], T is "cut" at the l^{th} level. B_i^j denotes the j^{th} index at i^{th} level of T. Sec. 4 and 5 will give detailed design of two index strategies. Table 1 lists most of the symbols used in this paper. Some symbols will be introduced in later sections.



Data Key	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Probability	0.08	0.01	0.04	0.03	0.06	0.10	0.05	0.07	0.10	0.06	0.02	0.04	0.05	0.05	0.07	0.16
Data Size	4	2	3	1	4	2	4	2	3	1	1	3	4	2	1	3

Fig. 1. An Example of Data Set

3.2 Bucket and Pointer Design

A *bucket* is the minimum logical unit for data transmission in wireless data broadcast systems. Data buckets and index buckets have different structures and sizes. An index bucket contains a complete index node; while a data item can take several data buckets. Data item size s_i is measured by the number of data buckets it occupies. A bucket has two segments: **head** and **payload**. For both index bucket and data bucket, its **head** has the same elements:

bId: id of a bucket, in the format of (i, j, n). For a tree-based distributed index bucket, it indicates the n^{th} recurrence of index B_i^j. For a data bucket, it denotes the j^{th} bucket of d_i (denoted as d_i^j) with $n = s_i$.
bType: the type of this bucket. E.g., a tree-based distributed index strategy has three types of buckets, i.e. control index, search index and data.
bLength: the total length or size of this bucket.
bOffset: the offset to the next nearest control index.

The **payload** segments of a data bucket and an index bucket are different. In a data bucket, the payload stores data. A data item may take up several data buckets of same lengths. On the other hand, in an index bucket, the payload stores index information, such as pointers, which indicate the time offsets to some other index buckets. A pointer contains the following elements:

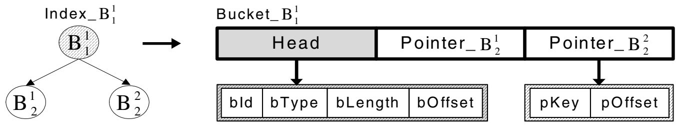


Fig. 2. An Example of Index Bucket Structure

pKey:	the **bId** of the bucket it points to.
pOffset:	time offset from current moment to the moment target bucket starts to broadcast.

For tree-based index strategies, an index bucket may contain several pointers, each pointing to one of its children. The number of pointers depends on the design of the index tree. Fig. 2 is an example index bucket storing an index node B_1^1 of a binary index tree, which has a head segment (the block in shadow) to "label" index B_1^1 itself, and a payload segment (two white blocks) to store the pointers of B_1^1. Since B_1^1 has two children B_2^1, and B_2^2, its payload segment should have two pointers, recording the location of B_2^1, and B_2^2.

4 B$^+$-Tree Based Distributed Index

We adopt B$^+$-Tree based distributed index strategy *BTD* introduced in [9]. To fit our model, we reformulate *BTD* with the bucket design in Sec. 3.2.

In *BTD*, B$^+$-Tree index is streamed on broadcast channel in *depth-first* manner, and it is "cut" at level l. Nodes from level 1 to l are *replicated part*, while others are *non-replicated part* which can be viewed as a number of subtrees rooted at the indices at level $l + 1$. Each index in the replicated part is a *control index* and has a *control table* to specify the search ranges of different subtrees.

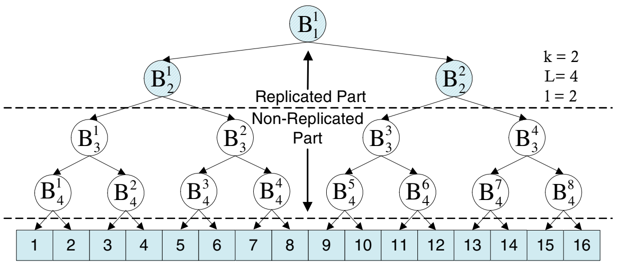


Fig. 3. An Example of B^+-tree cut at the 2^{nd} level

Fig. 3 is an example of a full binary B$^+$-Tree based distributed index structure with $k = 2$, $L = 4$, and $l = 2$. There are 16 data items in the data set D, represented by grey blocks at the bottom. Each index node B_i^j means the j^{th} index node on the i^{th} level of the tree. All the nodes above (including) the 2^{nd}

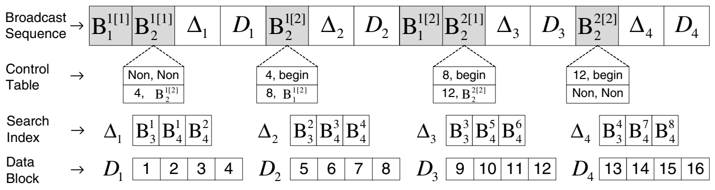


Fig. 4. An Example of $\mathbb{B}$ with Control Tables

level of the tree are control indices of the replicated part, while the other nodes below are search indices of the non-replicated part.

We traverse T according to distributed rules described in [9], and then append control table for each control index. During traversing process, data buckets and index buckets are interleaved on the same broadcast channel. $\mathbb{B}$ is a complete program streamed on a broadcast channel, including both index buckets and data buckets. Fig. 4 is an example of $\mathbb{B}$ with respect to the aforementioned index tree example in Fig. 3. $B_i^{j[1]}, \cdots, B_i^{j[k]}$ represent k appearances of B_i^j, where k is the same as branch number k in T.

Note that index bucket and data bucket may have different sizes. A data bucket is usually measured by KB. Each data bucket has size 1KB. However, the size of an index bucket is determined by the information stored in an index bucket. Therefore, we let $|\mathbb{B}|$ denote the cardinality of set $\mathbb{B}$, measured by the number of bucket, and $\|\mathbb{B}\|$ denote the total length of set $\mathbb{B}$, measured in the unit of one data bucket (KB). An index bucket may have different size from a data bucket, so we define "r" to indicate the ratio of data bucket size to index bucket size, i.e. $r = \textit{data bucket size/index bucket size}$. For instance, using the data set in Fig. 1 as an example, in Fig. 4 we have $|\mathbb{B}| = 34$, and $\|\mathbb{B}\| = 18/r + 40$, since there are totally 18 indices, 6 of which are control indices and the rest 12 are search indices. Additionally, *control tables* are used to specify the ranges of subtrees. For example, in the control table of $B_2^{1[2]}$, the first entry [4,begin] means that if the client is looking for a data item with key value ≤ 4, it needs to wait till the beginning of next broadcast cycle. The second entry [8, $B_1^{1[2]}$] implies if the client is looking for a data item with key value > 8, it should wait till $B_1^{1[2]}$ arrives. This control table indicates that the subtree immediately following it can only guide to data with key values in the range (4,8].

In *BTD*, index and data are interleaved on the same broadcast channel. As in Fig. 4, $\triangle_i$ denotes subtree in the non-replicated part, and is consist of search indices. For instance, $\triangle_2$ is the subtree rooted at B_3^2, with two children B_4^3 and B_4^4. D_i indicates the data buckets that $\triangle_i$ guides to, which is streamed sequentially by their key values. $\text{DFT}(\triangle_i)$ is the depth-first traversal of $\triangle_i$. For example, $\text{DFT}(\triangle_2) = B_3^2, B_4^3, B_4^4$. $\text{PATH}(B_i^j)$ is a path from root B_1^1 to node B_i^j (excluding the endpoint B_i^j), and V_i is a *distributed path* before each $\triangle_i$. For example, from Fig. 3 we can see that the distributed path for B_3^3 should be $V_3 = \{B_1^1, B_2^2\}$. After this, the broadcast sequence is defined as $\mathbb{B} = \{V_1, \text{DFT}(\triangle_1), D_1, V_2, \text{DFT}(\triangle_2), D_2, \cdots, V_R, \text{DFT}(\triangle_R), D_R\}$.

4.1 Performance Analysis of B$^+$-Tree Distributed Index

In this section, we analyze the performance of B$^+$-Tree based distributed index with respect to the expectation of *access latency* and *tuning time*.

Let's consider access latency first. Let R denote the number of subtrees after we cut T. The whole broadcast cycle is divided into $\mathbb{B}_1, \cdots, \mathbb{B}_R$ blocks, where $\mathbb{B}_i = \{V_i, \text{DFT}(\triangle_i), D_i\}$, for $1 \le i \le R$. P_i represents the access probability for block $\mathbb{B}_i$, which can be derived by summing up the probability of all data buckets that belong to data block D_i of $\mathbb{B}_i$, i.e. $P_i = \sum_{j \in D_i} p_j, for\ i = 1, \cdots, R$. Let v denote the average length of V_i, u the average length of $V_i + \triangle_i$, and x the average length of D_i. Note that u, v, and x are measured by data bucket(KB), while u_i, v_i, and x_i denote corresponding lengths for specific block $\mathbb{B}_i$. Hence, we have $u = (\|\mathbb{B}\| - \|D\|)/R$, $v = \sum_{i=1}^{R} \|V_i\|/R$, and $x = \|D\|R$.

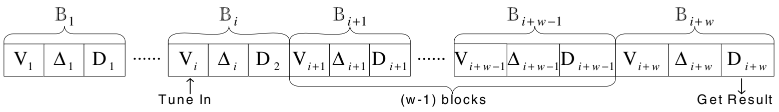


Fig. 5. An Example of a Client Searching for Data

Theorem 1. *If B^+-tree based distributed index and data are interleaved on one broadcast channel, then the average access latency is*

$$E(AL) = \frac{1}{R} \cdot \sum_{i=1}^{R} \Big(\sum_{w=1}^{R-2} ((\frac{1}{2} + w)u + wx) \cdot P_{(i+w)\%R} + (u - \frac{v}{2} + \frac{x}{2}) \cdot \frac{v}{u+x} \cdot P_i + (\frac{u-v}{2} + w(u+x)) \cdot P_i \cdot \frac{u-v+x}{u+x} \Big). \tag{1}$$

Proof. Assume that a client would like to retrieve data item d_j. It first tunes into the broadcast channel at block $\mathbb{B}_i$, and then waits for another w blocks to reach the block which contains the required datum d_j at $\mathbb{B}_{i+w}$. Next, the client waits for the first data bucket of d_j to come and begins to download, until it gets all the data buckets of d_j. Illustration of the whole process is shown in Fig. 5. There are three possible cases with respect to the length w:

Case 1: $\mathbf{1 \le w < R}$. We divide this case into 3 phases:

1. the client tunes into block $\mathbb{B}_i$. It takes an average $(u + x)/2$ time to visit $\mathbb{B}_i$;
2. the client waits $(w - 1)$ complete blocks, which takes $(w - 1)(u + x)$ time;
3. it finds the index directly pointing to d_j in $\triangle_{i+w}$ and download data. The average waiting time is $u + x/2$.

The expected access latency of this case is shown below, where b denotes the current block number, and d means the distance from b to the block it gets data:

$$E(AL|b = i, d = w) = \frac{u+x}{2} + (w-1)(u+x) + u + \frac{x}{2} = (\frac{1}{2} + w)u + wx$$

Case 2: $\mathbf{w} = \mathbf{0}$. The client tunes into V_i of block $\mathbb{B}_i$, and finds the pointer to required data item which is indeed in the following $\triangle_i$ of the same block $\mathbb{B}_i$. In this case, it takes only aforementioned phases 1) and 3), so we have:

$$E(AL|b = i, d = 0) = \frac{v}{2} + u - v + \frac{x}{2} = u - \frac{v}{2} + \frac{x}{2}$$

Case 3: $\mathbf{w} = \mathbf{R}$. Suppose the client tunes into block $\mathbb{B}_i$, and the required data is also in this block $\mathbb{B}_i$. Unfortunately, the client already missed the control index of this block when it tunes in, so it has to wait for the next control index in the next block to continue searching, and then wait for $\mathbb{B}_i$ to be broadcast again in the next *bcast*. In this case, the expected access latency becomes:

$$E(AL|b = i, d = R) = \frac{u - v + x}{2} + (w - 1)(u + x) + u + \frac{x}{2} = \frac{u - v}{2} + w(u + x)$$

According to the law of total expectation and the above three cases, we can get the average access latency as in Thm. 1.

Theorem 2. *The average tuning time for B^+-Tree based distributed index is*

$$E(TT) = \sum_{i=1}^{R} \frac{3u_i - v_i + (2+r)x_i}{r\|\mathbb{B}\|} + \frac{2L - l}{2} + \sum_{i=1}^{|D|} s_i p_i \tag{2}$$

Proof. The tuning time of searching and downloading one data item comprises the following phases:

Step I: The client tunes into broadcast channel, and searches for the right *control index*. Since the client can start searching only from a control index, we analyze this phase by three cases:

Case 1: *The first visited bucket is a control index.* Then the client could follow the control table to find the right control index in one more step, which is discussed in [3]. The probability of this case is $\sum_{i=1}^{R} v_i/\|\mathbb{B}\|$, and the average tuning time of this case is $\frac{2}{r}\sum_{i=1}^{R} v_i/\|\mathbb{B}\|$.

Case 2: *The first visited bucket is a search index.* The client needs to wait for the next nearest control index, and follow its control table to reach the target control index. This has a probability of $\sum_{i=1}^{R}(u_i - v_i)/\|\mathbb{B}\|$, and average tuning time is $\frac{3}{r}\sum_{i=1}^{R}(u_i - v_i)/\|\mathbb{B}\|$.

Case 3: *The first visited bucket is a data bucket.* The client also need to wait for the next control index, and then go to the target control index, with probability $\sum_{i=1}^{R} x_i/\|\mathbb{B}\|$. The average tuning time is $(1 + \frac{2}{r})\sum_{i=1}^{R} x_i/\|\mathbb{B}\|$.

Step II: The client searches for the index that directly points to the required data. The average number of visited index bucket in this step is $\frac{1}{r}\left(\frac{l}{2} + (L - l)\right) = \frac{1}{r}(L - \frac{l}{2})$.

Step III: The client sleeps until the required data appears, and then tunes in again to download data. The average downloading time is $\sum_{i=1}^{|D|} s_i p_i$.

Combining above steps, we have the average tuning time as in Thm. 2.

In order to get the actual values of the average access latency and average tuning time, we need to know the values of L, R, $\mathbb{B}$, u, v, and x. The total level L of an index tree is determined by the number of branches k of T and the size t of data set D. Since the total number of pointers at the bottom level of T should be equal to the number of data items, the number of leaf nodes on T should be at least $\lceil t/k \rceil$, and the number of nodes at the second lowest level of T should be at least $\lceil \lceil t/k \rceil / k \rceil$. In this way, we can calculate the size of each level inductively, until we reach the root of T. We define $N(L)$ as the set of nodes at the L^{th} level of T. There is an algorithm in [3] describing how to compute L and $N(L)$. With known L and $|N(L)|$, we can get $R = |N(l+1)|$. What's more, if T is a full k-ary tree, there is a theorem as follows.

Theorem 3. *If T is a full k-ary tree, then the total number of index buckets in a bcast is $\frac{k-k^L}{1-k} + k^l$, where $L = \lceil \log_k t \rceil$, $l < L$, and k, l are fixed parameters.*

The detailed proof of this theorem could be found in [3]. We can also get the value of other variables after the construction of $\mathbb{B}$.

5 Huffman-Tree Based Distributed Index

Huffman-tree index has been applied to the wireless broadcast environment ever since the last decades. It is an efficient index technique because it takes into account the access probability of data items when constructing the Huffman-tree. The popular data with higher probability reside closer to the root in Huffman-tree, which reduces search time when traversing from the root. Considering flat broadcast, we found that the distributed method could be extended to Huffman-tree based broadcast, which is an innovative idea that has not been considered before. In this section, we will discuss the construction of *Huffman-Tree based Distributed Index Scheme* (*HTD*) and perform a theoretical analysis on its efficiency.

The structure of index bucket and data bucket in *HTD* is almost the same as in *BTD*. The first step is to construct a k-ary Alphabetic Huffman-Tree following the methods introduced in [14]. Here we give an example of the construction process based on the data set and access frequency in Fig. 6. Note that if we normalize the access frequency in Fig. 6, we will get the same data set as in Fig. 1. The Alphabetic Huffman-Tree construction is shown in Fig. 7 and 8.

Stage 1: choose data nodes d_i, d_j as candidates to be merged when

1. there are no leaves between them,
2. the sum of their frequencies is the minimum,
3. d_i and d_j are the leftmost nodes among all candidates.

If the above conditions hold, we create a new index node d'_i with frequency equal to the sum of d_i's and d_j's frequencies, and replace d_i, d_j with d'_i in the construction sequence. This stage produces a tree T_0 without alphabetic ordering of the data nodes, as in Fig. 7, where we record the frequencies of each index node inside the circle as index key values.

Stage 2: record the level of each data node (leaf node) of T_0, denoted as L_i of data d_i. The root node level is 1. From bottom to the root, rearrange pointers such that for each level the leftmost two nodes have the same parent, and then the next two, and so on. We can generate an alphabetic Huffman-Tree T in this way, without changing the level of each node in T_0, as shown in Fig. 8.

We could easily extend this algorithm to construct k-ary Huffman-Tree, by merging at most k nodes in stage 1, and combining up to k nodes with the same parent in stage 2.

Data Item Key	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Frequency	23	4	12	10	17	31	15	21	29	19	7	12	16	14	20	48

Fig. 6. An example Data Set of Huffman-Tree based Distributed Index

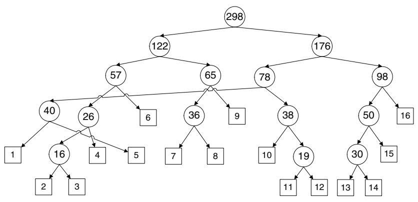


Fig. 7. The first step of constructing T_0

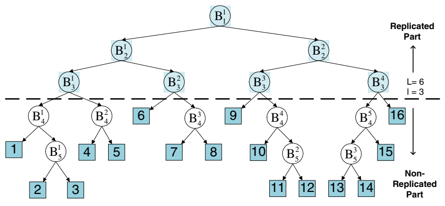


Fig. 8. The final Huffman-Tree T

After generating the alphabetic Huffman tree T, we cut T at level l, and perform a distributed traversal as Sec. 4. The index nodes above l is still called control index, and index nodes below l is search index. We append control tables onto control index in the same way as Sec. 4.

Note that there are two major difference between Huffman tree and B$^+$-tree, that the position of leaf nodes and the subtree sizes below l are different. It is possible that there might be data items above l in a Huffman tree, depending on which level we choose for l, since data items are not restricted to reside at bottom level of Huffman tree (they could also appear in higher level), which is a major difference with B$^+$-tree. Another difference is that sizes of subtrees below l may vary a lot in Huffman tree, but for B$^+$-tree each subtree has similar size.

The final broadcast sequence $\mathbb{B}$ generated in this example is illustrated in Fig. 9; and the access protocol is described in Alg. 1. When searching in a control table, client firstly compares the key value of request data and that of the first entry in control table; if request key is less than or equal to the first key, it should wait until the root of next bcast; if request key is greater, the client will go on to compare with the next entry in control table, and turn to doze mode

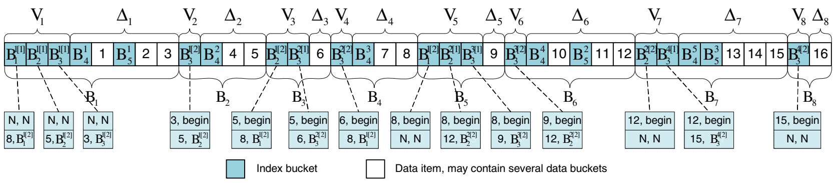


Fig. 9. The broadcast sequence of Huffman-Tree based Index

for *offset* time if request key is not greater than the next key in control table; otherwise, it will continue comparing with the rest entries of control table until it finds such an entry, or go to default bucket (the next index) if not found.

Algorithm 1. ***Retrieve Data***

```
    Input: $key_{req}$                        ▷ key value of request data $d_{req}$
    Output: $d_{req}$.
 1: Access randomly onto broadcast channel;
 2: $B_0$ = current bucket;
 3: if $B_0$ is data bucket & $B_0$.bId = $(key_{req}, 1, s_{req})$ then
 4:     Download data $d_{req}$;
 5: end if
 6: if $B_0$.bType $\neq$control then
 7:     Doze $B_0$.bOffset time till the next control index;
 8:     $B_0$ = current bucket;
 9: end if
10: Follow $B_0$'s control table and go to the pointed bucket;
11: Download data $d_{req}$;
```

5.1 Performance Analysis of Huffman-Tree Distributed Index

In this section, we analyze the system performance of Huffman-Tree based distributed index by evaluating its expected access latency and tuning time.

First let's consider access latency. Similar as *BTD*, all index and data buckets are interleaved on one broadcast channel. The whole broadcast cycle is divided into $\mathbb{B}_1, \cdots, \mathbb{B}_R$ blocks, where $\mathbb{B}_i = \{V_i, \text{DFT}(\triangle_i)\}$, for $1 \leq i \leq R$. Note that here V_i and $\triangle_i$ are different with those in *BTD*; V_i is still distributed path, but $\triangle_i$ may contain data. If there is no data node above l, V_i is the same as in *BTD*, and $\triangle_i$ is the whole subtree including data; however, if there is data node above l, V_i represents distributed path excluding data nodes, and $\triangle_i$ indicates all data items following V_i above l. We use P_i to represent the access probability for block $\mathbb{B}_i$, while P_i can be derived by summing up the probabilities of all data buckets that belong to $\mathbb{B}_i$, i.e. $P_i = \sum_{j \in B_i} p_j, for\ i = 1, \cdots, R$. Let v_i denote the length of V_i, and δ_i indicate the length of $\triangle_i$. Furthermore, an index bucket may have different size compared to a data bucket, so we continue to use "r" as the ratio of data bucket size to index bucket size.

Theorem 4. *If Huffman-Tree based distributed index and data are interleaved on one broadcast channel, then the average access latency is*

$$E(AL)=\frac{1}{\|\mathbb{B}\|}\sum_{i=1}^{R}(\sum_{w=1}^{R-2}(\frac{v_i+\delta_i}{2}+\sum_{j=i+1}^{i+w-1}(v_j+\delta_j)+v_{i+w}+\frac{\delta_{i+w}}{2})P_{(i+w)\%R}(v_i+\delta_i)$$

$$+(\frac{v_i+\delta_i}{2})P_i v_i+\sum_{i=1}^{R}(v_i+\delta_i)P_i\delta_i). \tag{3}$$

Proof. Assume a client want to get data item d_j. It first tunes into the broadcast channel at block $\mathbb{B}_i$. Then, it waits for another w blocks to reach the index which contains the pointer to d_j at $\mathbb{B}_{i+w}$. Within $\mathbb{B}_{i+w}$, the client waits for the first data bucket of d_j to be broadcast and begins to download, until it gets all the data buckets of d_j. There are three possibilities about the length of w:

Case 1: $\mathbf{1 \leq w < R}$. We can divide this case into three phases: 1) the client tunes into block $\mathbb{B}_i$, and takes an average $(v_i+\delta_i)/2$ time in it; 2) it waits through $(w-1)$ complete blocks, which takes $\sum_{j=i+1}^{i+w-1}(v_j+\delta_j)$ time; and 3) it finds the pointer to the datum in $\triangle_{i+w}$, and then download data, so the average waiting time is $v_{i+w}+\delta_{i+w}/2$. The expected access latency of this case:

$$E(AL|b=i,d=w)=\frac{v_i+\delta_i}{2}+\sum_{j=i+1}^{i+w-1}(v_j+\delta_j)+v_{i+w}+\frac{\delta_{i+w}}{2} \tag{4}$$

Case 2: $\mathbf{w = 0}$. The client tunes into V_i of block $\mathbb{B}_i$, and the pointer to required data is indeed in the following bucket of the same block $\mathbb{B}_i$. In this case, it only contains aforementioned phases 1) and 3) of the first case, so the expected access latency becomes:

$$E(AL|b=i,d=0)=\frac{v_i}{2}+\frac{\delta_i}{2}=\frac{v_i+\delta_i}{2} \tag{5}$$

Case 3: $\mathbf{w = R}$. Suppose the client tunes into block $\mathbb{B}_i$, and the required data is in the same block $\mathbb{B}_i$. Unfortunately, the client already missed the index buckets, so it has to wait for the next available index in the next block to continue searching, and then wait for $\mathbb{B}_i$ to be broadcast again in the next broadcast cycle. In this case, the expected access latency is:

$$E(AL|b=i,d=R)=\frac{\delta_i}{2}+\sum_{j=i+1}^{i+w-1}(v_j+\delta_j)+v_i+\frac{\delta_i}{2}=\sum_{i=1}^{R}(v_i+\delta_i) \tag{6}$$

Combining equation (4), (5), (6), using law of total expectation, we can get the average access latency as in Thm. 4.

Theorem 5. *The average tuning time for alphabetic Huffman-Tree based distributed index scheme is*

$$E(TT)=\frac{2\sum_{i=1}^{R}v_i+(2+r)|D|+3\sum_{i=1}^{R}\delta_i}{r\|\mathbb{B}\|}+\sum_{i=1}^{|D|}(\frac{l}{2r}+\frac{1}{r}(L_i-l)+s_i)p_i \tag{7}$$

Proof. The tuning time of searching and downloading one data item comprises the following steps:

Step I: The client tunes into broadcast channel, and search for the right index, following which it can get the required data on that same block. Consider these three cases:

Case 1: *The client first tunes into a control index.* Then the client could follow the control table to find the right control index in one more step, which is discussed in [3]. The probability of this case is $\sum_{i=1}^{R} v_i/\|\mathbb{B}\|$, and the average tuning time of this case is $\frac{2}{r}\sum_{i=1}^{R} v_i/\|\mathbb{B}\|$.

Case 2: *The first visited bucket is a data bucket.* The client need to wait for the next nearest control index, and then go to the target control index, with a probability of $|D|/\|\mathbb{B}\|$. Thus, the average tuning time of this case is $(1+\frac{2}{r})|D|/\|\mathbb{B}\|$.

Case 3: *The first visited bucket is a search index.* The client also need to wait for the next nearest control index, and follow its control table to reach the target control index. This has a probability of $\sum_{i=1}^{R} \delta_i/\|\mathbb{B}\|$, and average tuning time is $\frac{3}{r}\sum_{i=1}^{R} \delta_i/\|\mathbb{B}\|$.

Step II: Next, the client searches for the pointer that directly points to the required data. Then it sleeps until the required data appears, and tunes in again to download data. The average time of this step is $\sum_{i=1}^{|D|}(\frac{l}{2r}+\frac{1}{r}(L_i-l)+s_i)p_i$, where L_i is the level of data d_i in the Huffman-Tree.

Finally, summarizing the above steps, we can get the average tuning time of Huffman-Tree based distributed index as in Thm. 5.

6 Simulation

In this section, we use simulation results to evaluate the performance of *HTD* and *BTD*. Our system is implemented using Java 1.6.0 on an Intel(R) Xeon(R) E5520 computer with 6GB memory, and Windows 7 v6.1 operating system.

6.1 Simulation Settings

We simulate a base station with single broadcast channel, broadcasting a database with 10,000 data items [20], each of which has different sizes from 1KB to 4KB, and multiple clients requesting various sets of data items. The access probability of each data item satisfies the zipf distribution [13], a model for non-uniform access patterns [10,18]. Each data bucket is set of size 1KB, and we could calculate the size of each index bucket as [3] to be 0.1KB. Normally, in real applications, the ratio of index bucket size over data bucket size is $\frac{1}{r}=0.1$, but other papers never discuss about this. They assume index bucket is of the same size as data bucket, which is not accurate. Therefore, in our simulation we set up $r=10$, which is much closer to the reality scenario. Moreover, for each group of experiments, we generate 10,000 requests based on data access probabilities, in order to calculate the average access latency (AAL) and average tuning time (ATT) during data retrieval more accurately.

6.2 Simulation Results

When the size of the request set is 10,000, the ratio of index bucket size over data bucket size $\frac{1}{r} = 0.1$, we vary the size of database to compare AAL and ATT of *HTD* and *BTD*. From Fig. 10 we can see that *HTD* has much shorter average access latency than *BTD*, while both *HTD*'s and *BTD*'s average access latencies gradually increases as the database size increases. Note that the measurement of Y-axis denotes the unit time to read one data bucket. Thus, we can claim that *HTD* reduces the average response time of request retrievals. Moreover, as the database size increasing, the average access latency gap between *HTD* and *BTD* is growing larger, and the advantage of *HTD* becomes more obvious. From Fig. 11 we can see that no matter how database size increased, *HTD* always needs less average tuning time than *BTD* during retrieval, which means *HTD* is more energy-efficient. As the database size increasing, the average tuning times gap between *HTD* and *BTD* is growing larger, which implies that the energy advantage of *HTD* is getting more obvious.

Next, we evaluate the broadcast cycle length of *HTD* and *BTD*. Due to different tree structures, these two approaches have different Bcast length after index and data allocation, although the distributed traversal methods are similar. We use $||\mathbb{B}||$ to represent the length of one Bcast on broadcast channel. We consider the bucket size ratio r when analyzing the length of Bcast. As in Fig. 12, the Bcast of *BTD* is always longer than that of *HTD*. The reason is that *BTD* has much more index nodes than *HTD* due to its tree structure, even when they use the same data set and same cutting level. Therefore, using *HTD* for data broadcast will reduce the total length of data stream on broadcast channel.

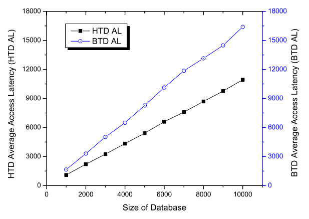


Fig. 10. AAL w. r. t. $|D|$

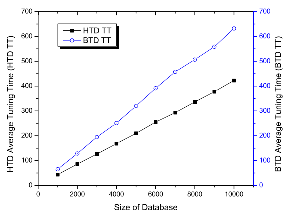


Fig. 11. ATT w. r. t. $|D|$

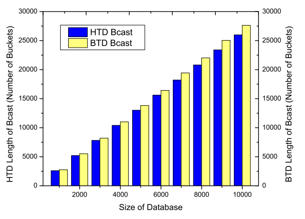


Fig. 12. $||\mathbb{B}||$ w. r. t. $|D|$

7 Conclusion

In conclusion, we are the first work to propose a promising strategy under wireless data broadcast environment, which is a novel Huffman-Tree index scheme combined with distributed index strategy. Specifically, we formally define an uniform communication environment, redesign and enhance *B^+-tree distributed index* (*BTD*) structure and broadcasting scheme, propose a novel *Huffman-tree based distributed index* (*HTD*), theoretically analyze each scheme under the same environment and same criteria, and then evaluate the performance of them by experiments. Simulation results show two major advantages of *HTD*: (1) it is more energy efficient, and (2) it reduces response time significantly. Therefore, *HTD* outperforms *BTD* in all major criteria.

All in all, our contribution includes three aspects. Firstly, we construct the uniform communication environment for wireless data broadcast system, and provide structured design of distributed index and Huffman-tree index. We follow the latest and most efficient construction for both index technologies, and redesign/modify some part of them such that they could be applied in the uniform communication environment. Next, we provide a general theoretical analysis to evaluate the performance of each index. Such analysis can be applied easily to majority indices commonly used in data broadcast. It can be a standard to evaluate the efficiency of an index technique. Thirdly, we simulate data broadcast system with a large number of numerical experiments, using the same group of sample data, such that the output will be reliable. Simulation results reveals that Huffman-tree distributed index is more power efficient and also responses much faster. Our future work includes developing more efficient index schemes for wireless data broadcast and provide more theoretical analysis on them.

References

1. Acharya, S., Alonso, R., Franklin, M., Zdonik, S.: Broadcast disks: Data management for asymmetric communication environments, pp. 199–210 (1995)
2. Chen, M., Yu, P., Wu, K.: Indexed Sequential Data Broadcasting in Wireless Mobile Computing. In: ICDCS 1997 (1997)
3. 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, Submitted to Information Sciences (2010)
4. Hsu, C., Lee, G., Chen, A.: Index And Data Allocation On Multiple Broadcast Channels Considering Data Access Frequencies. In: MDM 2002, pp. 87–93 (2002)
5. Hu, Q., Lee, W., Lee, D.: A Hybrid Index Technique for Power Efficient Data Broadcast. Distrib. Parallel Dat. 9(2), 151–177 (2004)
6. Hu, T., Tucker, A.: Optimal Computer Search Trees and Variable-length Alphabetic Codes. SIAM J. Appl. Math. 21(4), 514–532 (1971)
7. Hurson, A., Muñoz-Avila, A., Orchowski, N., Shirazi, B., Jiao, Y.: Power-Aware Data Retrieval Protocols for Indexed Broadcast Parallel Channels. Pervasive and Mobile Computing 2(1), 85–107 (2006)
8. Imielinski, T., Viswanathan, S., Badrinath, B.: Power Efficient Filtering of Data on Air. In: Jarke, M., Bubenko, J., Jeffery, K. (eds.) EDBT 1994. LNCS, vol. 779, pp. 245–258. Springer, Heidelberg (1994)
9. Imielinski, T., Viswanathan, S., Badrinath, B.: Data on Air: Organization and Access. IEEE TKDE 9(3) (1997)
10. Jung, S., Lee, B., Pramanik, S.: A Tree-Structured Index Allocation Method with Replication over Multiple Broadcast Channels in Wireless Environment. TKDE 17(3) (2005)
11. Lee, W., Zheng, B.: A Fully Distributed Spatial Index for Wireless Data Broadcast. In: ICDE 2005, pp. 417–418 (2005)
12. Lee, W., Lee, D.: Using signature techniques for information filtering in wireless and mobile environments. Distrib. Parallel Dat. 4(3), 205–227 (1996)
13. Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (1999)

14. Shivakumar, N., Venkatasubramanian, S.: Energy-Efficient Indexing For Information Dissemination In Wireless Systems. ACM, Journal of Wireless and Nomadic Application (1996)
15. Vijayalakshmi, M., Kannan, A.: A Hashing Scheme for Multi-channel Wireless Broadcast. Journal of Computing and Information Technology-CIT 16 (2008)
16. Vaidya, N., Hameed, S.: Scheduling data broadcast in asymmetric communication environments. Wireless Networks 5, 171–182 (1996)
17. Vlajic, N., Charalambous, C., Makrakis, D.: Wireless data broadcast in systems of hierarchical cellular organization. In: ICC 2003, vol. 3, pp. 1863–1869 (2003)
18. Wang, S., Chen, H.: Tmbt: An Efficient Index Allocation Method for Multi-Channel Data Broadcast. In: AINAW 2007 (2007)
19. Xu, J., Lee, W., Tang, X., Gao, Q., Li, S.: An Error-Resilient and Tunable Distributed Indexing Scheme for Wireless Data Broadcast. IEEE TKDE 18(3), 392–404 (2006)
20. Yee, W., Navathe, S.: Efficient data access to multi-channel broadcast programs. In: CIKM 2003, pp. 153–160 (2003)
21. Yang, X., Bouguettaya, A.: Broadcast-based data access in wireless environments. In: Jensen, C.S., Jeffery, K., Pokorný, J., Šaltenis, S., Hwang, J., Böhm, K., Jarke, M. (eds.) EDBT 2002. LNCS, vol. 2287, pp. 553–571. Springer, Heidelberg (2002)
22. Yao, Y., Tang, X., Lim, E., Sun, A.: An Energy-Efficient and Access Latency Optimized Indexing Scheme for Wireless Data Broadcast. IEEE TKDE 18(8), 1111–1124 (2006)
23. Zheng, B., Lee, W., Liu, P., Lee, D., Ding, X.: Tuning On-Air Signatures for Balancing Performance and Confidentiality. IEEE TKDE 21(12), 1783–1797 (2009)

Buffer Cache De-duplication for Query Dispatch in Replicated Databases

Takeshi Yamamuro, Yoshiharu Suga, Naoya Kotani,
Toshio Hitaka, and Masashi Yamamuro

NTT Cyber Space Laboratories
{yamamuro.takeshi,suga.yoshiharu,kotani.naoya,
hitaka.toshio,yamamuro.masashi}@lab.ntt.co.jp

Abstract. We propose a buffer cache de-duplication technique for query dispatch in replicated databases. In the field of replicated databases, there is the well-known problem called 'Buffer Cache Duplication' problem, which means that different buffer caches share some identical data. Unfortunately, existing approaches of de-duplication have shortcomings; the only SQL statements of queries (e.g. FROM and WHERE clauses) are insufficient to estimate exactly which data the queries reference for duplication-free dispatch. Our approach uses index access patterns to construct a look-up table that allows dispatchers to determine which database it should dispatch a query. We implement a prototype and demonstrate that under a certain condition around 90% of the duplication holds down to 12% in two databases, and it cuts down the referenced data on each buffer cache to approximately 40% in eight databases. Finally, we will discuss whether the condition can be applied to actual workloads.

Keywords: Replication, Query Dispatch, and De-duplication.

1 Introduction

There are several important Web services (e.g., Facebook and eBay) on the Internet, and they are central to our day-to-day lives. These and similar services can easily be launched by anyone based as cloud services nowadays. Typically these services consist of three layers; (1)Web servers, which interact with clients using HTTP, (2)Application servers, which execute internal logics, and (3)Database servers, which store and load data. As internal processing on (1) and (2) are independent between processes, these performances are easily improved by increasing servers (easy scalability). However, factor (3) makes it difficult to improve performances, and its penalties become an issue these days[1].

Various studies have been made to improve performances of database servers. Replication is an obvious approach, and there are many proposals[2,3,4]. Typically, queries are near-evenly distributed among replicated databases by load balancers, which follow some fixed strategies such as round-robin and least connections. Although useful, this approach suffers from the well-known problem

Table 1. Analysis of the duplication on two replicated databases

State	Ratio (%)
Duplicated	93.97
Non-duplicated	6.03

of 'Buffer Cache Duplication'[5,6]. The duplication means that same data are loaded on buffer caches on replicated databases. We assume a query 'A', which needs data 'X'. If query 'A' is dispatched to two replicate databases[1], and then common data 'X' lie in buffer caches of databases. Table 1 shows an actual duplication ratio for two databases that use round-robin techniques and DBT-1 for $3,600$ seconds in an advance evaluation. DBT-1 is a subset of TPC-W plotted out by Transaction Processing Performance Council[7], and a benchmark for web-based workloads. In the evaluation, we define the duplication as the intersection of working sets[2]. As a result, we found that the most of data on the buffer cache are duplicated.

In terms of restriction of actual or virtualized server's resources, it happens that buffer caches are too short to have frequently accessed data. In this scenario, the duplication leads to problems of database performances. De-duplication decreases working sets on each buffer cache, and a hit ratio of the buffer cache could increases. And then, it leads to improvement of response time and throughputs. Even though performances do not deteriorate, reducing the necessary amount of a buffer cache leads to saving resources of each server.

In previous studies, the duplication can be solved by two approaches: query-based[5] and relation-based[6]. The query-based one is that identical queries are transferred to same replicated databases on the assumption that required data have already been present in buffer caches. The relation-based one assigns batches of relations to databases, and queries are transferred to where required relations are. The report on the former approach indicates that performances fall as accessed data sizes increase[6]. The latter one has restriction of the type of workloads. We assume two databases ('A' and 'B'), and three relations ('X', 'Y', and 'Z'). Relation 'X' corresponds to database 'A', and relations 'Y' and 'Z' correspond to database 'B'. If most queries call relation 'X', databases loads become distorted.

We present a state-of-the-art query dispatch technique that overcomes the above problems and eliminate the duplication. Our approach focuses on index access patterns; more specifically the referenced counts of index leaves are periodically recorded as statistics to construct a look-up table for duplication-free query dispatch.

The contributions of this paper are the following.

1. An analysis of the duplication: we point out that it is difficult to dispatch queries without the duplication only based on its SQL statements.

[1] We assume any replicated database outputs same responses to same requests.
[2] We define working sets as referenced and identical parts of whole data in a database.

2. We propose a new query dispatch technique for de-duplication. We focus on index access patterns to estimate which data queries reference, and analyze data accessed simultaneously.
3. We implement its prototype using DBT-1, and show that under a certain condition the duplication decreases from around 90% to 12% in two databases. And also, working sets on each buffer cache is held down to approximately 40% in eight databases.
4. We model the duplication in buffer caches to analyze conditions, and finally show that our technique has effect on DBT-1 with the zipf-like distribution [11] close to actual workloads.

2 Problem Statement

2.1 Query Dispatch Problem

We assume that query sets Q is dispatched to N replicated databases in certain periods of time T. T means time spans (e.g. one day, one week, ...) in which same sets are input periodically. Then, query sets are defined as:

$$Q = \{q_t | t \in T\} \tag{1}$$

For query dispatch, queries in Q is transferred to one of N, and is defined as:

$$\gamma : Q \rightarrow 1, 2, 3, \cdots N \tag{2}$$

This paper uses the index access patterns of workloads to solve the duplication problem. Indexing is a common technique to quickly locate required information in databases, and is visualized as a tree composed of nodes and leaves. Database relations are formed by one or more blocks. A block is the unit of I/O, and occupies contiguous and fix-length data areas (e.g. 8KB in PostgreSQL). For example, one relation has K tuples, and one block is capable of holding two tuples. Then, the relation consists of $\lceil \frac{K}{2} \rceil$ blocks. A query triggers the search of blocks to locate the tuples needed to satisfy requests, and if the blocks are not in the buffer cache, it loads them. Therefore, in (2) our proposed technique classifies query sets Q, and minimizes working sets in each buffer cache.

2.2 Query Characteristics in the Web

Workloads in databases correspond to application programs (APPs) in the Web. In previous studies, it is typical to limit workloads, and have an effect in only those domains. In this paper, we assume the following workload characteristics.

1. Simple SQL statements which include only a few searched keys such as primary keys in each relation.
2. Most of data accesses in workloads is via indices.

Queries processed in databases from APPs in the Web tend to be short and simple; this feature was reported in a previous study[8]. From this viewpoint, our research considers only simple queries. As for the latter restriction, we take into account that most databases have at least two access strategies, sequential and indexed ones. The used strategy depends on access costs which are determined by database sizes, cardinality, and so on. However, as response time for these APPs becomes more important, the indexed access strategy is popular, and most of databases and queries tend to use its strategy compared to the sequential access strategy which is time-consuming. Accordingly, this paper assumes indexed-based queries. Because TPC-W used in the initial evaluation (Section 1) is a benchmark for APPs in the Web, response time makes a difference. Most of workloads in TPC-W consist of searches for products or authors, and to process these searches speedily the corresponding indices are used heavily. Therefore, TPC-W matches our assumption above.

2.3 Common Blocks between Queries

A naive approach to avoid the duplication and to solve the previous problems in the query-based and relation-based approach is to use tuples, which simply uses conditions in WHERE clauses with 'predefined and fixed' rules to dispatch queries. For example, there are 1000 tuples with id between 1 and 1000, and two databases ('A' and 'B'). The tuple-based approach is that dispatchers transfer queries for tuples with id between 1 and 500 to database 'A', and the others are dispatched to database 'B'. The tuple-based approach, however, suffers the same problem as the query-based approach, where increasing the amount of referenced data worsens performances. The cause is that there are referenced common blocks between queries. We assume that query 'A' needs blocks 'X' and 'Y', query 'B' needs blocks 'Y' and 'Z', and the common block is 'Y'. If the two queries are dispatched to deferent databases, block 'Y' is duplicated. In case that referenced blocks increase, the common blocks are central to the problem. Block information is managed by database internals and the naive approach is unable to judge whether blocks are common or not by means of only WHERE clauses. Table 2 shows the statistics of a duplication ratio in DBT-1 with the tuple-based strategy with fixed rules constructed to balance query loads. The average ratio is 87.51% and the standard deviation is 11.41%. It is clear that the tuple-based strategy has difficulty in eliminating the duplication. We think that its problem must be solved in some way.

Table 2. Duplication ratio of the tuple-based technique with fixed rule

Statistics	Value (%)
Average	87.51
Standard Deviation	11.41

3 Duplication-Free Query Dispatch

We introduce a state-of-the-art query dispatch technique that prevents the buffer cache duplication. At first, the technique estimates the common blocks between queries based on the index access patterns of workloads, that is, the index leaf counts of usage on buffer caches. Results are used to build a look-up table that can realize query dispatch without the duplication. The look-up table composes of keys in WHERE clauses of queries, and if keys of input queries are not in the look-up table, dispatchers transfer queries in the round-robin way. Our key idea is how to build the look-up table, and its procedure is briefly described below.

1. Referenced usage counts for index leaves on buffer caches are recorded once every T/g seconds, and these snapshots of referenced counts are continually collected g times.
2. A correlation between leaves is calculated, and correlated leaves are groupd. Section 3.1 describes the rational of why these leaves are grouped behind this approach.
3. If one relation has multiple indices, we need additional operations. From only index access statistics, we can't see whether these leaves reference same blocks, or not. Therefore, we build graphs, G, from reference relationship between leaves by means of its pointers to blocks, and apply a global min-cut algorithm, as is detailed in Section 3.2.
4. Finally, we classify grouped leaves into the number of databases so that the referenced number of leaves are evenly distributed, and then construct the look-up table that represents the rules of query dispatch.

Pseudo-code of its procedure above is shown in Algorithm 1.

3.1 The Look-Up Table for Query Dispatch

As explained in the beginning of Section 3, the look-up table for query dispatch is derived from g snapshots, which are composed of the referenced counts of the index leaves recorded every T/g seconds. Every T/g seconds, entries $\langle$ snapshot IDs, leaf IDs, and usage counts $\rangle$ are recorded for each leaf in buffer caches. The size of one entry is 13B, where each size of snapshot IDs, leaf IDs, and usage counts are 1B, 8B, and 4B, respectively. A total snapshot size is very small, because the area of loaded leaves on buffer caches is much lower than the area of data as shown in Table 3. For example, if T is $27,000$ seconds (7.5h), g is 300 seconds (5m), the size of buffer caches is 512MB, each size of leaves is 8KB, and the ratio of leaves is 2%, a total snapshot size is around 1.5MB. These recorded snapshots are input to Algorithm 1.

We focus on the correlations of the usage count of leaves to solve the common block problem, and explain this point of our observation in Fig.1. Typical database systems hold usage counts on each buffer cache content, mainly for buffer management. That is, when buffer caches are full, buffer managers pop some based on these usage counts. Therefore, every time contents are loaded

Algorithm 1. Generating the Look-Up Table

```
Require: A series of snapshots $S$, the number of databases $db\_num$, and threshold $th$
    of the global min-cut algorithm
Ensure: Classified key sets $K_1 \cdots K_{db\_num}$ for query dispatch
 1: $C \leftarrow 0$;
 2: for $i \leftarrow 1$ to $db\_num$ do
 3:   $K_i \leftarrow 0$;
 4: end for
 5: Calculate a correlated matrix $cr$ from $S$;
 6: while $cr$ has a positive element do
 7:   Get the positive pair of leaves, and update grouped sets $C$ to merge the pair;
 8:   Invalidate the pair;
 9: end while
10: if one relation has multiple indices then
11:   Build graphs $G$ by means of $C$;
12:   loop
13:     Try to classify $C$ to db_num, and calculate a maximum difference ratio $z$ from
        its average;
14:     if $z < th$ then
15:       break;
16:     end if
17:     Split a graph with maximum size in $G$, and update $C$;
18:   end loop
19: end if
20: while $C$ has a element do
21:   Get sets $C'$ with maximum size, and pick key sets $K'$in $C'$;
22:   Get database ID $t$ with its minimum amount;
23:   $K_t \leftarrow K_t \cap K'$;
24:   $C \leftarrow C - C'$;
25: end while
```

or referenced on buffer caches, counts are added up. Queries reference multiple leaves on buffer caches, and then the counts of two or more leaves are incremented simultaneously. From this observation, we assume that simultaneously-referenced leaves are correlated. These correlated leaves are grouped, and as a result the data areas that queries reference in common are linked.

In the case of Fig.1, query A accesses DATA6, DATA10, and DATA11 via LEAF3 and LEAF4, and query B accesses DATA11, DATA14, and DATA15 via LEAF4 and LEAF6. The counts of LEAF3 and LEAF4 are correlated since they are incremented simultaneously by query A, and LEAF3 and LEAF4 are grouped. In a similar way, LEAF4 and LEAF6 are grouped. Finally, LEAF3, LEAF4, and LEAF6 are placed in a same group. Fig.2 shows the actual correlations of the leaves of index 'item_pkey' on relation 'ITEM' and index 'author_pkey' on relation 'AUTHOR' in DBT-1. The peaked pairs are expected to be referenced simultaneously, and on closer examination, these leaves are accessed at a time in queries below.

Query(X)=SELECT * FROM ITEM, AUTHOR
WHERE i_id = 'X' AND i_a_id = a_id;

Relation 'ITEM' has i_id as a primary key with 'item_pkey' and i_a_id as a foreign key to relation 'AUTHOR', and relation 'AUTHOR' has a_id as a primary key with 'author_pkey'. As long as queries are processed via these indices, they are referenced simultaneously, and correlated.

The grouped correlated leaves above are apportioned among databases according to the first-fit rule (line 20-25)[9]. This rule is a heuristic technique for bin-packing problems, and packages are packed into bins whose amount of space is minimum. In the case of our approach, packages are regarded as grouped leaves, and capacities are regarded as the sum of usage counts. Finally, index keys included in these leaves are transformed into the table for query dispatch.

Table 3. The average ratio of contents of the buffer cache

Elements	Ratio (%)
NODE	0.02
LEAF	1.43
DATA	98.55

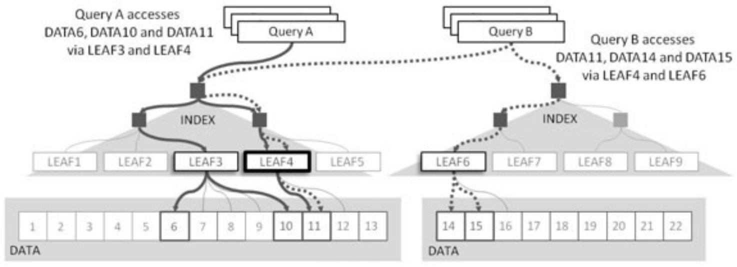


Fig. 1. Grouping based on the correlations between the leaves

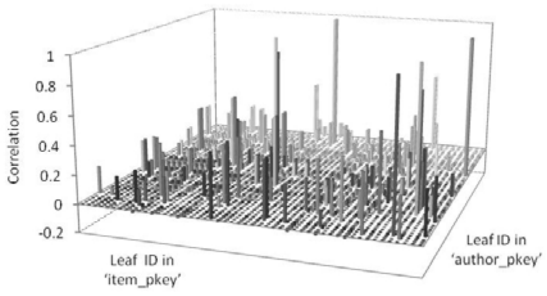


Fig. 2. Practical correlation in DBT-1

3.2 Exception: Multiple Indices in a Relation

One relation can include multiple indices. If queries access data via all these indices at one time, these usage counts in leaves might be correlated. However, if not, it is impossible to judge whether the leaves of multiple indices reference the same block, or not. Therefore, we present exceptional operations to avoid this case. We start with reference graphs G, where nodes are leaves grouped by correlations, and these nodes are connected by edges if common pointed blocks lie between them. If one relation has multiple indices (line 10) in Algorithm 1, it generates G by means of leaves groups (line 11), where G means disjoint graph sets. Grouped leaves are classified as a trial (line 13), if difference ratios from its averages exceed threshold th, largest graphs needs to be split by means of the global min-cut algorithm (line 17). The global min-cut algorithm is a technique to find the combinations of edges which have minimum cost. The well-known contraction algorithm[10] has complexity of $\theta(N^2 ln(N))$, where N is the number of nodes. The edge cost in our technique is regarded as the number of the common pointed blocks multiplied by the sum of usage counts. This process is looped until the difference ratio falls below th, or none of the edges could be cut (line 10-19).

4 Experimental

Our proposed technique was evaluated against DBT-1 which incorporated query dispatch functions. Three types of queries were implemented: ORDERING_MIX, SHOPPING_MIX, and BROWSING_MIX. The default one is SHOPPING_MIX, where the ratio of updating queries and referencing queries is 1 to 4. This query type is used in all experiments. In subsequent experiment, the number of items is $1,000,000$, the number of customers is $2,880,000$, eu/min is 800, and the average think_time is 7.2 (required settings in DBT-1). The threshold th in Algorithm 1 is set to 0.05, the number of snapshots g is 60 seconds, and execution time T is $3,600$ seconds.

The system configuration is that DBT-1 and each database run in individual servers. The database is a PostgreSQL v9.0beta4 with 512MB of shared memories and 'Streaming Replication', newly incorporated for replication in this version. All experiments were run on IBM x3550 servers with a dual-core 3.33 GHz Intel Xeon processor, 16GB of RAM, and 50GB SAS drives. These servers are connected by a L2 switch with 1000BASE-T in a rack.

The objectives of experiments are shown below.

– What parameters impact the duplication of buffer caches?
– How do working sets by queries change with the use of other techniques and the number of databases?

4.1 Performance Evaluation

There are many parameters in databases, but we found that FILLFACTOR, the ratio of tuples in one block, was a only indicator for the duplication. For example,

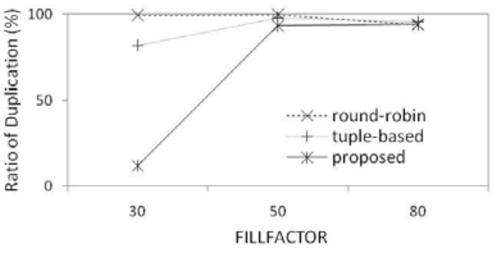


Fig. 3. The effect of de-duplication in buffer caches

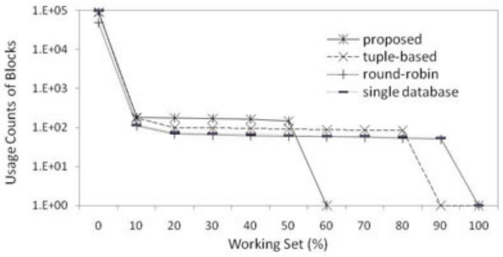


Fig. 4. The difference in the usage count of blocks

if FILLFACTOR=80, 80% of one block is filled with tuples, and the remaining space is reserved for updates. Thus database sizes increase as FILLFACTOR decreases. Fig.3 shows that the duplication ratio of the round-robin, the tuple-based, and the proposed technique for two databases with FILLFACTOR values of 30 ,50, and 80. Database sizes are 14GB, 7GB, and 4GB respectively. This result indicated that the round-robin technique yielded constant ratios regardless of conditions, while the tuple-based and proposed techniques allowed ratios to decrease as FILLFACTOR decreased. In particular, the proposed technique offered much lower ratios than the tuple-based technique, and if FILLFACTOR=30, the duplication was held down to around 12%. At FILLFACTOR=80, the three techniques had basically same effects on the duplication.

Fig.4 plots the distribution of the average working sets of these techniques with 30 of FILLFACTOR. As you can see, the proposed technique offers the lower working set ratio than the others. Moreover, the working set of the round-robin technique is exactly same as the set of a single database, and this result is same as noted in the previous study[6], where the working set of this technique hardly depends on the number of databases. Fig.5 shows that the working sets decrease corresponding to the increase of the number of databases. Finally, the average working sets of each database in eight databases were reduced by 40% compared to a single database. As the result, the decrease of working sets leads to de-duplication.

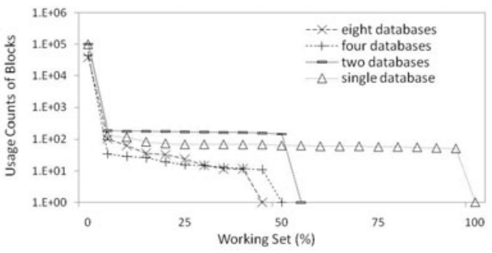


Fig. 5. The scalablity of databases

5 Analysis

A discussion of the experiment's results is shown below.

1. Why does the proposed technique keep the duplication low when the number of tuples in a block (FILLFACTOR) is low?
2. Could the aforementioned condition of the proposed technique be applied to realistic workloads? Why?

5.1 Modeling

An outline of duplication models is shown in Fig.6. We assume that query sets Q is split evenly into two parts: Q_A and Q_B. The probability of the duplication in a certain block is defined as P_{mc}. Given that the distribution of the number of duplicated blocks is binominal, $P(|N_{loss}|)$ is represented as follows.

$$P(|N_{loss}| = k) = \binom{N}{k} P_{mc}^{k}(1 - P_{mc})^{N-k} \tag{3}$$

Next, we consider probability P_{mc}. At first, we model query processing. Given that one query references $|WS|$ blocks out of $|N|$ blocks in databases and reference probabilities are uniform and independent, a referenced probability P_{ref} of a certain block are define as:

$$P_{ref} = \frac{|WS|}{N} \tag{4}$$

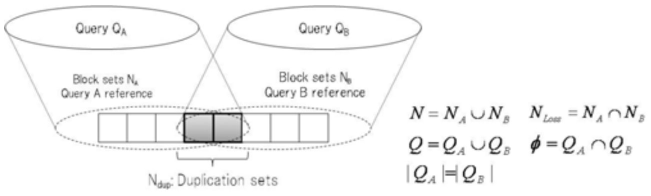


Fig. 6. The duplication model

The queries in Q defined above are independent and are input back-to-back. Thus the probabilistic distribution of the referenced number of queries related to one block is as follows.

$$P(X = l) = \binom{|Q|}{l} P_{ref}^{l}(1 - P_{ref})^{|Q|-l}, \tag{5}$$

where X means a stochastic variable of the number of queries above. Therefore, P_{mc} is represented as follows.

$$P_{mc} = \sum_{m=2}^{|Q|} P(X = m)p'_m \tag{6}$$

Given one block is referenced m times, p'_m is the conditional probability of its reference by both queries, Q_A and Q_B. Since p'_m is varied so as to classify query sets Q, it can not be simply modeled. However, if $|WS| = 1$, p'_m is always zero which prevents the duplication. Since $|WS|$ monotonically increases such as $1, 2, 3, \ldots$, we can expect it to quickly and asymptotically approach 1 from our observation. From this viewpoint, we assume p'_m is only defined as a function of $|WS|$, and (6) is rewritten as below.

$$P_{mc} = \sum_{m=2}^{|Q|} P(X = m)Q(|WS|) = P(X > 1)Q(|WS|) \tag{7}$$

Fig.7 shows that the estimation of the duplication is based on three parameters, Q, N, and WS. And $Q(|WS|)$ is defined as below. Although the error of about 10% is observed at FILLFACTOR=50, this model can roughly estimate actual duplication ratios.

$$Q(|WS|) = 1 - \frac{1}{|WS|} \tag{8}$$

In the model defined above, the higher the $P(X > 1)$ is, the bigger the number, N_{loss}, of duplicated blocks is. As $P(X > 1)$ represents the referenced probability that one block will be referenced at least twice, it is expected to increase with the number of tuples in one block. As long as the duplication conforms to the defined model, its interpretation is that if the number of tuples in one block decreases, $P(X > 1)$ is held down, and as a result of that the ratio of the duplication is kept low.

5.2 Zipf-like Distribution

In the previous subsection, the condition of our technique is that $P(X > 1)$ is held down, where one block is referenced stochastically infrequently by at least two queries such that a single tuple size is large, or one block size is small; that is, the number of tuples in one block is low. Although one of these conditions is to make FILLFACTOR smaller, it is unrealistic because it increases database

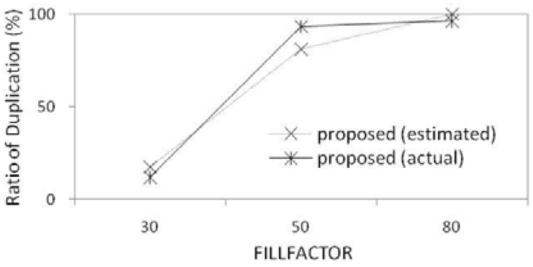


Fig. 7. Difference between estimated and actual values

sizes. Therefore, we will investigate different conditions for $P(X > 1)$ to analyze practical workloads without altering FILLFACTOR.

It is reported that workloads in the Web follow the zipf-like distribution[11], where the relative probability of one request for i'th most popular pages is proportional to $\frac{1}{i^\alpha}$, with α taking some values ranging from 0.64 to 0.83. Thus tuples in databases are expected to be referenced in accordance with this characteristics. From this observation, as long as frequent accessed tuples are scattered and located over mutliple blocks, frequently-referenced tuples and infrequently-referenced tuples are expected to be mixed up in each block. Given that infrequently accessed tuples in one block are regarded as nonexistent tuples, this situation is expected to be similar that the number of tuples packed in one block.

The evaluation of this hypothesis is shown in Fig.8, where we run the evaluation with the zipf-like distribution applied to DBT-1 with 80% of FILLFACTOR. As you can see, if $\alpha = 0.83$, the referenced data of our technique are relatively held down compared to the tuple-based one. Table 4 shows that the hit ratios of buffer caches are evaluated with these conditions. We found that the hit ratio of the proposed technique is higher than one of the tuple-based technique with $\alpha = 0.83$. What it comes down to is that given that tuples is partially accessed according to the zipf-like distribution, our technique is expected to be effective for de-duplication.

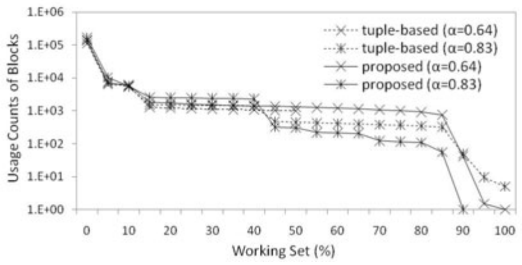


Fig. 8. A variance of working sets with the zipf-like distribution

Table 4. A comparison of the average hit ratios under the varing conditions

Conditions	Hit Ratio (%)
tuple-based, and zipf-like ($\alpha = 0.64$)	58.12
proposed, and zipf-like ($\alpha = 0.64$)	61.26
tuple-based, and zipf-like ($\alpha = 0.83$)	59.43
proposed, and zipf-like ($\alpha = 0.83$)	65.14

6 Related Work

The previous studies on de-duplication in replicated databases fall into two groups as follows.

– Cache cooperation between databases by means of global synchronism.
– Dispatchers transfer queries to databases considering each buffer cache in databases.

The former includes Oracle RAC (Real Application Clusters), and IBM DB2 pureScale8 as the well-known business products, and many techniques have been proposed[12,13,14]. These techniques need to implement a look-up table which searches where required blocks are on buffer caches in a database cluster. If required data are not in local buffer caches, databases check whether the data is held by one of the other buffer caches based on the look-up table. This approach avoids the duplication efficiently because of direct management of all buffer caches in a cluster. However, the look-up table is a shared resource, and must be synchronized as they update or reference it. This synchronization could become a bottleneck. From the standpoint of the implementation, existing databases must be extensively modified because buffer cache configurations are totally altered to permit inter-working with the other databases and the look-up table. The latter has two approach[5,6] as explained in Section 1: the query-based and the relation-based. Aside from the explained query-based technique (Section 1), there was the study that examined the size of returned values to estimate working sets of queries[15]. These approache can dispatch queries without synchronization based on predefined rules. However, the relation-based technique has little flexibilities in terms of workloads as explained in Section 1, and parameters such as SQL statements and the size of returned values are not good one estimating referenced data in databases.

This paper presents a way of realizing de-duplication in which dispatchers use the look-up table built from internal information, that is, the index access patterns; it overcomes the weaknesses of the previous approaches: needs for synchronization, poor flexibilities, and low efficiencies. Since the proposed technique can be implemented as middleware, existing databases require very little modifications. In practice, appended codes for PostgreSQL (to record snapshots) are around 50 lines.

Finally, Our technique is orthogonal to dynamic load balancing techniques. We assume query sets Q are input statically and periodically. However, actual workloads are dynamically changed, and therefore query dispatchers must handle these workloads to balance loads in addition to our technique. Several studies have examined about load balancing based on the usage of CPU and I/O, and outstanding connections to databases[6,16,17], and cooperation with these techniques is a future work.

7 Conclusion

In this paper, we presented a state-of-the-art de-duplication technique of buffer caches. Our approach records the usage counts of index leaves as snapshots from which the correlations of leaves are calculated. Common blocks between queries are estimated and identified through the snapshots, and it builds a look-up table for query dispatch. We constructed its prototype based on our technique, and evaluated it against DBT-1. We found that as long as a single block is referenced stochastically infrequently by at least two queries, our technique could eliminate the duplication. More specifically, these conditions are the following.

– A size of a single tuple is relatively-large to one block size; that is, a single tuple size is large, or one block size is small.
– Working sets of queries is small, or workloads follow the zipf-like distribution.

References

1. Ravi, J., et al.: A survey on dynamic Web content generation and delivery techniques. Journal of Network and Computer Applications 32(5), 943–960 (2009)
2. Daudjee, K., Salem, K.: Lazy Database Replication with Snapshot Isolation. In: Proceedings of 32nd International Conference on Very Large Data Bases (2006)
3. Mishima, T., et al.: Pangea: An Eager Database Replication Middleware guaranteeing Snapshot Isolation without Modification of Database Servers. In: Proceedings of 35nd International Conference on Very Large Data Bases (2009)
4. Krikellas, K., et al.: Strongly consistent replication for a bargain. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering (2010)
5. Pai, V.S., et al.: Locality-Aware Request Distribution in Cluster-based Network Servers. In: Proceedings of the 8th ACM Conference on Architectural Support for Programming Languages and Operating Systems (1998)
6. Elnikety, S., et al.: Tashkent+: Memory-Aware Load Balancing and Update Filtering in Replicated Database. SIGOPS Oper. Syst. Rev. 41(3), 399–412 (2007)
7. Transaction Processing Performance Council, http://www.tpc.org/
8. Sivasubramanian, S., et al.: Autonomic data placement strategies for update-intensive Web applications. In: Proceedings of the International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (2005)
9. Garey, M.R., et al.: Resource constrained scheduling as generalized bin packing. J. Combinatrial Teory, Ser. A, 257–298 (1976)
10. David, R.: Karger.: Global min-cuts in RNC, and other ramifications of a simple min-out algorithm. In: Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (1993)

11. Breslau, L., et al.: Web caching and zipf-like distributions: evidence and implications. In: Proceedings of IEEE Conference on Computer Communications (INFOCOM 1999), pp. 126–134 (1999)
12. Levy, H., et al.: Implementing Cooperative Prefetching and Caching in a Globally-Managed Memory System. In: Proceedings of the ACM SIGMETRICS 1998 Conference (1998)
13. Feeley, M.J., et al.: Implementing global memory management in a workstation cluster. In: Proceedings of the Fifteenth ACM symposium on Operating System Principles (1995)
14. Markatos, E.P., Dramitinos, G.: Implementation of a Reliable Remote Memory Pager. In: Proc. 1996 Usenix Technical Conf., pp. 177–190 (1996)
15. Cherkasova, L., Ponnekanti, S.R.: Optimizing a 'Content-Aware' Load Balancing Strategy for Shared Web Hosting Service. In: Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication System (2000)
16. Qin, X., et al.: Dynamic load balancing for I/O-intensive applications on clusters. ACM Transactions on Storage (TOS) 5(4), No.9 (2009)
17. Elnikety, S., et al.: A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites. In: The Proceedings of the 13th International World Wide Web Conference, WWW 2004 (2004)

Indexing for Vector Projections

Sean Chester, Alex Thomo, S. Venkatesh, and Sue Whitesides

University of Victoria, P.O. BOX 1700 STN CSC, Victoria, BC, Canada
{schester,sue}@uvic.ca, {thomo,venkat}@cs.uvic.ca

Abstract. The ability to extract the most relevant information from a dataset is paramount when the dataset is large. For data arising from a numeric domain, a pervasive means of modelling the data is to represent it in the form of vectors. This enables a range of geometric techniques; this paper introduces projection as a natural and powerful means of scoring the relevancy of vectors. As yet, there are no effective indexing techniques for quickly retrieving those vectors in a dataset that have large projections onto a query vector. We address that gap by introducing the first indexing algorithms for vectors of arbitrary dimension, producing indices with strong sub-linear and output-sensitive worst-case query cost and linear data structure size guarantees in the I/O cost model. We improve this query cost markedly for the special case of two dimensions. The derivation of these algorithms results from the novel geometric insight that is presented in this paper, the concept of a data vector's *cap*.

1 Introduction

The conceptual simplicity and expressiveness of vectors makes them widely applicable to modelling a diverse set of problems and domains, especially from business, the natural sciences, and engineering. A natural operation to apply to vectors is projection, and in this paper we study how to support efficiently identifying from a large database of vectors those which have sufficiently large projections in the direction of a query.

As an example scenario in which this is appropriate, consider a credit card company with a database of monthly card vectors, as in Table 1.[1] Analysts hired to detect suspicious credit card vectors could provide query vectors and thresholds so that monthly card vectors with sufficiently large projections onto those queries could be identified as suspicious. One analyst, for example, might be interested in large or unusual expenses that are perhaps outside the area of residence (a query $\mathbf{q}_{A0} = \langle -.5, 1 \rangle$), whereas another might be simply interested in high-volume, low-cost purchasing (a query $\mathbf{q}_{A1} = \langle 0, -1, \rangle$). The monthly card descriptions best matching the first analyst's preferences are those whose vectors have large projections in the direction of $\mathbf{q}_{A0}$ (tuples **a** and **b** in Table 1). The threshold is important, because the analysts have no idea beforehand how many vectors will be suspicious. For example, none of the tuples match $\mathbf{q}_{A1}$ well.

[1] For now, disregard τ and the last column, which will become clear later.

Table 1. A small example relation and a vector representation of each tuple. The second attribute is normalised to [1,5] and the first attribute is scaled to match the second attribute's domain by the function $f(x) = 4 * (max - x)/(max - min) + 1$.

id	% trans_ inside_ city	% trans_ above_$ 1000	scaled vector	dual point
a	25	.4	$\langle 1, 4\rangle$	$\bar{\mathbf{a}}^* = (-.25, .25\tau)$
b	15	.5	$\langle 3, 5\rangle$	$\bar{\mathbf{b}}^* = (-.60, .20\tau)$
c	5	.1	$\langle 5, 1\rangle$	$\bar{\mathbf{c}}^* = (-5, \tau)$

Supporting these applications can be costly if the set of vectors is large, since the naive approach requires a sequential scan. Ideally, in such circumstances, a database index could provide a more efficient (i.e., sub-linear) mechanism to retrieve those vectors that project strongly onto a query vector. At this point, no research directly addresses this need, so the full usefulness of vector representations (be it for projections or other algebraic operations) is constrained by the dirth of efficient (i.e., sub-linear) techniques for handling them. We discuss related work in more detail at the end of this paper (Section 5).

It is understandable that vectors are not efficiently indexed for projections, because creating such an index is a difficult problem. Indexing vectors in their natural form is not useful without *apriori* knowledge of the query vector. The difficulty is that an index organises the vectors so that similar vectors are near each other (logically in the case of secondary indices and physically in the case of primary indices). But the similarity of vectors to one another does not necessarily imply the similarity of their projections onto arbitrary query vectors. As an example, consider two arbitrary vectors $\mathbf{u}$ and $\mathbf{v}$. If a user issues a query *orthogonal to* $\mathbf{u} + \mathbf{v}$, then the vectors will be scored quite differently from each other. On the other hand, if the issued query is $\mathbf{u} + \mathbf{v}$ instead, then the vectors will have quite similar scores. Consequently, efforts to pre-organise the vectors can be quite substantially thwarted depending on user preferences. (See Figure 1.)

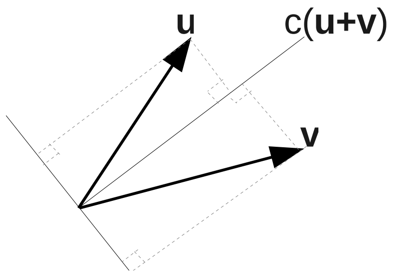


Fig. 1. An illustration in 2d of the difficulty in preprocessing vector projection. Notice that the projections of $\mathbf{u}$ and $\mathbf{v}$ are very similar in a direction like $\mathbf{u}$+$\mathbf{v}$, but substantially different onto a vector orthogonal to $\mathbf{u} + \mathbf{v}$. Without knowing which query directions are likely, it is difficult to predict whether the projections of $\mathbf{u}$ and $\mathbf{v}$ will be similar.

Our Results. In this paper we give the first indexing solutions to this vector indexing problem. Specifically, we:

- Introduce the query vector indexing problem and cast it into a geometry context in order to employ transformations and the novel geometric insight of a vector's *cap*, crucial to solving the problem (Sections 2 and 3).
- Using a duality transform, produce an algorithm for any dimension $d \geq 2$ to utilise an $\mathcal{O}(ns/b)$ simplicial partition tree data structure [1] to respond to vector projection queries, where s/b reflects the number of blocks occupied by each vector. The query cost of this index is $\mathcal{O}(n^{1-1/d+\epsilon} + ts/b)$ I/O's, where ϵ is a small constant and ts/b reflects the size of the output (Section 4.1).
- Demonstrate how additional insight in the primal space for the two dimensional case can produce a yet more efficient index based on an Interval Tree [2], querying in $\mathcal{O}(lg\ n + ts/b)$ rather than $\mathcal{O}(\sqrt{n} + ts/b)$ (Section 4.2).
- Prove geometrically how translating a dual query hyperplane along the x_d axis, a constant time operation, translates the entire dataset– effectively altering the statically defined threshold τ to any new, dynamic threshold. (Section 4.3).

2 Preliminaries

In this paper, we study the problem of efficiently retrieving from a set of vectors those that have a sufficiently large projection onto an arbitrary unit vector. Throughout the paper, we adopt the convention that boldface symbols denote vectors, that the magnitude of a vector $\mathbf{v}$ is denoted v, and that $\hat{\mathbf{v}}$ is the unit vector in the direction of $\mathbf{v}$. First recall that the *projection* of a vector $\mathbf{v}$ onto another vector $\mathbf{u}$ is the component of $\mathbf{v}$ in the direction of $\mathbf{u}$. More formally:

Definition 1. *The projection $\boldsymbol{\pi}_{\mathbf{u}}(\mathbf{v})$ of a vector $\mathbf{v}$ onto another vector $\mathbf{u}$ is the component of $\mathbf{v}$ in the direction of $\mathbf{u}$, given by $\left(\frac{\mathbf{v}\cdot\mathbf{u}}{u}\right)\mathbf{u}$.*

The objective in this paper is to respond *quickly* to *vector projection queries.* Formally, these queries are defined as follows.

Definition 2. *Given a set $\mathcal{D}$ of vectors $\mathbf{v} \in \mathbb{R}^d$ and a threshold $\tau \in \mathbb{R}^+$, the result of a vector projection query for query vector $\mathbf{q} \in \mathbb{R}^d$ is the set $\{\mathbf{v} \in \mathcal{D} : \pi_{\mathbf{q}}(\mathbf{v}) \geq \tau\}$.*

Later in this paper we apply a duality transform in order to produce an efficient indexing scheme to resolve these queries in higher dimensions. A duality transform replaces points (hyperplanes) with hyperplanes (points) in the same-dimensional Euclidean space, preserving both incidence and order. There are many such transforms, so to be specific, we use the common example defined below and illustrated in Figure 2:

Definition 3. *An initial point $p = (a_1, \ldots, a_d)$ or hyperplane $h = (b_d x_d = b_1 x_1 + \ldots + b_{d-1} x_{d-1} + c)$ is referred to as "primal." The duality transform transforms p into its "dual" hyperplane $p^* = (x_d = a_d - a_1 x_1 - \ldots - a_{d-1} x_{d-1})$ and transforms h into its dual point $h^* = (\frac{b_1}{b_d}, \ldots, \frac{b_{d-1}}{b_d}, \frac{c}{b_d})$.*

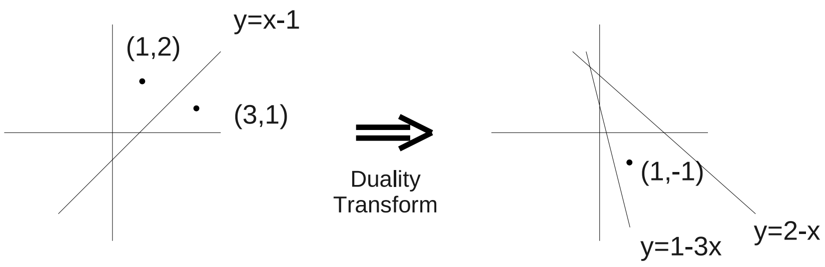


Fig. 2. An example of the duality transform we use in this paper. The point (1,2) becomes the line (y=2-x), the point (3,1) becomes the line (y=1-3x) and the line (y=x-1) becomes the point (1,-1). Notice how order is preserved with respect to the origin.

A critical property of duality transforms is that if a point p lies on the opposite side of a hyperplane h as the origin (i.e., p is *above* h), then h^* is above p^*.

Additionally, *halfspace range searchs* (or *halfspace range reporting*) are critical to this paper. Our second transformation yields a halfspace range search. Given a set of points and a query halfspace, the response to a halfspace range search is the set of points in the search space. Formally:

Definition 4. *Given a set $\mathcal{D}$ of points in $\mathbb{R}^d$ and a half-hyperplane h, the result of a halfspace range search is the set $\{p \in \mathcal{D} : p \in h\}$.*

3 Caps, Baseplanes, and Range Searching

In this section we detail the geometric techniques we employ in order to setup the use of the indexing data structures described in Section 4. We do this in a series of two techniques. First, we transform the vectors into *caps*. We then transform the caps into dual points using the duality transform. The two transformations can be combined into one $\mathcal{O}(d)$-time step per vector.

3.1 The Cap of a Vector

There are two key insights for designing our index. Firstly, recall from Definition 1 that the size of a projection $\pi_{\mathbf{q}}(\mathbf{v})$ is independent of the magnitude of $\mathbf{q}$. Hence $\pi_{\mathbf{q}}(\mathbf{v}) = \pi_{\hat{\mathbf{q}}}(\mathbf{v})$. That is to say, all queries we can treat as (or scale to) unit length. This simplifies calculations and enables us to represent the space of all possible d-dimensional queries as the unit (d-1)-sphere.[2]

The second insight is that for a given data vector $\mathbf{v}$, a representation of the set of queries for which it should be returned is computable. As the angular separation of the query direction from $\mathbf{v}$ increases, the size of the projection of

[2] Throughout the paper, we refer to the sphere in d dimensions as the (d-1)-sphere. For example, the outer surface of the Earth is (approximately) a 2-sphere.

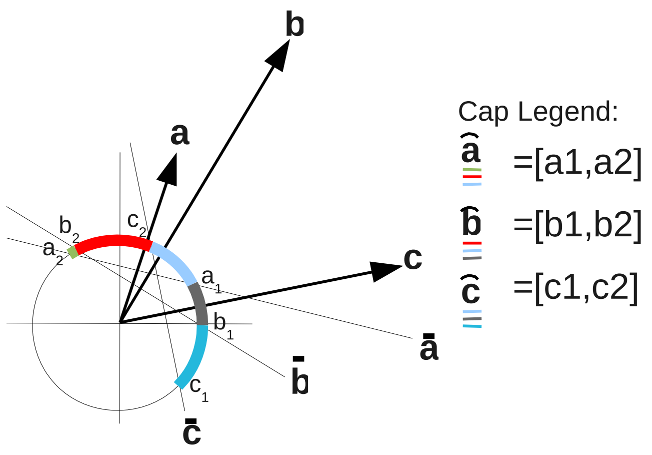

b
a
c
b2
c2
a2
a1
b1
c1
ā
b̄
c̄
Cap Legend:
$\widehat{\mathbf{a}}$ =[a1,a2]
$\widehat{\mathbf{b}}$ =[b1,b2]
$\widehat{\mathbf{c}}$ =[c1,c2]

Fig. 3. The caps and baseplanes corresponding to the scaled vectors from Table 1 for $\tau = 4$. The cap of $\mathbf{b}$ (denoted $\widehat{\mathbf{b}}$) spans the gray, blue, and red arcs, from b_1 to b_2. A query on the red arc, for example, is in $\widehat{\mathbf{a}}$ and $\widehat{\mathbf{b}}$, whereas a query on the turquoise arc is only in $\widehat{\mathbf{c}}$. The interval tree in 2d can be constructed by decomposing the circle into arc intervals $[c_1, b_1]$, $[b_1, a_1]$, $[a_1, c_2]$, etc., as illustrated by the coloured arcs. These arc intervals can be readily described by angles, as in the case of card c, which is delimited by the angles of $-.150\pi$ and $.276\pi$ from the positive x-axis.

$\mathbf{v}$ onto the query decreases monotonically. Therefore, the set is contiguous in the sense that it is the portion of the unit sphere contained within a particular halfspace. We call the *cap* of a vector $\mathbf{v}$ and denote it $\widehat{\mathbf{v}}$.

The halfspace that defines $\widehat{\mathbf{v}}$ is delimited by a hyperplane that we call the *baseplane* of $\mathbf{v}$ and denote $\bar{\mathbf{v}}$. The baseplane is defined to be the one passing through the sphere at points given by query vectors onto which the projection of $\mathbf{v}$ is exactly τ. It is clear that $\bar{\mathbf{v}}$ is orthogonal to $\mathbf{v}$: $\mathbf{v}/v$ is a radius of the unit sphere which bisects the chord drawn by $\bar{\mathbf{v}}$. The caps and baseplanes of the card vectors from Table 1 are illustrated in Figure 3.

Conveniently, the specifications of a cap can be computed quite effectively, independent of the dimension, because the cap is symmetric about the vector. Thus, we can make deductions by observing a planar cross-section of the unit sphere. The next lemma gives these specifications (see Figure 4).

Lemma 1. *The distance from the origin of the unit sphere to the base of the query cap of a vector* $\mathbf{v} = \langle v_1, \ldots, v_d \rangle$ *is* τ/v*, the radius of the query cap is* $\sqrt{(1-\tau/v)(1+\tau/v)}$*, and the equation of the plane defining the cap is* $\bar{\mathbf{v}} = (v_1 x_1 + \ldots + v_d x_d - \tau = 0)$.

Theorem 1. *Given a set of vectors* $\mathcal{D}$*, a threshold* τ*, and a query vector* $\mathbf{q}$*,* $\pi_{\mathbf{q}}(\mathbf{v}) \geq \tau \Leftrightarrow p \in \widehat{\mathbf{v}}$.

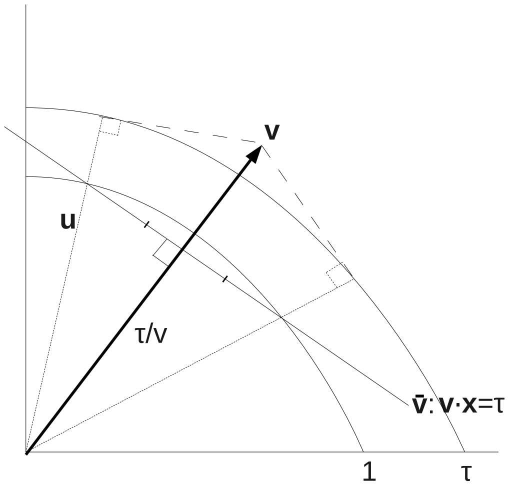


Fig. 4. An axis-parallel planar cross section of a query space, passing through the origin. The inner arc depicts all possible queries. The outer arc depicts vectors of length τ. The query space is the portion of the unit sphere bounded by the line ($\bar{\mathbf{a}}$) orthogonal to $\mathbf{a}$ and at a distance of $\frac{\tau}{a}$ from the origin. The projection of $\mathbf{a}$ onto any vector on this arc is of size at least τ.

3.2 Venturing into the Dual Space

Given the preceding discussion, it is clear that a dataset of n vectors can be interpreted as n caps or, under the assumption that queries *must* be normalised, equivalently as n baseplanes. A query $\mathbf{q}$ can be regarded as a point $p_{\mathbf{q}}$. Since a normalised query will always produce a point on the unit sphere, checking whether $p_{\mathbf{q}}$ lies above a baseplane $\bar{\mathbf{v}}$ is sufficient to determine if $\mathbf{q}$ is in $\widehat{\mathbf{v}}$. So, the problem is to determine the set of baseplanes above which $p_{\mathbf{q}}$ lies. It is to this problem that we will apply a duality transform.

Recall from our earlier introduction of a duality transform that it inverts "aboveness." Thus, if one point p is above a particular hyperplane h, then h's dual point h^* will be above the point's dual hyperplane p^*.

We convert each cap into a point by applying the duality transform to its baseplane, thus obtaining a set of n dual points. The position vector of any query can be transformed into a hyperplane. This transforms the problem into a halfspace range search, as described in Theorem 2.

Theorem 2. *Let $\mathcal{Q}$ denote a set of vector baseplanes and let $p > h$ denote that point p lies on the opposite side of the hyperplane h as does the origin (i.e., is above h). Then, for a given query $\mathbf{q}$, $\{h \in \mathcal{Q} : q > h\} = \{h \in \mathcal{Q} : h^* > q^*\}$.*

In other words, by applying a duality transform, the problem of determining in which caps a particular query lies becomes a case of halfspace range searching.

Using the particular duality transform given earlier, we transform the baseplane $\bar{\mathbf{v}}$ (namely $x_d = -\frac{v_1}{v_d}x_1 - \ldots - \frac{v_{d-1}}{v_d}x_d + \frac{\tau}{v_d^2}(\tau + v_d - 1)$) into the dual point $\bar{\mathbf{v}}^* = (-\frac{v_1}{v_d}, \ldots, -\frac{v_{d-1}}{v_d}, \frac{\tau}{v_d^2}(\tau + v_d - 1))$.

4 An Index for Vector Projections

The techniques described above allow us to reformulate the problem in such a way as to take advantage of existing efficient external memory data structures. In particular, we show in Section 4.1 how a simplicial partition tree can resolve queries with $\mathcal{O}(n^{1-1/d+\epsilon} + ts/b)$ I/O's, for a dataset of size n in d dimensions and a small constant ϵ, with t output vectors each occupying s bytes and a blocksize of b bytes per block of I/O. The value ts/b reflects the number of blocks of output. The data structure requires linear $\mathcal{O}(ns/b)$ space.

Then, we show how the use of an interval tree in two dimensions can improve the query cost from $\mathcal{O}(\sqrt{n} + ts/b)$ to $\mathcal{O}(lg\ n + ts/b)$, still in linear $\mathcal{O}(ns/b)$ space, in Section 4.2.

4.1 An Index for Arbitrary Dimension

By means of the two transformations described in Section 3, the vector projection problem can be reformulated as a case of halfspace range searching. The purpose of this is to take advantage of the extensive research that has already been conducted on the halfspace range searching problem. The external memory simplicial partition tree data structure given by Agarwal et al. [1] requires linear $\mathcal{O}(ns/b)$ space and can answer halfspace range search queries in $\mathcal{O}(n^{1-1/d+\epsilon} + ts/b)$ I/O's.

The series of transformations from a vector to a cap to a dual point can be arithmetically combined into one computation. Thus, as a result of Theorems 1 and 2, we have Algorithm 1 for preprocessing a dataset $\mathcal{D}$ into a simplicial partition tree index in order to efficiently respond to vector projection queries.

Algorithm 1

```
1. Create an empty point set $\mathcal{S}$
2. For each vector $\mathbf{v} = \langle a_1, \ldots a_d \rangle$ in the dataset:
3.     Compute the point $h_{\mathbf{v}}^* = \left(-\frac{a_1}{a_d}, \ldots, -\frac{a_{d-1}}{a_d}, \frac{\tau}{a_d}\right)$
4.     Add $h_{\mathbf{v}}^*$ to $\mathcal{S}$
5. Index the set $\mathcal{S}$ in the external memory simplicial partition tree
```

Then, for each query $\mathbf{q}$, one can compute the dual hyperplane q^* as $(x_d = q_d - q_1x_1 - \ldots - q_{d-1}x_{d-1})$ and execute a halfspace range search. Since the output is unordered, a subsequent $\mathcal{O}(ts/b)$ postprocessing step on the query results can explicitly compute the ranks. The last column of Table 1 describes the dual points produced from each original card tuple.

4.2 A Logarithmic Query-Time Index for Two Dimensions

In two dimensions, we can do better than $\mathcal{O}(\sqrt{n} + ts/b)$ if we keep the problem in the primal space (i.e., do not apply the technique of Section 3.2). This is because the unit sphere in two dimensions is a one-dimensional curve on which any position can be uniquely identified by an angle with respect to a reference axis. A cap, the intersection of the unit circle with a halfplane is simply an arc that can be readily represented by an angular interval, a start angle to an endpoint angle. Thus, assessing on which arcs a given query point lies is effectively the *interval stabbing problem.*

The interval stabbing problem is optimally supported in external memory by the interval tree of Arge and Vitter [2]. This structure uses linear $\mathcal{O}(ns/b)$ space, and can respond to queries with optimal $\mathcal{O}(lg\ n + ts/b)$ I/O's.

We arbitrarily decide to use the positive x-axis as a reference for the angles. Let $\theta_{\mathbf{v}}$ denote the angle a vector $\mathbf{v}$ makes with the positive x-axis, $\arccos(a_x/a) + c\pi/2$, for $c \in \{0, 1, 2, 3\}$ depending on the quadrant. Then, the angular interval of the arc can be computed as $\theta_{\mathbf{v}} \pm \arccos \frac{\tau}{\mathbf{v}}$ (refer to the right angle triangle in Figure 4).

For the example card database of Table 1, the arc intervals for $\tau = 4$ are illustrated in Figure 3. The vector corresponding to card c, for example, has an interval of $[c_1, c_2] = \arctan .2 \pm \arccos \frac{4}{\sqrt{26}} = [-.150\pi, .276\pi]$.

4.3 Dynamic Thresholds

Until this point, we have assumed that the user is interested in a threshold query in which the threshold is set statically by an administrator. Here we discuss how the orthogonality of the data vector to the baseplane of its cap allows us to efficiently transform the query to temporarily set a new, dynamic threshold. In fact, the threshold τ can be regarded as an arbitrary seed to initialise the data structure.

Recall from Algorithm 1 that each vector $\mathbf{v}$ is transformed into a point $h^*_{\mathbf{v}} = \left(-\frac{a_1}{a_d}, \ldots, -\frac{a_{d-1}}{a_d}, \frac{\tau}{a_d}\right)$. Consider what happens if $\mathbf{v}$ is scaled to $c\mathbf{v}$: it is transformed to the new point $h^*_{c\mathbf{v}} = \left(-\frac{ca_1}{ca_d}, \ldots, -\frac{ca_{d-1}}{ca_d}, \frac{\tau}{ca_d}\right) = \left(-\frac{a_1}{a_d}, \ldots, -\frac{a_{d-1}}{a_d}, \frac{\tau}{ca_d}\right)$. The direction of the vector is captured by the first d-1 coordinates of the dual point and its magnitude is described by the last coordinate. This is intuitive since the baseplanes of the caps of $\mathbf{v}$ and $c\mathbf{v}$ are parallel to each other. The sufficiency of the first d-1 coordinates in capturing the direction of the baseplane results from the fact that the nullspace of a line only spans d-1 dimensions and it is from translating the nullspace of the $\mathbf{v}$ that $\bar{\mathbf{v}}$ is derived.

We exploit this fact as follows. Recall that a query $\mathbf{q}$ will be transformed into a dual halfspace $q^* = (x_d = q_d - q_1x_1 - \ldots - q_{d-1}x_{d-1})$. This is intentionally expressed in point-intercept form: the value of q_d shifts the query up or down the x_d axis. Given that the relationship between the magnitude of a vector $\mathbf{v}$ and the threshold τ is expressed along the x_d axis, such a shift effectively changes the threshold. If the user wishes to instead use a threshold of τ', one need only adjust q^* to $\left(x_d = \frac{q_d\tau'}{\tau} - q_1x_1 - \ldots - q_{d-1}x_{d-1}\right)$.

Theorem 3. *For a vector projection query index initialised with a threshold τ, the response to a query vector $\mathbf{q} = \langle q_1, \ldots, q_d \rangle$ for another threshold τ' is the same as the response to query vector $\langle q_1, \ldots, q_{d-1}, q_d\tau'/\tau \rangle$ with threshold τ.*[3]

5 Related Work

The closest problem to the one we study in this paper is that of providing indices for linear optimisation queries. A linear optimisation query, if one interprets tuples as vectors, ranks them against a query vector by dot product. The first result was due to Matoušek [10], who demonstrated that the top result for such a query is always on the convex hull of the dataset. Chang et al. [6] extended this idea to produce an algorithm to produce the top k responses based on the observation that since the top result is on the convex hull, the top k results must be in the first k convex layers. A different approach was given by Agarwal et al. [1], in which, like us, the authors employed a duality transform technique to arrive at a case of halfspace range searching. The similarity of the solutions results from our observation in Section 3.1 that $\pi_{\mathbf{q}}(\mathbf{v}) = \pi_{\hat{\mathbf{q}}}(\mathbf{v}) = \mathbf{v} \cdot \hat{\mathbf{q}}$. Two other particularly interesting papers have addressed linear optimisation queries. Marian et al. [7] consider records that are distributed across different sources, as in a search engine. These ideas could be quite relevant to extending our work to a distributed database setting. Tsaparas et al. [11] use vector projections to create an index for very general top k queries, but only for problems in the plane.

A technique important to our approach is the use of a duality transform, which allows us to cast the problem into an instance of halfspace range searching. For more information about duality transforms we refer to the Computational Geometry text of de Berg et al. [4] and to the survey by Matoušek [9], both which demonstrate their use nicely and repeatedly. Range searching is a canonical problem in Computational Geometry, of which halfspace range searching is a specific, well-studied case. In higher dimensions, the best well-known algorithm is the one due to Matoušek [8]. A similar algorithm with improved $\mathcal{O}(nlg\ n)$ preprocessing time was given this year by Chan [5]. For a more general weighted case of the problem, Arya et al. [3] have provided tight lower bounds to match the upper bounds of Matoušek. The external memory simplicial partition tree of Agrawal et al. [1] is an adaptation for disk of Matoušek's data structure.

The interval tree data structure mentioned for our logarithmic-cost two dimensional index is from Arge and Vitter [2].

6 Conclusion

In this paper we have introduced the problem of efficiently reporting vectors with large projections in a query direction and the concept of a vector's cap, the component of the unit d-sphere bounded by a hyperplane orthogonal to the vector and at a distance from the origin proportionate to the vector's magnitude.

[3] Recall that the thresholds are from $\mathbb{R}^+$.

Semantically, the cap of a vector is a representation of the set of exactly those queries for which the vector is part of the solution set.

Using this insight led to a general dimension index, based on previous literature and requiring $\mathcal{O}(n^{1-1/d+\epsilon} + ts/b)$ I/O blocks to respond to an arbitrary query. We also gave an interval tree-based index for the particular case of two dimensions, which led to an improvement to $\mathcal{O}(lg\ n + ts/b)$ blocks of I/O per query. Finally, we described how our indices readily support adaptive thresholds.

It is clear that there is much opportunity to extend this work given the novelty of the problem. In particular, our future work will focus on aiding the query engine by estimating the size of a result set for a given query. Additionally, we intend to experimentally validate the relative efficiency of the techniques described, thus interpreting their performance in practice. It would also be very interesting to determine asymptotic lower bounds on the I/O cost for queries on an $\mathcal{O}(ns/b)$ data structure.

References

1. Agarwal, P.K., Arge, L., Erickson, J., Franciosa, P.G., Vitter, J.S.: Efficient searching with linear constraints. Journal of Computer and System Sciences 61, 194–216 (2000)
2. Arge, L., Vitter, J.S.: Optimal external memory interval management. SIAM Journal of Computing 32(6), 1488–1508 (2003)
3. Arya, S., Mount, D.M., Xia, J.: Tight lower bounds for halfspace range searching. In: Proceedings of the 26th Annual Symposium on Computational Geometry, pp. 29–37. ACM, New York (2010)
4. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications, 3rd edn. Springer, Heidelberg (2008)
5. Chan, T.M.: Optimal partition trees. In: Proceedings of the 26th Annual Symposium on Computational Geometry. ACM, New York (2010)
6. Chang, Y.C., Bergman, L., Castelli, V., Li, C.S., Lo, M.L., Smith, J.R.: The onion technique: indexing for linear optimization queries. In: Proceedings of the 26th SIGMOD International Conference on Management of Data. ACM, New York (2000)
7. Marian, A., Bruno, N., Gravano, L.: Evaluating top-k queries over web-accessible databases. ACM Transactions on Database Systems 29, 319–362 (2004)
8. Matoušek, J.: Reporting points in halfspaces. Computational Geometry: Theory and Applications 2(3), 169–186 (1992)
9. Matoušek, J.: Geometric range searching. ACM Computing Surveys 26(4), 422–461 (1994)
10. Matoušek, J., Schwarzkopf, O.: Linear optimization queries. In: Proceedings of the 8th Annual Symposium on Computational Geometry. ACM, New York (1992)
11. Tsaparas, P., Palpanas, T., Kotidis, Y., Koudas, N., Srivastava, D.: Ranked join indices. In: Proceedings of the 19th International Conference on Data Engineering, pp. 277–288. IEEE, Los Alamitos (2003)

Assessment of Cardiovascular Disease Risk Prediction Models: Evaluation Methods

Richi Nayak and Ellen Pitt

Computer Science Discipline, Queensland University of Technology
Brisbane, QLD, Australia
r.nayak@qut.edu.au

Abstract. This paper uses a real world anaesthesia time-series monitoring data in the prediction of cardiovascular disease risk in a manner similar to exercise electrocardiography. Models derived using the entire anaesthesia population and subgroups based on pre-anaesthesia likelihood of complications are compared in an attempt to ascertain which model performance measures are best suited to populations with differing pre-test probability of disease. Misclassification rate (MR) as a measure of prediction model performance was compared with Kappa statistic, sensitivity, specificity, positive and negative predictive values and area under the receiver operating characteristic curve (AUC). In this medical application of data mining, MR is shown to be dependent on the prevalence of disease within the population studied but AUC and Kappa statistic are shown to be independent of disease prevalence.

Keywords: Cardiovascular risk, prediction, misclassification rate, accuracy, area under receiver operating characteristic curve (AUC), Kappa statistic.

1 Introduction

Assessment of cardiovascular risk in specific patients and in the general population is an important component of medical care. Such importance is related to the relatively high incidence and prevalence of disease and the high disease mortality and morbidity [1]. General cardiology research has considered various methods for the prediction of cardiovascular disease risk [2]. One of them is exercise electrocardiography (exECG). ECG test performance has been improved by consideration of non ECG related parameters (exercise tolerance, heart rate (HR) variability, HR recovery) as well as patient risk factor status (hypertension (HT), hyper-lipidaemias, smoking, diabetes and obesity) [3]. Of importance in interpretation of exECG benefit is the false positive rate which is associated with potential unnecessary further investigation as well as additional clinical risk and cost.

Data mining methods are increasingly being applied to time series data and in the medical domain [4]. Recent examples of this include the use of intensive care monitoring data in the prediction of multi-organs system failure [5], ICU survival [6] and use of laboratory data in the prediction of progression to renal failure in diabetic nephropathy [7]. The most appropriate methods for pre-processing and prediction modelling are considered to be data dependent and examples of medical domain application have used decision tree (DT), artificial neural networks (ANN) and logistic regression (LR) prediction methods. Issues of dealing with large time series data and unbalanced data have been addressed. The importance of feature selection has been noted in these studies as wrapper methods have been shown to perform poorly [8]. Decision tree (DT) methods have been shown to be more comprehensible and variables used have domain relevance. The accuracy of DT is somewhat lower than that for models derived using Naive Bayes (NB) and Tree Augmented Naive Bayes (TAN) [5].

Several measures for the evaluation of prediction models exist and these include misclassification rate (MR), accuracy, mean absolute error (MAE), root mean squared error (RMSE), computational complexity, comprehensibility of the generated classifier together with sensitivity, specificity, positive and negative predictive values and the area under the receiver operating characteristic curves (AUC) and Kappa statistic [10]. In general data mining applications, MR, MAE and RMSE are considered standard measures for assessment of model performance. Measures of performance used in medical domain include sensitivity, specificity, positive and negative predictive value as well as the area under the receiver operating characteristic curve [9, 10]. These measures are based on the number of true positives (TP), true negatives (TN), false positives (FN) and false negatives (FN) and are defined in Table 1.

Table 1. Evaluation measures used in prediction model assessment

Measure	**Definition**	
Accuracy	Percentage of correctly classified instances	(TP+TN/ (TP+FP+TN+FN)) * 100
MR	Percentage of instances misclassified	1- Accuracy
Sensitivity / Recall	Ratio of class instances predicted by rule or DT (Proportion of those with the disease and a positive test)	TP/TP+FN
Specificity	Ratio of class instances not satisfying a rule and not being n the class. (Proportion of those without disease who have a negative test result)	TN/FP+TN
Positive predictive value	Proportion of instances with a positive result and the disease or disease risk	TP/TP+FP
Negative predictive value	Proportion of those without the disease or disease risk who do not satisfy the rule or have a negative test	TN/TN+FN
Area under ROC curve	Represents the relationship between sensitivity and specificity such that higher AUC represent the best balance between the ability of a rule to correctly identify positive and negative cases	Area under the curve plotting TP against FP
Kappa statistic	Measure which allows for improvement in accuracy over that which would be obtained by chance alone. Difference between observed and expected agreement as expressed as a fraction of maximum difference	I_o = TP+TN/ TP+TN+FP+FN I_e=((TN+FN)(TN+FP)+(TP+FP)(TP+FN))/n^2 where n = TP+TN+FP+FN $\kappa = I_o - I_e / 1 - I_e$

The presence of unbalanced data is an issue in assessing the performance of modelling methods [7]. Classifier performance can be biased. This issue can be addressed using pre-processing methods that under-sample the majority class such that classes have an equal or otherwise nominated class distribution. The kappa statistic also takes into account any bias related to class distribution [12]. The maximum value for Kappa is 1 representing perfect agreement while Kappa values of 0.21-0.40 and those of 0.41-0.6 represent fair and moderate agreement respectively. Values between 0.61 and 0.80 show substantial agreement [12].

In anaesthesia practice, data collected is the same as that assessed during exECG and, if shown to be useful in predicting cardiovascular risk, is available at no additional cost or procedural risk. Therefore, the goal of this study is to assess whether the techniques of data mining can develop simple models for the prediction of cardiovascular risk and to determine which methods of model performance are best suited to this area of the medical domain.

2 The Proposed Cardiovascular Risk Prediction Method

This anaesthesia time series data represents 5 months (January to May, 2007) of non-cardiac anaesthesia procedures at a tertiary care hospital. Heart rate and ST segment level values were available for most cases in the dataset and were chosen as input variables. Systolic blood pressure was available for those cases in which invasive haemodynamic monitoring (arterial line) was used. Cases were grouped according to the intensity of monitoring (Table 2). For prediction of cardiovascular disease, the target variable chosen was that of a diagnosis of a cardiovascular related disease made within a 10 year period (1999 to 2008). Risk factor variables were extracted from the hospital health record database and included a history of cigarette smoking (both past and current), lipid disorder, hypertension, diabetes mellitus and obesity. These were considered together with age and gender from the demographic table of the anaesthetic information management (AIM) database.

2.1 Pre-processing and Datasets

Time series data were used to derive variables based either on raw time series data (dataset A) or time series data from which outliers (dataset B) and noise (dataset D) or both (dataset C) have been removed. Another dataset was based on counts of physiological variable values outside 1, 2, 3 and 4 standard deviations from the norm for each case and this same dataset included the count of variable values outside the accepted normal range for each variable (dataset E). A further dataset was based on summary statistics for segments of the time series data, either complete segment or portion of the segment (dataset F). The final pre-processing method, Symbolic Aggregate Approximation (SAX) transformation, was used to derive the final dataset (dataset G). The time dimension and variable values were normalised and cluster membership based on time series patterns was included for each of the physiological variables and different alphabet size and segment count. Derived variables for datasets A to D, included summary statistics for HR, ST segment level and systolic blood pressure (SBP) as well as variables representing heart rate recovery and classifying ST segment changes as they would be in an exECG setting. Table 3 shows the components of the datasets analysed here.

The time-series derived datasets and the datasets comprised of time-series and risk factor variables were subjected to several feature selection methods and the method found to produce the most compact and comprehensible models was the subset filter method, *Cfs* [11]). The details of these experiments are not presented here but results shown here are based on *Cfs* subsets.

Table 2. Classification of cases according to intensity of operative monitoring

Risk group	Definition	Number
General population	All valid cases	3377
***Low risk* subgroup**	Cases with basic monitoring only	2764
***High risk* subgroup**	Cases for which invasive haemodynamic monitoring (arterial line) was considered appropriate based on pre-operative assessment of anaesthetic and surgical risk. Represents a subgroup of general population and includes those for which more extensive ECG monitoring was used	613
***Very high risk* subgroup**	Cases for which 5 ECG leads were monitored and data available for 3 ECG leads. All had arterial line for continuous BP monitoring. Subset of *High Risk* subgroup	249

Table 3. Variables used in model development

Variable group	Variable type	Variable examples
Time-series	Cardiovascular	HR, ST, SBP
Demographic	Patient related	Age, gender
	Procedure related	Duration of monitoring (trend length)
Clinical	Risk factor	Smoking, lipid disorder, HT, DM, Obesity
	Target	Coronary vascular disease

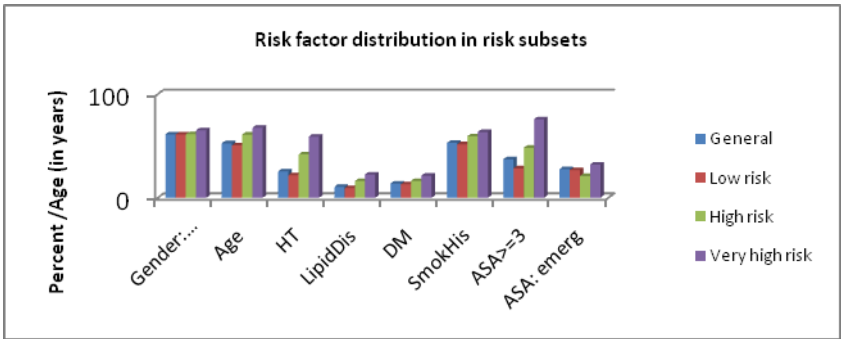


Fig. 1. Risk factor characteristics of cases in risk subgroups

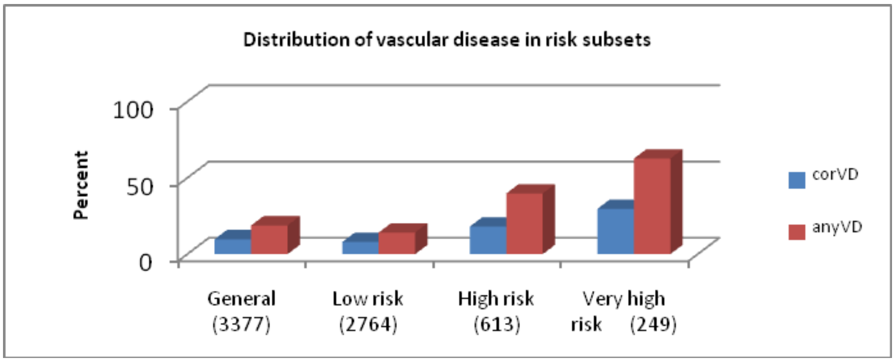


Fig. 2. Vascular disease distribution in risk subgroups

The risk factor characteristics of the population subgroups are shown in Figure 1. Figure 2 shows the distribution of vascular disease, both coronary vascular (*corVD*) and any vascular disease (*anyVD*) in each of the pre-operative risk subgroups as well as the *general population*. The increase in vascular disease prevalence with increased intensity of monitoring is seen and, for all subgroups, the prevalence of any vascular disease is mostly twice the prevalence of coronary vascular disease (*corVD*). In the *general population*, the prevalence of coronary vascular disease is almost 10% while that for *anyVD* is approximately 18%. For the *very high risk* subgroup, the prevalence of *corVD* is over 20% while that for *anyVD* is almost 60%.

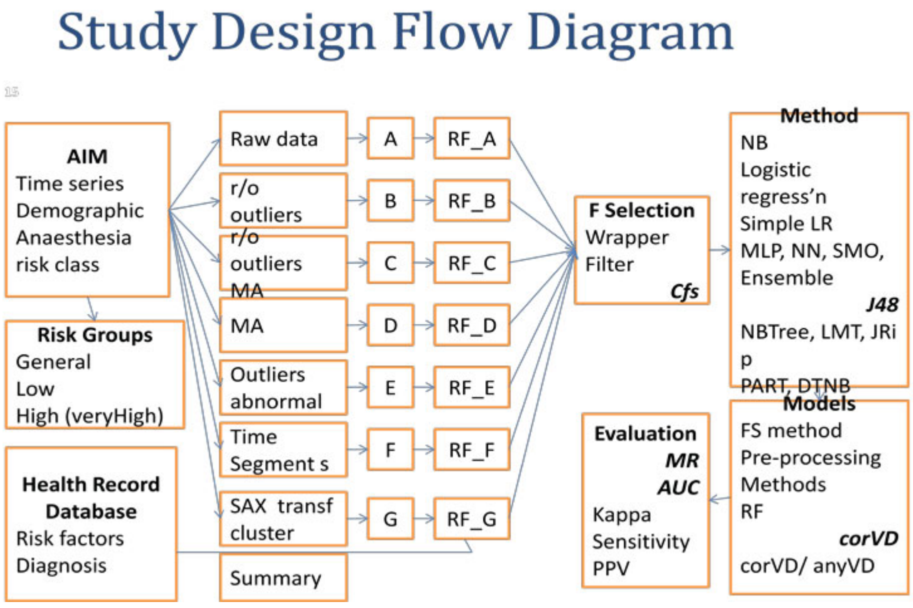


Fig. 3. The Proposed Methodology

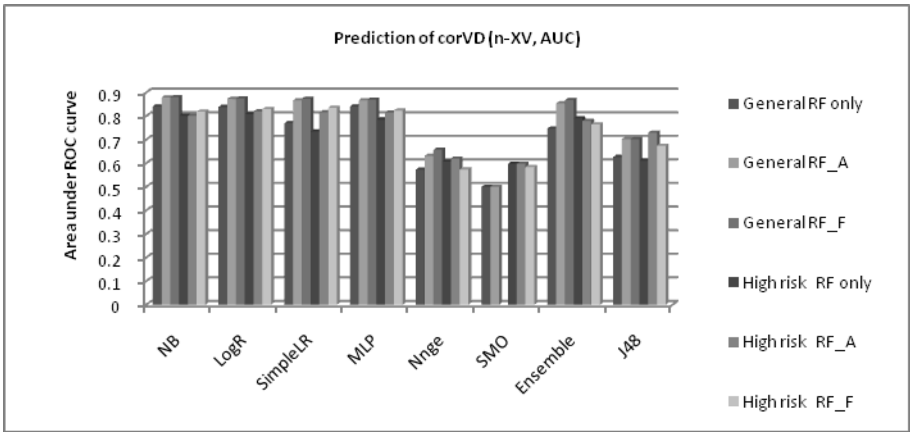


Fig. 4. Comparison of methods in prediction of *corVD* in selected datasets and using stratified cross validation (AUC)

2.2 Prediction Model Development

Several prediction methods such as logistic regression, neural networks, support vector machine etc were used in building the models. Models derived using decision tree method J48 (WEKA implementation of C4.5 [11]) were shown to be the most comprehensible and compact with acceptable performance (figure 4). The performance for models based on risk factors or time series variables alone are compared with the performance of datasets based on a combination of risk factor and time series variables. Study design is summarised in figure 3. The dataset was subjected to several feature selection methods and results of these are not presented here. Models presented here were derived using *Cfs* [11] reduced subsets.

3 Model Evaluation

Several measures of model performance are used in the evaluation of the prediction models here. Since class distribution for the datasets being studied here is unbalanced, majority class undersampling (WEKA pre-processing method, *SpreadSubsample* selection) is used in some models and their performance is compared with that of models based on stratified *n*-cross validation. The *SpreadSubsample* method creates subsets of cases with a user chosen distribution of class variable. For this section of the data analysis, subsets with equal class distribution were created [11].

3.1 Misclassification Rate for Balanced and Unbalanced Data

The misclassification rates associated with models based on these data are compared with models evaluated using stratified but unbalanced data subsets. Figure 5 shows the performance data for the prediction of *corVD*. MR for models based on stratified *n*-cross validation of unbalanced datasets shows a marked difference between the the two risk categories (*high risk* and the *general* population). The MR for the *high risk*

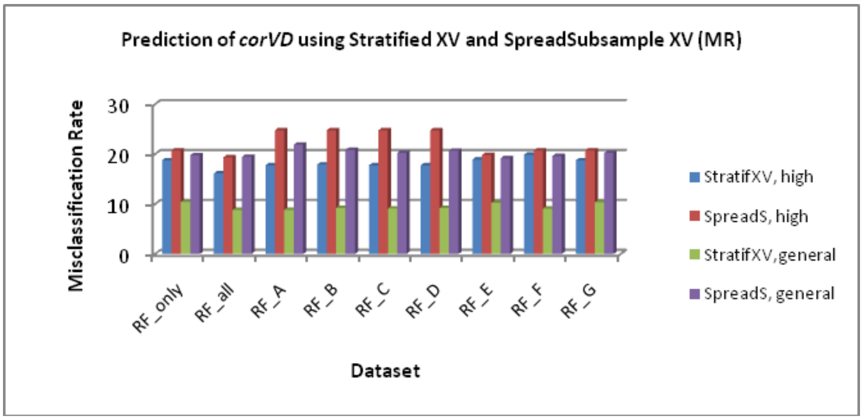


Fig. 5. Decision tree prediction of *corVD* using both stratified *n*-cross validation and *Spread-Subsample n*-cross validation (misclassification rate)

category is almost twice that of the *general* population. This does not imply an inherently worse model but relates primarily to the class distribution for the two groups. In subsets of these datasets with balanced class distribution, the differences are much less. It can be seen that the model is able to correctly classify cases more accurately than chance alone would allow (distribution of *corVD* cases in the population studied is 50% and the model misclassfication rate is only 18-25%).

3.2 Area under ROC Curve in Balanced and Unbalanced Data

Similar data for models evaluated using AUC are shown in Figure 6. For balanced and unbalanced data, the area under the ROC (AUC) is similar for the *high risk* population and there is a marginal increase for the *general* population. This suggests that

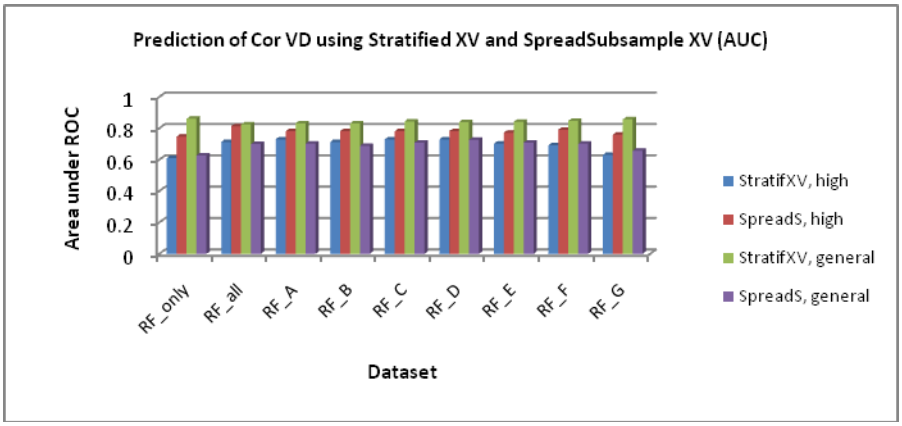


Fig. 6. Prediction of coronary vascular disease using both stratified cross validation and SpreadSubsample cross validation and J48 decision tree (area under ROC)

for the evaluation of models based on this dataset, the AUC is less biased by the class distribution, that is by the pre-test probability of disease.

3.3 Sensitivity, Specificity and Predictive Values

The use of other performance measures is shown in the Table 4.

Table 4. Other performance measures for models based on all RF and time-series variables in the prediction of corVD in both *general population* and *high risk* subgroup

Other performance measures for *corVD* models, *general* population, all variables								
	Sensitivity		**Specificity**		**PPV**		**NPV**	
	General population	***High risk subgroup***	***General population***	***High risk subgroup***	***General population***	***High risk subgroup***	***General population***	***High risk subgroup***
RF_only	0.281	0.297	0.975	0.946	0.554	0.55	0.926	0.859
All	0.375	0.351	0.943	0.875	0.418	0.382	0.933	0.859
A	0.356	0.441	0.961	0.91	0.5	0.521	0.932	0.881
B	0.369	0.441	0.959	0.91	0.494	0.521	0.933	0.881
C	0.341	0.441	0.96	0.91	0.483	0.521	0.931	0.881
D	0.369	0.441	0.953	0.91	0.459	0.521	0.933	0.881
E	0.329	0.396	0.959	0.894	0.464	0.454	0.929	0.87
F	0.411	0.396	0.946	0.867	0.453	0.396	0.937	0.867
G	0.26	0.279	0.953	0.904	0.374	0.392	0.922	0.85

It can be seen that these measures are less sensitive to class distribution. For models based on the full variable set and both RF and time-series variables for the *general population*, the sensitivity ranges from 0.26 to 0.37 while for the *high risk* subgroup sensitivity ranges from 0.28 to 0.44. The variation in specificity, PPV, and NPV values between the models based on the *general population* and the *high risk* population vary less than do the MR values for the same models.

3.4 Kappa Statistic in Unbalanced Data

The Kappa statistic compensates for the variation in class distribution by measuring the agreement between the predicted and the observed categorisations of the dataset and corrects for chance based agreement. A comparison of Kappa statistic values as they relate to MR and AUC is seen in Figures 7-9. Figure 9 shows the kappa statistic values for models derived from datasets RF_only, RF_A, RF_F and Figure 7 and Figure 8 show MR and AUC for the corresponding models. For all risk categories, the Kappa statistic increases in association with the addition of some time-series derived variables to the RF only based model. However, for each risk category the Kappa statistic values are in the same range. This is also seen in Figure 8 showing AUC for the risk related subsets and the *general* population. Misclassification data (Figure 7) shows obvious differences in model performance based on the class distribution in the subsets studied.

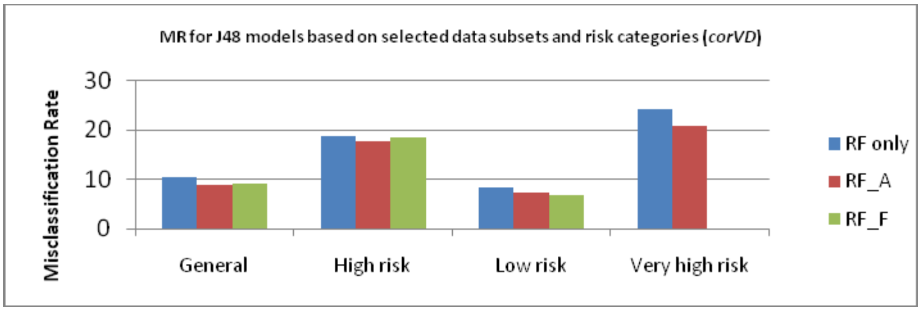


Fig. 7. Use of MR for evaluation of model performance based on selected subsets (RF_only, RF_A and RF_F) in the general and risk subset groups (DT J48, *corVD*)

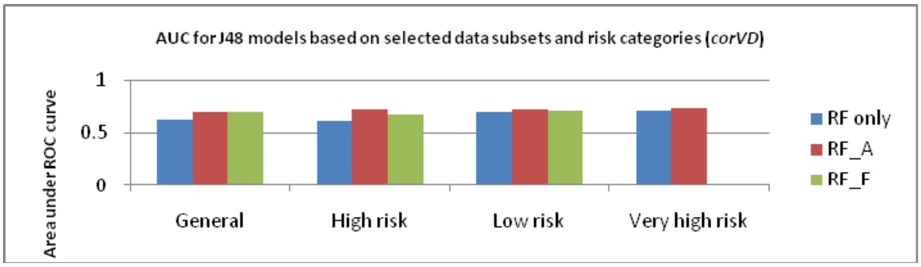


Fig. 8. Use of AUC for evaluation of model performance based on selected subsets (RF_only, RF_A d RF_F) in the general and risk subset groups (DT J48, *corVD*)

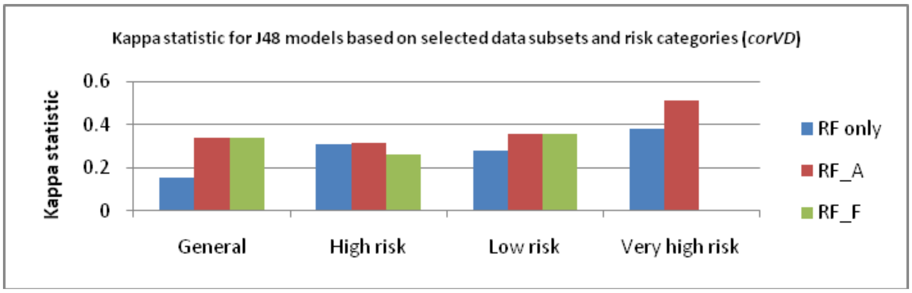


Fig. 9. Kappa statistic for evaluation of model performance based on selected subsets in the general and risk subset groups (DT J48, *corVD*)

3.5 Discussion

The results presented above show the effect of disease prevalence on the model performance as assessed using several measures. In the evaluation of models based on a dataset with unbalanced class distribution, there is a reduction in MR consistent with the low prevalence of disease in the *general population*. When class distribution is artificially balanced, the MR increases, however it remains below that which would be expected if prediction was random or based solely on disease prevalence. For the *high risk* subgroup, MR for unbalanced data is consistent with prevalence of disease in that group. Evaluation of the models based on modified dataset with balanced class distribution shows a limited increase in MR but again, the increase is less than would be expected if prediction was random, that is for 50% with *corVD*. Chance alone would result in a MR of 50%. Models, however, have a MR of only 20-25%.

Consideration of MR in comparison to AUC and Kappa statistic for models based on selected datasets containing risk factor variables again shows the marked disparity in model performance according to disease prevalence in the subgroup being assessed. The *low risk* and *very high risk* subgroups are included in this analysis and the MR ranges from 7% to 24 %. The same models assessed by Kappa statistic show markedly less variability. The performance of RF_A based models varies little across the *low risk* to *high risk* subgroups (0.3o to 0.35) while for the *very high risk* subgroup, Kappa statistic is approximately 0.5 for the same dataset. A likely explanation for this is the relatively small number of cases in this group (249 compared to 3377 for the *general population* and 2764 for the *low risk* subgroup. The increased Kappa statistic values for the *low risk* subgroup and *general population* suggest the value of this performance measure in cases whose operative risk was considered to be low. It is in these groups that identification of increased risk of cardiovascular disease is of greatest benefit.

For the evaluations based on AUC, the variability is reduced even further with AUC values ranging from 0.60 to 0.70 for all groups except for the very high risk subset (0.75). Again the slightly higher value for this subset is unlikely to be significant and is more likely in association with the smaller sample size. A trend toward better performance in models based on risk factor variables in addition to time-series variables is seen for all of the evaluation measures. These models were based on *Cfs* reduced datasets and, for the small number of variables in the RF_only based models, performance has been underestimated. However, use of all RF variables in combination with *Cfs* reduced time series subsets resulted in models of increased complexity without improvement in accuracy. This study has demonstration the effectiveness of AUC in another setting within the medical domain. The findings are in keeping with those noted in prediction of ICU survival [6].

Models developed in this study, if validated in a broader population, may represent a screening method of cardiovascular disease risk for which there is no additional cost or procedural risk. Assessment of the models in broader populations requires the use of appropriate evaluation methods. Results obtained for the real world dataset studied here show the value of different evaluation measures and stress the limited value of misclassification rate in assessing risk for populations with differing pre-test probability of disease. AUC and Kappa statistic are shown to be far less sensitive to class distribution and thus represent better measures of prediction model performance. Such findings are consistent with those noted in prediction of ICU survival [6].

While Kappa statistic and AUC are shown to be more appropriate than MR in the assessment of anaesthesia data based models, the issue of false positive prediction remains. Addition of other time series variables, such as the more precise representation of HR variability (R-R interval) to model input, may improve model performance and may better predict the risk of SCD. While the current study relied heavily on time-series statistical summary data, further efforts to address the issue of variable time-series duration may include re-sampling the data or use of dynamic time warping techniques.

4 Conclusions

This study has shown that (1) anaesthesia related time-series data when subjected to data mining methods can predict disease risk more effectively than disease prevalence alone would allow. AUC for sample models based on a combination of RF and time-series variables range from 0.65 to 0.75; (2) MR as a model performance measure is much more susceptible to bias by the prevalence of disease in the population group being studied; (3) Kappa statistic and AUC as measures of prediction model performance are much less biased by class distribution; (4) Sensitivity and specificity measures add little to AUC; and (5) Positive and negative prediction values are less affected by disease prevalence than MR but somewhat more so than AUC and Kappa values. They do however offer important information to clinicians faced with clinical decisions in the setting of limited background information.

References

1. WHO, D. UN Data: Age-standardized mortality rate for cardiovascular disease (per 100,000 population), `http://data.un.org/Data.aspx?d=WHO&f=inID%3aMBD22` [cited 2009, February 7]
2. Pearson, T., et al.: AHA guidelines for primary prevention of cardiovascular disease and stroke: 2002 update: consensus panel guide to comprehensive risk reduction for adult patient without coronary or other vascular disease. Circulation 106, 388–391 (2002)
3. Mora, S., et al.: Enhanced Risk Assessment in Asymptomatic Individuals With Exercise Testing and Framingham Risk Scores. Circulation 112, 1566–1572 (2005)
4. Bellazzi, R., Zupan, B.: Predictive data mining in clinical medicine: Current issues and guidelines. International Journal of Medical Informatics 77(2), 81–97 (2008)
5. Ramon, J., et al.: Mining data from intensive care patients. Advanced Engineering Informatics 21(3), 243–256 (2007)
6. Luaces, O., et al.: Predicting the probability of survival in intensive care unit patients from a small number of variables and training examples. Artificial Intelligence in Medicine 45(1), 63–76 (2009)
7. Cho, B.H., et al.: Application of irregular and unbalanced data to predict diabetic nephropathy using visualization and feature selection methods. Artificial Intelligence in Medicine 42(1), 37–53 (2008)
8. Chen, L., et al.: Decision tool for the early diagnosis of trauma patient hypovolemia. Journal of Biomedical Informatics 41(3), 469–478 (2008)
9. Cios, K.J., Moore, G.W.: Uniqueness of medical data mining. Artificial Intelligence in Medicine 26(1-2), 1–24 (2002)
10. Sokolova, M., Lapalme, G.: A systemic analysis of performance measures for classification tasks. Information Processing and Management 45, 427–437 (2009)
11. Witten, I.H., Frank, E.: Credibility: Evaluating What's Been Learned. In: Data Mining: Practical Machine Learning Tools and Techniques, p. 173. Morgan Kaufmann, San Francisco (2005)
12. Armitage, P., Berry, G.: Statistical Methods in Medical Research. Blackwell Sciences Pty, Ltd., Malden (1994)

Visual Analysis of Implicit Social Networks for Suspicious Behavior Detection

Amyn Bennamane[1], Hakim Hacid[1], Arnaud Ansiaux[1], and Alain Cagnati[2]

[1] Alcatel-Lucent Bell Labs France, Route de Villejust, 91620, Nozay, France
[2] Ministère de l'Intérieur, ST(SI)^2
{firstname.lastname}@alcatel-lucent.com, alain.cagnati@interieur.gouv.fr

Abstract. In this paper we show how social networks, implicitly built from communication data, can serve as a basis for suspicious behavior detection from large communications data (landlines and mobile phone calls) provided by communication services providers for criminal investigators following two procedures: lawful interception and data retention. We propose the following contributions: (i) a data model and a set of operators for querying this data in order to extract suspicious behavior and (ii) a user friendly and easy-to-navigate visual representation for communication data with a prototype implementation.

1 Introduction

Social networks, although an old topic, have gained an impressive importance with the appearance of on-line social networks like Facebook[1], Twitter[2], and Orkut[3]. This type of networks, generally built through an explicit declaration of the social relation by the user, is only the visible part of the iceberg. In fact, social networks could be implicitly constructed on different situations where advanced analysis is needed [15][13]. Modeling a situation as a social network is motivated by the need of understanding a person or a group of persons (or a resource) not only as a particular case but as a whole thanks to the relation she/it may have with other individuals/groups. Considering the relations between individuals and their behavior instead of considering only individuals is of a great interest and has been leveraged in several situations like: viral marketing [12], terrorism attacks [8], criminal networks identification [16], to cite only few.

The power of social networks, as well as communication means, can be used in unlawful ways. An illegal operation can range from a simple illegal use of others' communication means to more serious activities such as terrorism. To identify suspicious operations, investigations are conducted by investigators who are generally not experts in Information Technologies (IT) and/or data analysis. In fact, investigators are mainly field workers, they usually use IT as an additional toll and precious source of information to enrich their knowledge about specific

[1] http://facebook.com
[2] http://twitter.com/
[3] http://orkut.com/

cases. Spreadsheet programs are used in this context to manage lists, to sort and make calculations in the hopes of building meaningful relations between the data, thus extracting valuable knowledge. Spreadsheets are very powerful tools for analyzing tabular data [9], but are not apt for revealing patterns in semi- or non-structured data with many complex relationships and cross-references, as well as significant meta-data. Unless the user knows what she is looking for, it's very difficult to directly extract new knowledge from the tables.

The major problem facing our users is the ability to extract meaningful and useful facts from large, heterogeneous, and disparate datasets. This work is performed in the VIGIEs research project where the goal is to provide French authorities with a tool to capture, store, and analyze in an effective way all intercepted information from fixed telephony, VoIP[4], mobile telephony, etc. but also from the Internet. In this paper we consider the case of telecommunication data (landlines and mobile telephony) provided by services providers for investigators for handling suspicious cases as an illustrative case. We propose the following contributions: (i) a data model and a set of operators for querying this data. These operators are used by investigators to formulate queries in order to extract suspicious behaviors and (ii) a user-friendly and easy-to-navigate visual representation for communication data with a prototype implementation.

The rest of this paper is organized as follows: Section 2 motivates the work with an example and presents a quick description of some related work by focusing only on social networks and visualization. Section 3 presents the proposed general data model for handling multi-channel communication analysis as well as a set of proposed operators to interact with the data model. Section 4 presents a prototype implementation of the tool integrating some of the proposed concepts as well as additional concepts required by investigators. Finally, Section 5 concludes and provides some future directions of this work.

2 Related Work

2.1 Example of a Scenario

As mentioned above, the use of electronic data is necessary in many cases to provide proof of the innocence or the guilt of a person. Hereafter is a description of a real criminal case, shown in a demonstration with the French authorities on July 2010, highlighting some of the challenges in this area: Wilhelm Gatter is an important European business man visiting Paris. On the 22nd of June 2010, while joining a meeting, he is kidnapped by an organized and well coordinated group. The group seems to be organized into several teams, each taking responsibility for specific part in the kidnapping. Few hours later, several cars were left trailing, burned and the group and the victim were not found. The kidnappers used a succession of cars to bring the hostage escape outside the Paris region. The first of these cars is burned to remove traces and was found on June 25, 2010. In the debris of the fire, several mobile phones were found including one which has been

[4] VoIP: Voice Over IP.

partially damaged. After few days, technical services have successfully rebuilt its IMEI[5]: “#45449826248390”.

Starting from this IMEI, investigators ask now for the associated phone number. The result of that query shows that the phone was linked to several SIM cards and therefore multiple subscriptions for few time ago. The authorities want now to retrieve the customers’ details of these subscriptions and formulate the following to services providers: “could you provide us with customers’ details associated to MSISDN number “#33632399599”?”. After this query, the authorities want to recover the call logs for all the communications of that number since 04/01/2010. The authorities are looking into their database if the numbers that have called or have been called by the considered number, one is known in an earlier case. Authorities continue their investigation and want to know the details about specific customer: “Could you provide us with the coordinates of the client associated to number: “#33975621491”?”.

The number “#33975621491” is a number known to the police, an interception is also in progress on this issue. An interesting conversation from a cyber-cafe is intercepted. Authorities are asking for the result of interception in text format with a tool to make the transcript “speech to text”. The authorities now wish to recover the history of calls made and received by Mr. Gatter: “Can you give us the history of communications number “#33643618644”?” The authorities now wish to have the history of calls made on two cells: one corresponding to the hotel where the Mr. Gatter was staying, and that corresponding to the place of his abduction: “Can you give us the history of communication cells 184,380 and 252,310 on 22/06/2010?”

From the communication perspective, this data comes from two mandatory functions provided by telecommunication operators and service providers: *lawful interception*, which occurs in real time once the targets are identified, and *data retention*, which is applied afterwards to understand or extract behaviors. In both cases, the main problem remains to manage the complexity of the data and their volume while considering strong time constraints imposed by authority actions.

2.2 Social Networks Analysis

A social network is a graph representation of all the interactions that occur between people. This structure may be inferred or extracted from common interests between users where nodes are persons and objects, and edges are the existing relationship between them. Social Network Analysis (SNA) deals with the understanding of the underlying phenomena in social networks [14]. SNA considers the understanding of the social interactions not only at the node level by introducing indicators such as centrality to measure the importance of a node, but also considers the links between the nodes by considering communities and the phenomena in those communities like influence.

There are several measures which may be applied on a social network to understand the underlying structure like: diameter, transitivity, centrality, cohesion,

[5] IMEI: International Mobile Equipment Identity.

density, etc. but centrality is the most frequently used one. There has been a lot of effort done in analyzing and mining social networks in the previous years to discover a variety of phenomena [10].The most recent ones include: detecting anomalies [18], predicting interests of the social entities [1], learning influence probabilities [6], identifying trends [5], mining mobility behavior [4], etc. With current on-line social networks and more generally, communication networks, the amount of information regarding both the nodes and the interactions became more available. In addition, the available size of social networks became much larger than what is considered in classical studies. As an example, *Facebook* has more than 450 M users (statistics of July 2010). The existing measures may not have meaning anymore in such a context and the hidden structures may not be extracted using the existing techniques. As a consequence, more work is necessary to propose better techniques and strategies on large social graphs.

2.3 Data Visualization

It is commonly agreed human brain can better process, analyze problems, and extract interesting information visually rather than lists of data [3]. Our work is closely related to data visualization, a vast field involving techniques ranging from simple data positioning over a plane to more complex representations such as neighborhoods, communities, etc. In the graph visualization, many of the issues are related to layout optimization, to present a graph in the most readable way within a reasonable time. TopoLayout algorithm [2] relies on state of the art of graph layout algorithms to isolate prominent graph components and apply individually the most appropriate layout algorithm.

More users oriented, some initiatives intend to give an ability to field experts to work by harnessing their competences as much as possible. *Systematic Yet Flexible (SYF) Discovery* [11] presents an approach centered around the dynamics of the field expert in order to assist her in her investigation. SYF is a generic Framework that follows the overall evolution of analysis tasks, cancels them, executes them over new data sets, annotates them, etc. *Vizster* [7] proposes operations based on SN metrics such as visual (centrality) clustering.

In this work, we are mainly interested in criminal networks visual analysis which found a significant rise in scientific literature after the events of September 11^{th}. Xu and Chen [17] present a state of the art of structural features of criminal networks, as well as a classification of visualization and SNA tools into three categories which rely on data representation as graphs, easily understandable by the humans. Our approach for SNA and criminal network detection is a semi-automatic approach in the sense that we consider automated generation of social graphs but the user intervenes to implement the analysis chain that is needed to verify her hypothesis. As a first contribution, we propose to adopt an approach that offers functionalities in graph manipulation, which we call “visualization operations”.A second contribution we propose is an approach that constraints the data model that would permit to define simple operators in defining visual queries over graph data.

3 A General Model for Communication Data Representation

The objective we are expecting is the following: Having a set of interactions (i.e., activities) happening between people using communications means, how to enable savvy users (i.e., users who don't have as their primary function data analysis or computer science, e.g., criminal investigators) to extract valuable facts from huge interaction data? We first describe a generic data model we have proposed to handle multi-channel communications. Then we move to the description of a set of basic operators built on top of the model to answer some of the investigators' needs.

3.1 Data Model

To be able to provide a simple tool and high analysis capabilities for the user we started by observing the way investigators work when analyzing communication (interaction) data. At a logical level, users generally follow activities of a certain entity, e.g., a person. These activities could be performed on different channels, e.g., phone, email, money transfer, etc. Once each channel is analyzed alone, the conclusion from each channel are aggregated to have a higher level perspective on the case. Intuitively, while the consideration of basic interactions serve to draw very fine grained conclusions, the aggregation serves as an interpretation level. Thus, our model intends to translate this observation. We start by defining a specific structure called *s-Graph* to translate the followed entity.

Definition 1. ***(Super graph (s-Graph))*** *A directed, weighted, and labeled graph capturing a highest level of interaction between real or virtual objects. An s-Graph is an aggregation of several sub-graphs, called property graphs (p-Graph). Nodes of this graph represent the objects with their properties and the links the interaction between those objects.*

Let Ω denotes a s-Graph of objects, say persons, defined as $\Omega(\overline{V}, \overline{A}, \overline{L}(\overline{V}), \overline{W}(\overline{A}))$ with $\overline{V} = \{\overline{v}_1, ..., \overline{v}_n\}$ representing a set of n nodes of the graph (corresponding to a set of persons in this case). $\overline{A}$ represents of set of arcs linking the set of nodes of the graph. It should be noted that the this is a virtual set of arcs (as we will see it in the following paragraphs, which is built by the aggregation of several arcs coming from sub-graphs). Since Ω is a directed and weighted graph, we can define a function $\omega : \overline{V} \times \overline{V} \rightarrow R^+$ such that:

$$\forall \overline{v}_i \in \overline{V}, \overline{v}_j \in \overline{V}, (\overline{v}_i, \overline{v}_j) \in \overline{A} \ iff \ \omega(\overline{v}_i, \overline{v}_j) \in R^+ \quad (1)$$

Thus, ω associates a weight for all the couples of nodes which have a common interactions. This means that there is no arc between the two nodes. Ω is a directed graph, the following property apply then for each arc:

$$\forall \overline{v}_i \in \overline{V}, \overline{v}_j \in \overline{V}, (\overline{v}_i, \overline{v}_j) \neq (\overline{v}_j, \overline{v}_i) \quad (2)$$

$\overline{L}(\overline{V})$ represents a set of labels associated to each node of the super graph. We denote this set with $\overline{L}$ and we define it $\overline{L} = \{\overline{l}_i | 1 \leq i \leq m, \forall i, j, \overline{l}_i \neq \overline{l}_j\}$. We define a function $\ell : \overline{V} \rightarrow \overline{L}$ such that:

$$\forall \overline{v}_i \in \overline{V}, \ell(\overline{v}_i) = \overline{l}_i, \overline{l}_i \in \overline{L} \wedge \overline{l}_i \neq \overline{l}_j \implies \ell'(\overline{l}_i) \neq \ell'(\overline{l}_j) \wedge \overline{v}_i \neq \overline{v}_j \implies \ell(\overline{v}_i) \neq \ell(\overline{v}_j) \tag{3}$$

Where ℓ' is the inverse function of ℓ. Equation 3 explicits the fact that each node of the s-Graph is identified with a unique label. At this stage, examples of labels may include, but not limited to, social security number, combination of attributes, e.g., first name, last name, and date of birth, etc. We consider the label as a passive attribute where its role is more informative. In contrast, as this will be explained later, there are other active attributes, called *linking properties* which are used for transporting information and communication.

Intuitively, since a s-Graph is an aggregation of graph (as defined in Definition 1), all its components should also be an aggregation of sub-components composing those aggregated sub-graphs. We introduce the notion of *Property* which is similar to the definition of an attribute in the ER model except that in our case, we consider as a property only attributes describing nodes of the s-Graph and which have a connectivity capability. This means that Properties are those attributes which help in materializing the links between individuals such as phone numbers, email addresses, bank accounts, etc. Let $P = \{p_1, ..., p_k\}$ be the set of linking properties (or properties for short).

Definition 2. ***(Linking property)*** *A linking property is an attribute which can identify and capture the existence of a type of interaction between real or virtual objects.*

Thus, a property is different from a label defined in the previous paragraphs since: (i) the labels have only an informative role where properties have an active role (i.e., building connexions between individuals), and (ii) A property may have a label, (iii) a property has a type describing its behavior. We denote by $p(\overline{V}_i) \in P$ the set of linking properties attached to the super node $\overline{V}_i$. To be a complete as possible, let's consider T as a set of types defined as follows: $T = \{t_i | 1 \leq i \leq s, \forall i, j : t_i \neq t_j\}$ We define a function $\tau : P \rightarrow T$ such that:

$$\forall p_i \in P, \tau(p_i) = t_i, t_i \in T \wedge t_i \neq t_j \Rightarrow \tau'(p_i) \neq \tau'(p_j) \tag{4}$$

Since a linking property is a part of a node of a super graph which enables linking to other nodes, each property can be considered as a separate part of the system for, e.g., a deeper analysis. By doing that way, we can build several graphs based on the type of each property. We introduce then a graph structure that we call a *property graph* (or *p-Graph* for short).

Definition 3. ***(Property graph (p-Graph))*** *A directed, weighted and labeled graph structure which: (i) links nodes having the same type and (ii) materializing an interaction.*

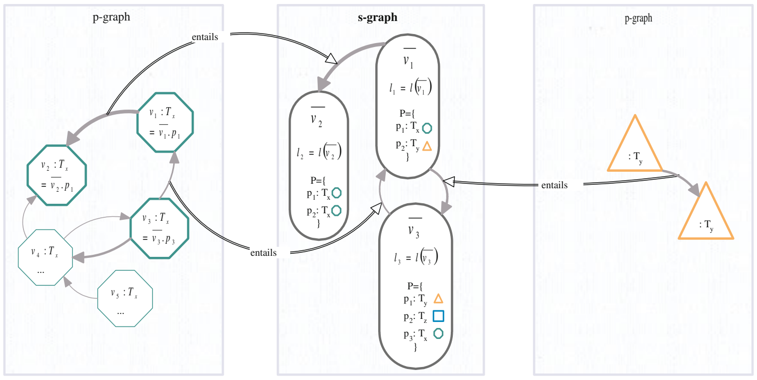

p-graph
s-graph
p-graph
entails
entails
entails
$v_1 : T_x = \overline{v_1}.p_1$
$v_2 : T_x = \overline{v_2}.p_1$
$v_3 : T_x = \overline{v_3}.p_3$
$v_4 : T_x$ ...
$v_5 : T_x$ ...
$\overline{v_1}$ $l_1 = l(\overline{v_1})$ P={ p1: Tx p2: Ty }
$\overline{v_2}$ $l_2 = l(\overline{v_2})$ P={ p1: Tx p2: Tx }
$\overline{v_3}$ $l_3 = l(\overline{v_3})$ P={ p1: Ty p2: Tz p3: Tx }
: Ty
: Ty

Fig. 1. Illustration of the different concepts of the data model and their relations

Formally, a p-Graph, denoted by G can be written as $G(V, A, L(V), W(A))$ With V corresponds to the set of k vertices representing the set of linking properties, i.e., each node of the graph is a linking property. This is done for notation simplicity and ease of explanation. Thus, we abuse the notation of $\tau(p_i)$ by applying τ on the nodes v_i: $\tau(v_i)$. A is a set of arcs resulting from connecting linking properties. The arcs are constrained by the type of the properties as explained in the previous paragraphs. We revisit then the definition of arcs in Equation 1 by adding this specific type constraint:

$$\forall v_i \in V, v_j \in V, (v_i, v_j) \in A \; iff \; \tau(v_i) = \tau(v_j) \wedge \omega(v_i, v_j) \in R^+ \tag{5}$$

The labels are defined in the same way as in Ω. At this stage we can intuitively understand that an s-Graph is composed of several p-Graphs. Thus we can denote this: $\Omega = \cup_{i=1}^{k} G_i$. Finally, the following applies: (i) A node of an s-Graph may have several properties of the same type. For example, if we consider $\overline{V}_i$ as a specific person, its not excluded to have a person with many phone numbers and email addresses. Intuitively, an inclusion relation is defined between the property nodes, *p-Graph* and the nodes of the *s-Graph*. (ii) Nodes of the s-Graph don't necessarily have the same number of properties. This can be considered as a consequence of the previous observation. This property is a pure ground constraint and is useful to to translate the diversity of information an analyst may have on analyzed objects.

3.2 Operators for Visual Social Networks Analysis

The defined data model supports the representation of different communication channels but is not enough to be a useful tool for end-users in their analysis. In this section, we define a first set of operators that translate users' needs in terms of information and hypothesis checking. The choice of following operators oriented strategy instead of functionality based strategy is motivated by: (i) the heavy process (in terms of resources and time) needed to implement each need of the user as a functionality in the tool, and (ii) the increasing need of expressive power by the user which joins and complicates (i). These operators are intended to enable users in expressing their basic needs and then, by combining them, expressing increasingly complex operations. Finally, two type of operators are proposed: (i) data definition operators and (ii) data manipulation operators.

Data definition operators. This type of operators are dedicated to satisfy the need of creating component of the data model, e.g., nodes, links, etc. This type of operators is needed to enable the investigators to, e.g., group interactions into one physical object. We define two operators: (i) *CREATE* and (ii) *ASSOCIATE.*

CREATE : This operator helps in creating a new component of the model, e.g., a node (both in s-Graph and p-Graph), an arc, a linking property, etc. Depending on the nature of the component, i.e., node, arc, property, etc. the structure of this operator changes.

ASSOCIATE ($\oplus$) : This operators enables the association of a node (or an arc) of a p-Graph to a node (or an arc) of the s-Graph. Formula 6 illustrates the definition of this operator in the case of a node of a p-Graph which is associated to a node of an s-Graph. The association operator is defined as $\oplus : V \times \overline{V} \rightarrow \overline{V}$:

$$\forall v_i \in V, \forall \overline{v}_j \overline{V} : v_i \oplus \overline{v}_j = \overline{v}'_j \ where \ p(\overline{v}'_j) = p(\overline{v}_j) \cup \{v_i\} \tag{6}$$

Data manipulation operators. This type of operator is intended to build queries over the data in the sense of manipulation like selecting a set of data, filtering according to some criteria, grouping, etc. According to the most frequent queries fired by users, we have defined a set of five main operators: (i) Origin of the arc ($\overrightarrow{\eta}$), (ii) target of the arc ($\overleftarrow{\eta}$), (iii) neighborhood (Θ), (iv) union ($\bigcup$), and (v) intersection ($\bigcap$).

Origin and Target of an arc ($\overrightarrow{\eta}$, $\overleftarrow{\eta}$) : These two operators capture two of the most operations performed by investigators when analyzing a communication network: the origin and the target of a communication. Intuitively, this operator operates on the arcs associated to an individual which are generally useful to understand or capture social phenomena like chains and information flows. Formula 7 formalizes the two operators:

$$\forall v_i, v_j \in V, \forall j = 1..k, \overrightarrow{\eta}\ (v_i) = \{(v_i, v_j) \in A\}\,; \overrightarrow{\eta}\ (v_i) = \{(v_j, v_i) \in A\} \tag{7}$$

Neighborhood (Θ) : While the two previous operators operate on the links by considering their direction, the neighbourhood operator retrieves the nodes that are directly linked to a specified node. This is an important operator if since it enables to build more complicated operators which help in understanding the surrounding activity to a particular user (or communication channel). The neighbourhood operator, $\Theta : V \rightarrow V^*$, could be formalized as follows:

$$\forall v_i \in V, \Theta(v_i) = \{\forall v_j \in V, j = 1..k : (v_i, v_j) \in A \vee (v_j, v_i) \in A\} \tag{8}$$

Union ($\bigcup$) : The previous operators are defined on a particular node. The union operates on more than one node and translates the need of recovering nodes which have participated to different interactions with selected nodes useful for reducing the set of nodes which are analyzed in more detail.

$$\forall v_i, v_j \in V, v_i \bigcup v_j = \{\Theta(v_i) \cup \Theta(v_j)\} \tag{9}$$

Intersection ($\bigcap$) : Retrieves common nodes who have been contacted by all the specified nodes. This is an important operation since it enables investigators to extract important nodes and understand information flows, sequences, and mainly common communications linking individual through others.

$$\forall v_i, v_j \in V, v_i \bigcap v_j = \{\Theta(v_i) \cap \Theta(v_j)\} \tag{10}$$

Most of these operators are defined to apply on nodes of the same p-Graph. Generally, and as specified previously, investigators check their hypothesis by moving from a type of interaction to another type. To make it possible to leverage the whole capabilities of the data model, we propose to extend the definition of some of the basic operators to make them able to take into consideration the existence of different types of interactions (i.e., p-Graphs) and a global type (i.e., s-Graph). The operators which are considered by this extension are: the neighborhood, the union, and the intersection.

The basic idea behind the extension is as follows: for any operation between nodes of different p-Graphs (i.e., different communication channels), the answer to the query passes absolutely through the corresponding nodes of the s-Graph. In addition to that, a set of meta-data are generally attached to each interaction, e.g., time, location, etc. These meta-data are currently used through the association of this information to the links. Due to the lack of space, we don't detail these operators.

4 Prototype Implementation: SemanticXL

The proposed approach is intended to feed a tool, SemanticXL, which will be used by real investigators to solve criminal cases. As noticed in the beginning, we have heavily used RDF[6] for data representation, storage and integration in the tool but since this not the focus of the paper, we do not go in more details.The high level architecture of the system is shown in Figure 2. The first level is the data integration level where the different data sources are transformed to an RDF representation with an additional use of a communication ontology. The second level is the visualization level where graph representations are drawn to materialize the link between the different individual items. The proposed data model as well as the operators fit into an intermediary level between the visualization and the querying, i.e., in the translation level. The role of this level is to capture the actions of the user, translate them to the data model to fire

[6] Resource Description Framework (http://www.w3.org/RDF/).

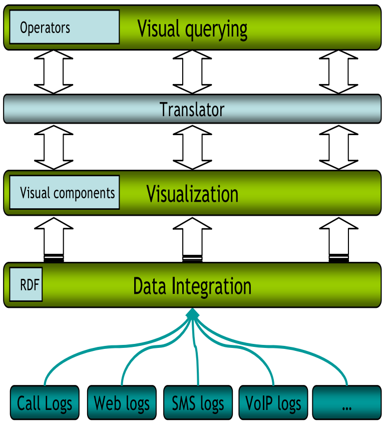


Fig. 2. Illustration of the different levels of the application

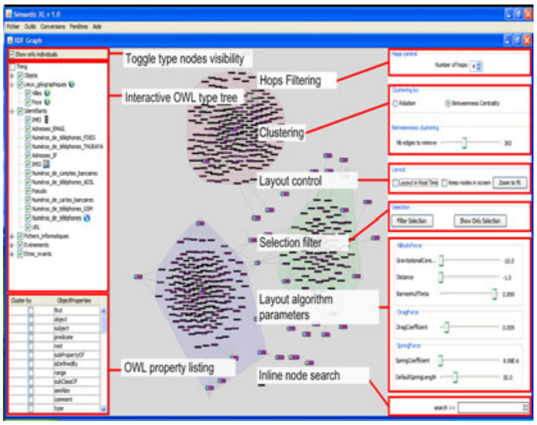


Fig. 3. SemanticXL: Illustration of the current prototype

a SPARQL[7] query over the RDF data store. On the other hand, it translates a result offered by the RDF data store into a coherent representation with the proposed data model.

Figure 3 illustrates a prototype version and the current status of SemanticXL. This prototype in its very basic version has been implemented at the French Ministry of Internal Affairs as an initiative to help investigators in better using the communication data. We have heavily improved it by adding other functionalities and by abstracting the problem of communication data analysis. Although the operators are not completely implemented, some of them are in the form of functionalities in this version and illustrate very well the interest of such approach. In the following, we describe the different functionalities available in the tool.

The most important part for the user is certainly the central view. It contains a graph representing individuals with nodes, and relations with edges. By default, the user has the ability to control the scale and move the nodes. Technically, this visualization draws nodes using the Prefuse API[8]. Nodes are laid out with a physical layout algorithm called force-directed layout (FDL) algorithm. This algorithm, or more generally type of algorithms, implements a physical simulation of interacting forces. Nodes repel mutually, links behave like springs, and friction forces are applied. The Prefuse implementation of FDL can be executed once with a succession of iterations, or regularly to obtain a real time animation and a dynamic and interactive layout.

The initial implementation of this algorithm has a problem inherent to its working principle, and we experienced difficulties to get around it. A configuration of the algorithm exists to prevent the nodes from going out from the visible

[7] SPARQL Query Language for RDF: http://www.w3.org/TR/rdf-sparql-query/
[8] Prefuse API: http://prefuse.org/doc/api/

visualization area, but this solution causes the nodes to crowd around the borders of the area, which is not satisfactory. We customized the Prefuse layout system by adding an invisible attraction force between all nodes in the center of the visualization area.

The left side contains another view which translates the proposed model. Technically, the data model is implemented using an OWL ontology where each OWL type represents a linking property. This is transparent for the user who still works with the logic of communication channels separation. A filtering feature supports the navigation between one p-Graph and another simply by using a check-box as a visual component. Finally, the right view concentrates most of the functionalities (operators) that allow to handle the central view. As an example operator, the neighborhood which is translated in the *Hops filtering* functionality. Its working principle is that the user selects a node and then she chooses a value for the hops. All nodes that are distant to the selected nodes to more than the selected value are then hidden. Other functionalities are implemented in the interface in addition to the operators, in particular the clustering[9]. All these functionalities and the prototype received a good feedback[10].

5 Conclusion and Future Work

This paper described a model and a tool for visual analysis of communication data (i.e., social networks). We proposed a general model for multi-channel communications analysis, a very important contribution for this area. The model is mainly based on the real experience of field workers, i.e., investigators. This work intends also to bring a support for users who are mainly novice users in IT. We presented a prototype implementing the different proposals of the paper. Although the prototype implements the operators for this version as functionalities, the contribution has received a positive feedback and seems to be promising for the targeted users.

As a future work, we plan to improve SemanticXL by offering visual representation of the operators for a higher flexibility for the users. From the data perspective, we think at integrating wider range of data coming mainly from the Web to open the system to external sources, an important need of the investigators. Our immediate next target is to improve the prototype to be demonstrated for the French authorities in the frame of the VIGIEs project around June 2011. A study about the completeness of the operators will be performed to decide whether this set is sufficient or not. Finally, even if the whole work has been done under the supervision of a user who is expert in criminal social networks analysis, testing the tool with real users will be a priority once the operators are visually represented to learn potential improvement axis.

[9] This functionality is not detailed in this paper since this is not in its main scope.
[10] The presentation of the paper, if accepted, will be followed by a presentation of the tool on real datasets to show its capabilities.

References

1. Agarwal, A., Rambow, O., Bhardwaj, N.: Predicting interests of people on online social networks. In: IEEE CSE 2009, pp. 735–740. IEEE Computer Society, Washington, DC, USA (2009)
2. Archambault, D., Munzner, T., Auber, D.: Topolayout: Multilevel graph layout by topological features. IEEE Trans. Vis. Comput. Graph. 13(2), 305–317 (2007)
3. DeFanti, T.A., Brown, M.D., McCormick, B.H.: Visualization: Expanding scientific and engineering research opportunities. Computer 22(8), 12–25 (1989)
4. Giannotti, F., Nanni, M., Pedreschi, D., Renso, C., Trasarti, R.: Mining mobility behavior from trajectory data. In: IEEE CSE 2009, pp. 948–951. IEEE Computer Society, Washington, DC, USA (2009)
5. Gloor, P.A., Krauss, J., Nann, S., Fischbach, K., Schoder, D.: Web science 2.0: Identifying trends through semantic social network analysis. In: IEEE CSE 2009, vol. 4, pp. 215–222 (2009)
6. Goyal, A., Bonchi, F., Lakshmanan, L.V.: Learning influence probabilities in social networks. In: WSDM 2010, pp. 241–250. ACM, New York (2010)
7. Heer, J., Boyd, D.: Vizster: Visualizing online social networks. In: INFOVIS, p. 5 (2005)
8. Krebs, V.E.: Mapping networks of terrorist cells. Connections 24(3), 43–52 (2001)
9. Lakshmanan, L.V.S., Subramanian, S.N., Goyal, N., Krishnamurthy, R.: On query spreadsheets. In: ICDE, pp. 134–141 (1998)
10. Mislove, A., Viswanath, B., Gummadi, K.P., Druschel, P.: You are who you know: inferring user profiles in online social networks. In: WSDM 2010, pp. 251–260. ACM, New York (2010)
11. Perer, A., Shneiderman, B.: Systematic yet flexible discovery: guiding domain experts through exploratory data analysis. In: IUI, pp. 109–118 (2008)
12. Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: KDD 2002, pp. 61–70. ACM, New York (2002)
13. Roth, M., Ben-David, A., Deutscher, D., Flysher, G., Horn, I., Leichtberg, A., Leiser, N., Merom, R., Mattias, Y.: Suggesting friends using the implicit social graph. In: SIG-KDD (2010) (to appear)
14. Stanley, W., Katherine, F.: Social Network Analysis: Methods and Applications Structural Analysis in the Social Sciences, 1st edn. Cambridge University Press, Cambridge (1994)
15. Wen, Z., Lin, C.-Y.: On the quality of inferring interests from social neighbors. In: KDD 2010, pp. 373–382. ACM, New York (2010)
16. Xu, J., Chen, H.: Criminal network analysis and visualization. Commun. ACM 48(6), 100–107 (2005)
17. Xu, J.J., Chen, H.: Criminal network analysis and visualization. Commun. ACM 48(6), 100–107 (2005)
18. Zhan, J., Oommen, B.J., Crisostomo, J.: Anomaly detection in dynamic social systems using weak estimators. In: IEEE CSE 2009, pp. 18–25. IEEE Computer Society, Washington, DC, USA (2009)

Compositional Information Extraction Methodology from Medical Reports

Pratibha Rani[1], Raghunath Reddy[1], Devika Mathur[2], Subhadip Bandyopadhyay[2], and Arijit Laha[2]

[1] International Institute of Information Technology, Hyderabad
[2] Infosys Technologies Ltd., SETLabs, Hyderabad
{pratibha_rani,raghunath_r}@research.iiit.ac.in,
{subhadip_b,devika_mathur,arijit_laha}@infosys.com

Abstract. Currently health care industry is undergoing a huge expansion in different aspects. Advances in Clinical Informatics (CI) are an important part of this expansion process. One of the goals of CI is to apply Information Technology for better patient care service provision through two major applications namely electronic health care data management and information extraction from medical documents. In this paper we focus on the second application. For better management and fruitful use of information, it is necessary to contextually segregate important/relevant information buried in a huge corpus of unstructured texts. Hence Information Extraction (IE) from unstructured texts becomes a key technology in CI that deals with different sub-topics like extraction of biomedical entity and relations, passage/paragraph level information extraction, ontological study of diseases and treatments, summarization and topic identification etc. Though literature is promising for different IE tasks for individual topics, availability of an integrated approach for contextually relevant IE from medical documents is not apparent enough. To this end, we propose a compositional approach using integration of contextually (domain specific) constructed IE modules to improve knowledge support for patient care activity. The input to this composite system is free format medical case reports containing stage wise information corresponding to the evolution path of a patient care activity. The output is a compilation of various types of extracted information organized under different tags like past medical history, sign/symptoms, test and test results, diseases, treatment and follow up. The outcome is aimed to help the health care professionals in exploring a large corpus of medical case-studies and selecting only relevant component level information according to need/interest.

Keywords: Information Extraction, Medical document mining, Health care application, Clinical Informatics.

1 Introduction

Clinical Informatics (CI) is a recent field of IT application research emphasizing better quality of patient care in simultaneity with cost optimization. This in turn promises a huge scope of business application in health care industry. The core technology behind this lies in the domain of electronic health care data management and information extraction from unstructured documents, the two parallel mainstreams in CI. The resulting applications induce better decision making in different contexts like better treatment provision, enhancing quality of life of patients and so on.

In this paper we consider the IE field and present a compositional approach for information extraction from free format text related to patient care process. Our aim is to extract information relevant to different context in the form of passage/collection of sentences from documents and present them in a composed, self contained format. This approach has one essential generic concept; a document creation is an outcome of evolution of a compound activity in a specific domain. From initiation to completion the compound activity is viewed in terms of granules of interlinked sub-activities at different intermediate stages creating interdependent contextual information packets as output. These contextual information are distributed along the corresponding document(s) in an entangled manner. An illustration of this concept in a patient care process is given in Fig. 1 as an ordered activity network. The document creation is actually carried out following the underlying activity network in patient care domain. By relevant extraction of information we emphasize the fact that our approach will extract and organize information pertinent to these individual contexts.

Information extraction from medical documents has been addressed mostly from discrete perspectives where the interest is usually on a few specific components. The major challenges in building a holistic approach for relevant information extraction from texts generated in patient care process are non explicitness, repetition of information across the document in varied expression and overlapping of information belonging to different implicitly expressed contexts. The bottom up view of a document creation through compilation of information artifacts generated from integrated sub-activities, as emphasized in this paper, helps to overcome this problem. Along the same line of thought, we propose individual modules for information extraction from each class (sign/symptom, past medical history etc.) and thus the whole process can be viewed as an integrated system. The extracted fragments of information from different classes are ultimately compiled to make a complete structure.

The rest of the paper is organized as follows. Section 2 discusses the motivation and section 3 presents related studies in context to this and similar problems. The methodologies and proposed algorithms are discussed in section 4. In section 5 we present experimental study and results along with discussion on the computational aspect of this problem. Finally section 6 concludes the article with some comments on our focus and nature of the solution.

2 Motivation

Let us consider the representation of a patient care activity in terms of activity flow (Fig. 1) which is the motivation pivoting subsequent development of concepts and discussions in this paper. As depicted in the figure, we perceive

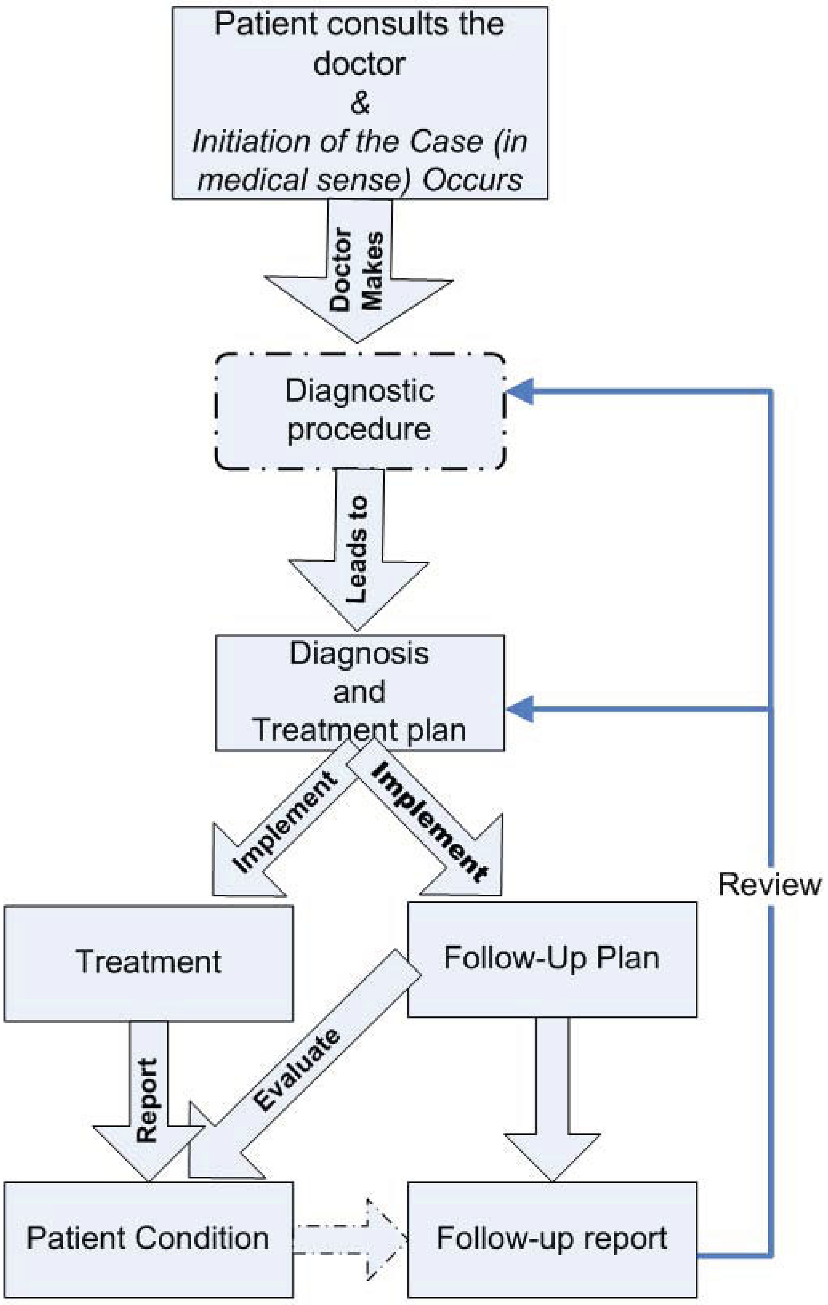


Fig. 1. Patient care process flow

a patient care process through different sub activities namely collection of past medical history, observing sign/symptom, suggesting tests, observing test results, confirming a diagnosis, prescribing treatment and pursuing a follow up. Thus different types of information like symptoms, medical history etc. are generated contextually in the execution of the sub-activities. The relevant information in context to a physician's interest is actually contained in six different medical entities namely past medical history, sign/symptom, test and results, diagnosis, treatment and follow up. These entities can be assumed to represent six different classes of information with some class specific or contextual characteristics which we explore and exploit to construct heuristic extraction rules. For example, the vocabulary, semantics and sentence format corresponding to sub-activities symptoms collection and medical test result composition are markedly different. The first type is a mixture of deep semantics related to feelings and observations on clinical events often forming different types of regular expressions, where as the second type is more prominent with typical medical vocabulary.

With the process flow, information piles up across the (sub)activity layers to form a medical case report. This fragmented view of a document through different contexts is the key motivation for compositional approach. Hence for extracting relevant information from a medical document we can map the relevance of the information artifacts with the underlying sub-activities and hence can form clear guidelines on the target set. Thus an approach that is composed of differentiated extraction task for individual types of information seems to be a natural choice.

From the activity point of view, a patient care process starts with patient coming to a doctor and continues through subsequent stages viz., diagnosis, treatment, follow up, review. Intermediately it iterates in the diagnosis-treatment-follow up-review-diagnosis or treatment-follow up-review-treatment cycle (or a cycle combining part/whole from these two cycles) until a conclusion (cure, patient quitting treatment or death of the patient) is reached. Thus the evolution of a patient care process described in terms of combination of sub activities, in perception of a physician's context, is natural and seems to be justified enough to work upon. The relevant information classes across different research reports created by physicians thus remain the same. There may be ornamental changes in presentation of the corresponding texts but the class specific characters stay close to the information class which can be exploited for information extraction.

It can be noted that different contexts may correspond to different types of information as relevant and hence the construction of the extraction rules will change. For example, from a pharmacists point of view, the medicines prescribed and the corresponding chemical groups might be of more interest. Here the consideration of the contextual nature of information within a document and with respect to a user is the differentiator of our extraction approach. Depending on the context a user (e.g. a physician) might be interested in specific information, like finding suitable tests given a set of symptoms or the set of treatments given a disease. The proposed approach of information extraction can be applied easily and efficiently to address such needs.

Thus any document arising from a patient care activity has the inherent structure consisting of six relevant information class with class specific characteristics which we intend to exploit for information extraction. This is explained in the subsequent discussions. It can be noted that the diagnostic procedure itself is made up of a complex flow of activities that we have not considered separately.

3 Related Work

Information extraction from medical document is a long standing area of research addressed by a mixture of research communities during past few decades. An excellent survey is available in [6]. Among the relevant papers, [17] uses a SVM based supervised model to annotate unseen terms in new texts and contexts based on manually annotated terms by domain experts. [21] presents an approximate dictionary-based biological concept extraction method where the basic idea is to capture the significant words rather than all words of a concept which is more related to biomedical field rather than a patient care scenario. In [20] document retrieval is done on the basis of concepts and their relations. The basic difference between our approach and these studies is that our focus is on relevant part within a document rather than the document as a whole.

The study in [8] has some similarity with our thought process but they explore more in terms of hidden relation extraction using conditional random field based approach. [9] uses a graphical model based on extension of the basic Latent Dirichlet Allocation framework for indexing PubMed abstracts with terminological concepts from ontology. Similar type of study consisting of sub-topic extraction is considered in [11]. Study conducted in [13] can be identified as a part of the whole scenario that we have considered here. In [13] a NLP application is designed to extract medical problems from narrative texts in clinical documents that come from a patient's electronic medical record.

[15] proposes a biological ontology (provided in UML S) based technique for extracting summarization of texts obtained from BioMed Central. [19] implements a medical Information Extraction (MedIE) system that extracts a variety of information from clinical medical records. Taking the help of section headings they perform ontology based extraction of medical terms, graph-based extraction of relations using link-grammar parser and text classification using ID3-based decision tree. [10] presents a framework for patient data extraction along the line of methodology describes in [19] with an automated storing process in a relational database.

It can be noticed that at micro-level the output of our approach is a collection of sentences belonging to different types of information related to the sub-activities in the process flow of patient care. Application for executing similar (micro-level) task, perceived mainly as passage level information extraction, is well discussed in literature, viz., [11,12,14,18]. But the integrated approach to combine them for a single purpose of IE from patient care data is little discussed in the existing literature which is the focus of this study and the value addition of this article. Also note that our integrated approach can harness any such technological/methodological advancement in re-usability context and increase the net value addition.

4 Compositional Information Extraction Method

The underlying activity flow of a patient care process illustrated in Fig. 1 is the motivation behind the compositional information extraction approach. As we have noticed, the activities involved in different levels of a patient care process generate different classes of information which are of interest to a typical user (e.g., a physician). Each of the class has some unique characteristic structure, type of key word and phrase, semantics, vocabulary and so on. Due to these profound interclass differences, a single holistic approach is not an appropriate way to address the problem. Instead we plan for differentiated modules, one for each class of information and combine them on a common platform for the ultimate execution. The information extraction process adopted in this paper is an integrated system of three parallel and mutually interacting building blocks:

1. Regular expression based pattern matching.
2. Dictionary based lookup and matching.
3. Heuristic based passage extraction algorithms.

We extract patient related information like age, gender etc. using regular expressions. Dictionary based lookup along with regular expression based pattern matching is used for identifying medical tests and test results. Regular expressions are required to identify test result related texts which contain numerical information separated by measuring units like 130/80 $mmHg$, 9.84 gm/dl etc.

Dictionary based lookup is the common approach used for identifying medical entities like disease, diagnosis, drugs, treatment, test and results, sign/symptoms and follow-up. A category specific dictionary contains a list of words related to the corresponding medical entity which can always be improved through domain expert intervention and hence the extraction efficiency can be improved as well. It is important to note that test, sign/symptom and past medical history information are many a times overlapping which need some methodology to differentiate. So by carefully analyzing the available case reports we develop heuristics to handle this.

In the next subsections we discuss information extraction methods for the class past medical history and follow up in detail as there are some typical complexities to handle.

4.1 Extracting Past Medical History Passages

Past medical history related information in the type of documents we have considered has some inherent characteristic which restricts the direct applicability of some reported approaches like in [7] and [16]. The authors of [7] propose a robust corpus-based approach for temporal analysis of medical discharge summaries.This learning based approach requires large trained corpus and will be *history pattern* = {for several years, was on therapy, ago, past, history, year previously, no previous, years previously, many years, when aged, week before, months earlier, years earlier, previous year, before admission, prior to admission, prior to presentation}

present pattern = {while, treated, initial, demonstrated, showed, confirmed, investigations, reveal, complained, initially, given, treatment, exam, diagnostics, presentation, received, arrived, admission, now, normal, diagnosed, was admitted, presented to, on along, presented with, admitted with, admittance, upon arrival, indicative, indicated, discharge, on arrival, yet, so far, shortly, presently, recently, follow up, meanwhile, within, physical examination, on physical examination}

Fig. 2. Patterns of History and Present text phrases

computationally very expensive when we just need to extract past medical history. The authors of [16] investigate four types of information found in clinical text that helps in understanding what textual features can be used in extraction of past medical history which are typically not much prominent in the research report format.

One of the basic problem in the documents we consider is that the semantics in these documents do not explicitly express past event related structure since the documents are written as reports of events observed some time back and hence the current context is also expressed in past format. Also precise chronological statement is not available; even if it is there it is mostly not in an explicit hierarchical order. Thus the usual approaches that rely on graphical representation of temporal events etc. have a limited or no scope of applicability. We use heuristic based approach to tackle this problem.

After analyzing the case reports we found that past medical history usually appears in first half of the case report. Also it may be as a group of sentences in the beginning or embedded within the case report. We use the spatial order of sentences present in the report to extract the past medical history. The narration of the case is assumed to be the present (on current state of patient) and anything cited before in time will be past history.

We use simple *Allen's temporal logic* [7] to find whether one sentence comes before, after or has no relation with other sentence to identify past medical history sentences and present sentences. We then identify frequent keywords found in both type of sentences and use them in extracting passages related to past medical history.

We use **Mafia** tool [1] along with a sliding window based algorithm *FindWordSeq* (Fig. 3) to find frequent text patterns or words found in past medical history and present sentences. Mafia tool is designed to find single frequent words which we utilize at the outset. Then after removing the duplicates we use these single frequent words in a sliding window based algorithm *FindWordSeq* to find frequent continuous sequence of words (up to size 3). Time complexity of *FindWordSeq* algorithm is $O(n)$, where n is number of words in a file. Note that

FindWordSeq Algorithm:

```
1. Find frequency of each word in the text /*store it in a Hashmap*/
2. Repeat step 3 for window size k= 1, 2,3
3. While (not end of text)
   (a) Use a sliding window of size k
   (b) If (all the words in the window are frequent) /*use Hashmap values*/
         – Output the word sequence
   (c) Read new text in window
```

Fig. 3. Algorithm to find Frequent Word Sequences

Identifying Frequent Word Patterns:

1. Use *Allen's Temporal Logic* to make two set of files – one set containing sentences belonging to past medical history and other set containing non past medical history sentences (present).
2. Extract frequent single words from the two set of files using Mafia tool.
3. Use algorithm *FindWordSeq* to find frequent continuous word sequences from the two set of files.
4. Remove duplicates from both the sets.
5. Remove common phrases present in both the sets.
6. Label the frequent text phrases of past medical history set files as *history pattern* and the frequent text phrases of non past medical history set files as *present pattern*.

Fig. 4. Identifying frequent word patterns of Past Medical History and Present Sentences

choice of Mafia tool is just for the sake of scalability and open access. Otherwise frequent single word finding can be addressed independently without any difficulty. After analyzing the available documents we found that existing frequent continuous word patterns are of maximum length 3, so we set the upper limit of sliding window size to 3. We also observed that inclusion of text patterns found by *FindWordSeq* improves the performance of the past medical history extraction module.

The process of finding frequent text patterns is explained in Fig. 4. The obtained frequent text phrases in past medical history sentences (*history pattern*) and the frequent text phrases in non past medical history sentences (*present pattern*) are shown in Fig. 2. We then use a simple heuristic based algorithm *ExtractHistory* (Fig. 5) to extract past medical history passages. Using the *history pattern* and *present pattern* set this algorithm exploits the chronological sentence order present in the case report to extract the past medical history of a patient. One single scan of the text provides the required information. Time complexity of this algorithm is linear and depends on the number of sentences.

ExtractHistory Algorithm:

1. Mark each sentence as history category sentence or present category sentence on the basis of phrases or words in *history pattern* and *present pattern* respectively.
   (a) If a sentence contains words in *history pattern* mark it as history category sentence.
   (b) If a sentence contains *present pattern* mark it as present category sentence.
   (c) If a sentence contains words both in *history pattern* and *present pattern* mark it as history category sentence.
2. The sentence belonging to history category sentence marks the beginning of past medical history (PMH). Go on including sentences in PMH until first present category sentence is encountered. (The sentences marked in step 1 helps in identifying the beginning and end of past medical history).
3. Repeat step 2 to find past medical history sentences that are found at different places in the report.

Fig. 5. Algorithm for Extraction of Past Medical History Sentences

4.2 Follow Up Text Passage Extraction

Two major problems of extracting follow up related information are notably less volume (in terms of sentence length and/or number of sentences) of it's description and non-explicitness of the structure. However the spatial information of it's location can help in the extraction. The follow up text passage is usually present at the end of a medical case report. We first find the most frequent keywords found in follow up. If a sentence contains these keywords then it is added to follow up passage. We also use the heuristic of including the sentences of last paragraph before *Conclusion* or *Discussion* section as follow up since by manual analysis we found them usually to be follow up sentences.

5 Experiments and Results

We use three softwares namely Eclipse [2], UIMA (Unstructured Information Management Architecture) [3] and MySQL [4] for developing the tool. Most of the coding is done in Java apart from some preprocessing tasks done using Perl. We use UIMA framework for implementing the algorithms as it provides a good platform for integrating different sub-modules for executing a complex task through combination of simpler sub-tasks. Also it provides support for important functions like entity annotation, regular expression annotation, POS tagging etc. For experimentation we use the heart disease related research reports collected from *Journal of Medical Case Reports* (a open source archive [5]). The collected data is first preprocessed to remove figures, links, references etc and then converted into simple text files. These files are supplied as input to the developed tool.

A typical medical case report is in general, but not limited to, a free text format description of a patient care event starting from past medical history related description followed by observations and discussions on symptoms/signs, medical tests and results leading to diagnosis, treatment details covering medication, surgery or different therapy and concluded by follow up. The typical features of such type of texts described in a research paper format in the *Journal of Medical Case Reports* are realized in the following aspects:

1. There are two broad sections containing the main case presentation and discussion but typically the information from different fields are entangled. One notable thing is that the follow up part is often found in the discussion part entangled with other informations not of interest to us.
2. The sequence of events describing a patient care process is not followed quite often and later issues are described first.
3. The domain dependence of the vocabulary.
4. Occurrence of numerical values with typical units and characters.
5. The whole report is written after the completion of treatment. Hence recognition of the past medical history related sentence/passage is very confusing.

We use training corpus to learn the regular expressions and other notable features to derive heuristic rules. The results of the manually tagged test corpus is compared with the system extracted results to judge the performance. We consider sentence as the unit level of comparison and compute Precision and Recall by comparing the system extracted sentences and the corresponding manual extraction. Since the information belonging to the different information classes are not distributed uniformly within and as well as between documents, a better way of performance measurements is to consider individual classes over the corpus and evaluate the performance separately for them. We use the macroscopic method of combining the results in which equal weight is given to all the samples and so, Precision and Recall values are averaged over all the test samples.

Table 1 presents the overall Precision, Recall and F measure values obtained over test corpus for classes Past Medical History, Sign/Symptom, Test and Test Results, Disease/Diagnosis, Treatment and Follow up. Note that F_1 measure is the harmonic mean of Precision and Recall measures while F_2 gives twice weightage to Recall and $F_{0.5}$ gives twice weightage to Precision. Definition of F_2 and $F_{0.5}$ measures are given below:

$$F_2 = (1 + 2^2) \cdot Precision \cdot Recall/(Precision + 2^2 \cdot Recall)$$

$$F_{0.5} = (1 + 2^2) \cdot Precision \cdot Recall/(2^2 \cdot Precision + Recall)$$

5.1 Discussion

Results show excellent performance for Diagnosis and good performance for Past Medical History. For Follow up performance is average. But for Signs/Symptoms, Test and Treatment performance is less than average. One major reason of poor

Table 1. Performance measure values for the six information classes

Class	Precision	Recall	F_1	F_2	$F_{0.5}$
Diagnosis	1	0.849206	0.918455	0.875614	0.965704
Signs/Symptoms	0.457173	0.412698	0.433799	0.420887	0.447528
Past Medical History	0.774376	0.588889	0.669014	0.61852	0.728485
Test and Results	0.714361	0.379107	0.49534	0.418376	0.607003
Treatment	0.5	0.39418	0.440828	0.411602	0.474522
Follow up	0.444709	0.620075	0.517951	0.574746	0.471371

performance for Sign/Symptom, Test and Treatment is that corresponding dictionaries do not cover all possible cases. We expect that consulting a domain expert will significantly improve the performance for these classes. It seems that we also need to apply natural language processing techniques for Sign/Symptom class where the patient's feeling related expression characterizes the texts heavily. Same applies for Follow up also because it is expressed through deep semantics and occupies comparatively very less portion in the case reports.

An obvious point arises here regarding the comparison of our approach with others. We would like to emphasize that there is hardly any room for such a study since our approach is a holistic one and the related literature, as discussed in section 3, are mostly focused towards extraction of a few specific information type from specific kind of medical documents. While the existing literature focuses on the efficient extraction of some atypical information from a document, we concentrate on the relevant extraction of information by first defining "relevance" contextually through characterization of document creation process and user's perspective. Thus when the context shifts from a physician's interest to that of a pharmacist, the relevance might lie in a few of the medical entities, mentioned before in section 2. For example a pharmacist's interest may be in the proposed medicines and the corresponding chemical groups if such information is contained in the text.

Another point to be noted is the performance aspect of our approach. There are scopes to improve the extraction performance by adapting techniques from Natural Language Processing or exploring inter class associations of the medical entities mentioned earlier. But we want to emphasize that the ingenuity of our approach lies in the perception of "relevance" of information through document creation process where information artifacts are generated stage wise in the evolution of a compound activity. The compound activity is carried out through execution of interrelated sub activities. Our aim is to propose the construction of a system that addresses relevant information extraction in a holistic manner. Once the system construction is addressed, performance enhancement can be achieved by adapting efficient extraction processes for individual module construction. Thus the paper is not intended to explore the achievement in performance/efficiency aspect but to introduce a novel idea.

6 Conclusion

In this paper we proposed a compositional approach for extracting necessary information for six different classes namely past medical history, sign/symptom, test and results, diagnosis, treatment and follow up from a free format medical case report emerging from a patient care process. We perceive a patient care process as a sequence of multiple sub-processes and the aforesaid information classes are considered to be outcome of different sub-process(es). We proposed a module based approach where each type of information is extracted using specific module(s) and the modules are integrated to work as a single composite system. The methodology is mainly based on heuristic approach that uses regular expressions, dictionary and different types of rules that are learned from the training corpus and the associated openly available resources. It can be noted that the ingenuity of the paper is the inception of a holistic and relevant IE approach from medical case research reports which is a kind of novel in it's class and hence not much scope for comparison with existing approach and/or study on the performance measure/enhancement is conceived.

Similar idea can be ported to any domain for extraction of relevant information from a free format text. Once we perceive the process flow of the events underlying the text generation, we can define the relevance as the informational artifacts generated at the granular process levels and set the information class and execute the subsequent stages to construct a composite system.

References

1. http://himalaya-tools.sourceforge.net/Mafia/
2. http://www.eclipse.org/
3. http://incubator.apache.org/uima/
4. http://www.mysql.com/
5. http://jmedicalcasereports.com/
6. Afantenos, S., Karkaletsis, V., Stamatopoulos, P.: Summarization from medical documents: a survey. Artif. Intell. Med. 33(2), 157–177 (2005)
7. Philip, B., Deshpande, P., Lee-and, Y.K., Barzilay, R.: Finding Temporal Order in Discharge Summaries. In: EMNLP (2006)
8. Bundschus, M., Dejori, M., Stetter, M., Tresp, V., Kriegel, H.-P.: Extraction of semantic biomedical relations from text using conditional random fields. BMC Bioinformatics 9(1), 207 (2008)
9. Bundschus, M., Dejori, M., Yu, S., Tresp, V., Kriegel, H.-P.: Statistical modeling of medical indexing processes for biomedical knowledge information discovery from text. In: BIOKDD 2008 (2008)
10. Han, H., Choi, Y., Choi, Y.M., Zhou, X., Brooks, A.D.: A Generic Framework: From Clinical Notes to Electronic Medical Records. In: CBMS 2006, pp. 111–118 (2006)
11. Hearst, M.A.: Multi-paragraph segmentation of expository text. In: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, pp. 9–16 (1994)
12. Mangold, C.: A survey and classification of semantic search approaches. Int. J. Metadata Semant. Ontologies 2(1), 23–34 (2007)

13. Meystre, S., Haug, P.J.: Natural language processing to extract medical problems from electronic clinical documents: Performance evaluation. J. of Biomedical Informatics 39(6), 589–599 (2006)
14. Mooney, R.J., Bunescu, R.C.: Mining knowledge from text using information extraction. SIGKDD Explorations 7(1), 3–10 (2005)
15. Morales, L.P., Esteban, A.D., Gervás, P.: Concept-graph based biomedical automatic summarization using ontologies. In: TextGraphs 2008, pp. 53–56 (2008)
16. Mowery, D.L., Harkema, H., Dowling, J.N., Lustgarten, J.L., Chapman, W.W.: Distinguishing historical from current problems in clinical reports: which textual features help? In: BioNLP 2009, pp. 10–18 (2009)
17. Takeuchi, K., Collier, N.: Bio-medical entity extraction using support vector machines. Artif. Intell. Med. 33(2), 125–137 (2005)
18. Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic annotation for knowledge management: Requirements and a survey of the state of the art. Web Semantics 4(1), 14–28 (2006)
19. Zhou, X., Han, H., Chankai, I., Prestrud, A., Brooks, A.: Approaches to text mining for clinical medical records. In: SAC 2006, pp. 235–239 (2006)
20. Zhou, X., Hu, X., Lin, X., Han, H., Zhang, X.-d.: Relation-Based Document Retrieval for Biomedical Literature Databases. In: Li Lee, M., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, pp. 689–701. Springer, Heidelberg (2006)
21. Zhou, X., Zhang, X., Hu, X.: MaxMatcher: Biological concept extraction using approximate dictionary lookup. In: Yang, Q., Webb, G. (eds.) PRICAI 2006. LNCS (LNAI), vol. 4099, pp. 1145–1149. Springer, Heidelberg (2006)

A Framework for Semantic Recommendations in Situational Applications

Raphaël Thollot[1] and Marie-Aude Aufaure[2]

[1] SAP BusinessObjects, 157 rue Anatole France, 92309 Levallois-Perret, France
raphael.thollot@sap.com
[2] Ecole Centrale Paris, MAS laboratory, 92290 Chatenay-Malabry, France
marie-aude.aufaure@ecp.fr

Abstract. Information overload is an increasingly important concern as users access and generate steadily growing amounts of data. Besides, enterprise applications tend to grow more and more complex which hinders their usability and impacts business users' productivity. Personalization and recommender systems can help address these issues, by predicting items of interest for a given user and enabling a better selection of the proposed information. Recommendations have become increasingly popular in web environments, with sites like Amazon, Netflix or Google News. However, little has been done so far to leverage recommendations in corporate settings. This paper presents our approach to integrate recommender systems in enterprise environments, taking into account their specific constraints. We present an extensible framework enabling heterogeneous recommendations, based on a semantic model of users' situations and interactions. We illustrate this framework with a system suggesting structured queries and visualizations related to an unstructured document.

1 Introduction

Information overload is a well-known issue in information systems, and users face the difficult task of choosing among many existing sources those likely to satisfy their needs. Not only more and more data are generated by a profusion of web sites and services, but users access it in new diversified ways, e.g., using multi-function mobile devices. Therefore, systems need to be developed to facilitate the selection of information adapted to the current user context.

These considerations have partly driven two related research areas, recommender systems and context awareness. The first one aims at predicting items of interest for a given user, and the second intends to adapt information and services to the current user's environment. Our work aims at defining a framework bringing these two approaches together.

1.1 Recommender Systems: Web and Enterprise Perspectives

Recommender systems (RS) are considered a major technological trend in both industrial and academic environments. The goal of a recommender system (RS) is to provide personalized recommendations to improve and accelerate the user's navigation in a vast space of available resources. Techniques have been developed to serve this purpose, using similarity measures between items, users and user-item pairs. These techniques can mainly be categorized in content-based (CB) and collaborative filtering (CF) approaches. The CB approach considers a user may be interested by items similar to those he liked in the past. On the other hand, CF techniques consider that a user may appreciate items that other users with similar tastes have appreciated. The two approaches are often combined into hybrid systems to address their respective shortcomings [2].

Recommender systems have become popular thanks to web sites like Amazon, Netflix or Google News. In e-commerce contexts, these techniques directly benefit both the user – with an improved navigation – and the service provider – by encouraging more sales from each customer. To the best of our knowledge, little has been done to apply RS techniques in corporate environments, where users are employees of the same company, even though such systems could positively impact their productivity. We consider that web and corporate environments are often very different given the nature of resources, business users' needs and security constraints. To best serve business users' needs, a corporate RS should enable recommendations of very heterogeneous resources like a customer from a CRM system, an application on a company's Intranet, a common process to follow, experts who might help, etc. Besides a lot of work has been done in web settings to leverage users' feedback coming through ratings, which we believe are difficult to generalize to varied corporate resources. For instance, users are not likely to rate personal emails or processes as they would rate cultural products on a web site. Moreover, some special cases are bound to raise critical privacy or political issues (e.g., employees rating themselves or their manager).

1.2 Context-Awareness and Situational Applications

Initially considered in so-called "intelligent environments", e.g., to deal with data acquired from sensors, the notion of context has since been used with varying definitions. Dey defines the context as the state of any entity that impacts the interaction between the application and the user [7]. This definition highlights the fact that context cannot be limited to device adaptation or location sensitivity.

As opposed to the experience of a user limited to one given web site inside a browser, a business user evolving inside a company's network may interact in varied ways with several communicating applications. Ideally, these applications would share the same knowledge and representation of the user's interests and intentions to better assist her and ease inter-operability. Such user-centric applications are called *situational applications* and they emphasize the need to leverage a shared representation of users, resources and their interactions. Various context-aware systems have been developed and presented in the literature, most of them using specific user and context models. Unfortunately, sharing these models has not necessarily been a priority, which lead to many distinct semantics and little shared understanding.

1.3 Motivations of Our Work

We briefly presented the growing interest for recommender systems on the web, and we consider such systems would be very valuable if applied in corporate environments. However, doing so means taking a certain number of constraints into consideration:

Resources and interactions heterogeneity. Resources of interest for a business user are bound to come from various sources and be of very different nature, be it documents, applications, structured data, etc. Interactions, or more generally relations between resources may also carry different semantics (user1 *is reading* document1, document1 *written by* user2, user1 *manages account* customer1, document1 *mentions* customer1, etc.).

Applications inter-operability. Users exploit different applications to perform their daily activities and these applications rarely share much of the user's knowledge, which leads for instance to unnecessary repeated input or incomplete users' profiles and context models.

Security constraints. It is frequent, in web settings, that all resources can be recommended to a user. On the other hand, this is rarely possible with corporate resources where security constraints are often more complex (e.g., critical data from a finance or human resources department is not available to everybody).

Recommendations usually base on some representation of a user's profile (preferences, history, etc.). Context-aware systems are complementary and go beyond preferences to characterize dynamic interactions with available resources.

Given the broad range of resources to consider and the importance of sharing models between applications to enable inter-operability (especially in a corporate environment) such system requires the definition of a common meta-model to describe users and their environments.

Multiple recommender systems have been developed to serve different needs, and it is hard to consider that one RS could be generic enough to serve relevant recommendations for any type of resource.

In response to these challenges, the work described in this paper brings the following contributions: (a) We define a framework for secure recommendations in corporate environments. (b) Varied resources and interactions between them are dynamically and homogeneously represented with semantic web technologies, enabling inter-operability. (c) Specific recommendations may be developed thanks to the framework modularity. We experiment our framework by implementing a recommendation scenario in a Business Intelligence (BI) context, successfully combining the semantics of different models (BI queries, security rules, entities extracted in an unstructured document).

2 Situation Modeling

Situational applications base on a situation model, defined by Heckmann as the combination of a user model, a context model and a resource model [10]. In both web and corporate settings, interactions between resources can be manifold and we believe their semantics is crucial to interpret users' interests. This section describes our approach to provide a homogeneous representation of available resources and their interactions thanks to a common graph-based meta-model.

2.1 Graph Repository

A corporate recommender system should integrate knowledge and resources from various source systems (CRM, social network, BI data, etc.). We consider a common unifying ground is necessary to achieve this, and graphs are a natural and general representation. We thus rely on a simple graph-based meta-model describing different types of available nodes and relations in an RDFS schema noted S_{base}. Building and maintaining the complete graph of available resources would be extremely costly, which leads to adopt a modular approach. The Graph Repository GR can thus be seen as an aggregation of several partial graphs $G_i = (N_i, E_i)$, defining nodes N_i and edges E_i identified by URIs. Each graph G_i is populated and maintained by a provider $P_i = (G_i, S_i)$. S_i is the custom schema that P_i may declare (when registering to the repository) to define specific types, extending S_{base}. To enable schema-based reasoning, the complete schema S is obtained by merging provider-specific schemas: $S = S_{base} \cup (\bigcup_{i=1}^{m} S_i)$. Eventually, with $G = \bigcup_{i=1}^{m} G_i$, the graph repository can be noted $GR = (G, S)$.

Many providers can be considered, for instance to describe the social network between employees (e.g., from an LDAP directory) or relations between account managers and customers (from a CRM system). However, most corporate systems impose security constraints like access-control rules. Therefore, providers may have to authenticate on remote systems either with the credentials of a given user, or using a specific account if more access rights are necessary (and granted by the IT department).

2.2 Situation Statements

Heckmann et al. introduced *situation statements*, a data structure used to represent the unit of information in situation modeling [9]. Our work bases on this structure to elaborate on an aggregated and dynamically maintained representation of a user's situation. This section briefly introduces situation statements, which have been described in more details in [10].

Situation statements have been defined as extended triples representing assertions formed of a subject S, a predicate P and an object O, like 'Maggie *reading* document1'. Additional metadata M integrate temporal restrictions, privacy settings, origin and confidence. A statement can then be noted as a tuple (S, P, O, M), with $M = (origin, t, l, s, c)$. The *origin* indicates which *agent* created this statement. We define *agents* as resources of the system which can create statements (providers, users, operators, client applications, etc.). The timestamp t and the expected lifetime l are used to handle time-related validity constraints. The privacy setting s indicates whether this statement is public, private or has custom access authorizations. Finally, the confidence attribute c allows agents

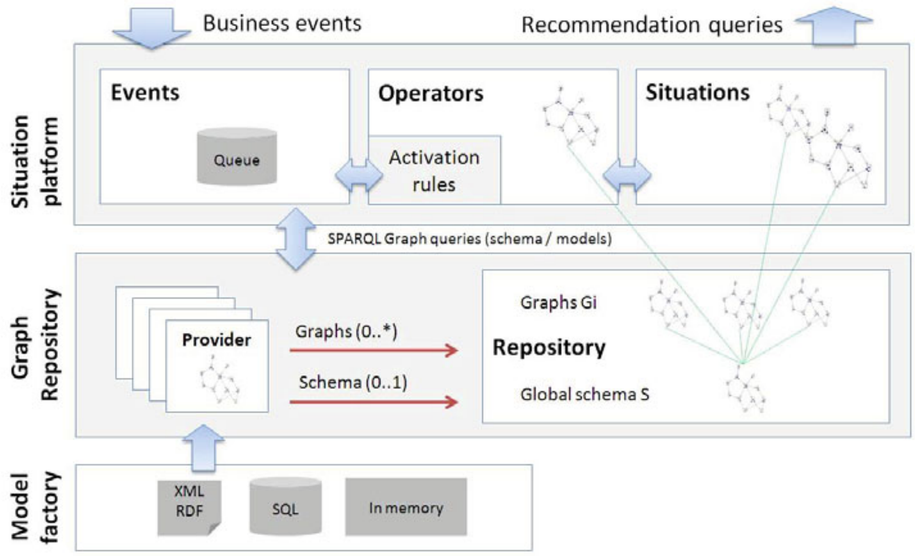


Fig. 1. Architecture overview: main components of the situation management platform

– like operators described in section 3.4 – to qualify the reliability level of the statement, which is key in distributed environments (e.g., situational applications inside a corporate network). A statement (S, P, O, M) is an extended type of relation between two resources S and O of the repository GR. This approach differs from the one proposed by Heckmann et al., where distributed ontologies (including GUMO [11]) were used to represent resources without security restrictions.

In a statement, the predicate P is crucial since it defines the semantics of the interaction between S and O. The expected lifetime attribute can be overriden for each statement, but the predicate P determines its default value.

At a given point in time, the situation of a user can be defined as the consolidation of valid statements. The user's situation finally takes the form of a graph, centered on the user to reflect its particular importance [3, chap. 2]. The next section presents the overall architecture of a situation management platform, handling among other things dynamic aspects.

3 Situation Platform

We presented our approach to model situations using a uniform data structure called situation statement, in conjunction with an extensible graph repository. This section presents the high-level architecture of our situational platform and its major components.

3.1 Architecture Overview

The role of the situation platform is to continuously maintain a complete and consistent view of users' situations. Doing so implies a constant monitoring of interactions between resources, and actions to reflect these events in situations graphs. Required operations are performed thanks to a set of core components, illustrated in Figure 1 and described below.

Graph repository. This repository is the central component, referencing all partial graphs managed by registered providers (see section 2.1). Models are built by a factory which allows and controls the creation of graphs backed by SQL databases, by XML/RDF files or simply held in memory.

Situations. The situation management component mainly consists of a cache keeping graphs of monitored situations in memory, or initializing them from stored statements on a per-need basis.

Business events. An event defines an interaction between two resources and can be seen as a specific kind of statement. Events are sent to the platform which queues them up so they can be further analyzed asynchronously.

Activation rules. Rules are defined to react to events and activate specific operators with determined parameters. These rules may express constraints on the event itself and information present in the *graph repository.*

Operators. Operators are used to perform core operations on situations graphs. They can add new statements or update/delete existing ones. The way operators are used and applied in response to events is determined by a set of activation rules.

3.2 Business Events – Situation Dynamics

Business events are the crux of dynamic situation management, they describe interactions between resources. Therefore, they are very similar to statements. Events can be external as well as internal (raised by the platform itself). External events are sent by client situational applications to contribute to and benefit from the more complete, aggregated view of a user's situation (including recommendations).

The situation platform reacts to these business events to continuously update situations. Resulting operations or actions may require a long processing time and need to be run asynchronously. Events are thus prioritized and queued.

The platform exposes an interface through which events can be sent and added to the queue. The event 'Maggie *reading* document1' can be described as a statement and sent to the platform. Additional statements may be embedded to enrich the event description (e.g., document1 *sentBy* John, Maggie *hasMeeting* meet1, meet1 *hasParticipant* John, etc.). This relies on a uniform event description using the RDF-S schema defined by the graph repository. The listing below is the RDF representation of the previous event.

```
<rdf:RDF xmlns:rdf="..." xmlns:rdfs="..." xmlns:repo="http://.../grepo/1.0#">
  <rdf:Description rdf:nodeID="A0">
    <rdf:type rdf:resource="http://.../#MeshStatement"/>
    <repo:hasSource rdf:resource="http://.../users/Maggie"/>
    <repo:hasPredicate rdf:resource="http://.../predicates/read"/>
    <repo:hasTarget rdf:resource="http://.../document1"/>
    <repo:hasOrigin rdf:resource="http://.../clients/EmailClient_Plugin"/>
    <repo:priority>2</repo:priority>
  </rdf:Description>
  <rdf:Description rdf:about="http://.../clients/EmailClient_Plugin">
    <rdfs:label>EmailClient_Plugin</rdfs:label>
    <rdf:type rdf:resource="http://.../#Agent"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://.../users/Maggie">
    <rdf:type rdf:resource="http://.../#UserAgent"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://.../document1">
    <repo:content>Our revenue target was not reached...</repo:content>
    <rdfs:label>document1</rdfs:label>
    <rdf:type rdf:resource="http://.../#UnstructuredDocument"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://.../predicates/read">
    <rdf:type rdf:resource="http://.../#StatementPredicate"/>
  </rdf:Description>
</rdf:RDF>
```

The next two sections present activation rules and operators which can be added to the platform to evaluate and respond to events with different semantics.

3.3 Activation Rules

In event-driven architectures, the *Event-Condition-Action* (ECA) structure can be used to define active rules, with clear declarative semantics [5]: `ON` ***`event`*** `IF` ***`condition`*** `DO` ***`action.`*** Rules are used to react to specific events and trigger additional processing, if some conditions are met.

We adopt the ECA structure to define activation rules. The *event* part is used to filter events to which the rule applies. The *condition* part defines additional conditions that need to be satisfied. Finally, the *action* part defines operations to perform when the rule is activated. This approach is similar to the one presented by Beer et al. with their CAIPS system [4].

Below is an example of rule used to activate a stemming operator when a user interacts with an unstructured document (e.g., an email). This may be used before applying other operators related to text processing (for instance an operator performing Named Entity Recognition, NER).

```
<rule>
  <event>
    eventUser:=(SUBJECT HAS_TYPE "http://.../#UserAgent")
    AND eventDoc:=(OBJECT HAS_TYPE "http://.../#UnstructuredDocument")
  </event>
  <condition >
    NOT(EXISTS(s,
      (s.SUBJECT IS "http://.../operators/stemming")
      AND (s.PREDICATE IS "http://.../predicates/has_processed")
      AND (s.OBJECT IS eventDoc))
  </condition>
  <action>
    CALL "http://.../operators/stemming"
      WITH_PARAMS eventDoc
      ON_BEHALF_OF eventUser
  </action>
  <description>
    Stems an 'unstructured document' a 'user agent' interacts with
  </description>
</rule>
```

In this example, the *event* block filters events to keep only those stating that a user agent (or a node of type user agent) is interacting with an unstructured document.

The *condition* is an execution guard, it avoids to call multiple times the stemming operator on the same document. This aspect – and more generally dependencies between operators – is presented in more details in the next section. In our case, rules conditions can be expressed as queries on the graph repository.

The *action* part indicates the activated operator (stemming) and parameters to pass through. The complete call syntax is `CALL` *`operator`* `WITH_PARAMS` *`params...`* `ON_BEHALF_OF` *`user`*. The list of parameters *params* results from the event-filtering and condition-evaluation process, using bound variables. In the above example, the only parameter passed is the document mentioned by the event, and operations will be performed on behlaf of the concerned user. The operator will thus be able to access and operate on this user's situation. The next section discusses operators in more details.

3.4 Operators and Recommendations

Operators are key components which maintain situations graphs by adding, updating or deleting statements. These operations represent either explicitly declared facts (e.g., Maggie *reading* document1) or knowledge resulting of additional processing (for instance a NER result, like document1 *mentions* entity37).

The situation platform manages a pool of registered operators which can be developed to interpret the specific semantics of various providers and/or events. Many operators can be considered to perform a broad range of tasks. Table 1 lists examples of operators implemented for the experimentation scenario, discussed in section 4.

Table 1. Examples of operators. This include system operators and specific ones related to the experimentation (see section 4).

Operator	**Description**
Expiry checker (system)	Checks statements of a given situation to determine and remove those that are outdated. Outdated statements are stored as historical data.
Consistency checker (system)	Conflicting statements have to be resolved to ensure the situation view is coherent.
Stemmer	Applies stemming to unstructured resources.
NER	Performs Named Entity Recognition on text resources, extracting entities with various dictionaries. Extracted entities are mapped to objects of the data warehouse the user can see.
Query recommendation	Combines results of entity extraction with the semantics of a business domain to suggest meaningful queries.

The previous section described how activation rules are used to trigger operators with dynamic parameters. We distinguish two types of operators based on the way they are invoked. The first category includes generic operators, simply called with the `CALL` *`operator`* `WITH_PARAMS` *`params...`* command. Operators of this category do not access any specific user's situation and the `ON_BEHALF_OF` part of the command is not necessary. On the other side, the second category contains *situation operators*, that is operators which perform their actions on behalf of a dynamically determined user. Such operators are invoked with parameters as well the concerned user's situation.

Some tasks can be performed independently, but two operators may also declare mutual dependencies. For instance, an operator performing Named Entity Recognition may require that stemming has been applied first to increase the recall. Such dependencies can be expressed using conditions in rules, checking the existence of a given statement. In the previous example, the stemming operator indicates it has processed a document by adding the statement stemmingOperator *hasProcessed* document1. The activation rule for the NER operator can impose the dependency by simply adding a condition to check if this statement exists.

Situation operators are distinguished from others since they can modify users' situations. Among them, some may contribute specific statements interpreted as recommendations (e.g., document2 *suggested for* Maggie). Like all situation operators, recommendation operators benefit from the user's situation view, the semantics of which can be used to better personalize and dynamically adapt recommendations made to a user.

4 Experimentation

In this section we present results obtained with a prototype implementation of the situation platform. We illustrate the applicability and interest of the approach by integrating a first recommender system, Text-To-Query (T2Q) [20].

This system suggests structured queries to be executed on a corporate data warehouse, related to a text document.

4.1 Prototype Platform Implementation

Graph Repository. The architecture presented in section 3.1 strongly relies on the graph repository, populated by various providers. The implementation enables the uniform data representation using RDF and RDFS, backed by the java library Jena. Providers create and populate graphs through a factory, enabling in-memory, file-based or SQL-backed storage of data.

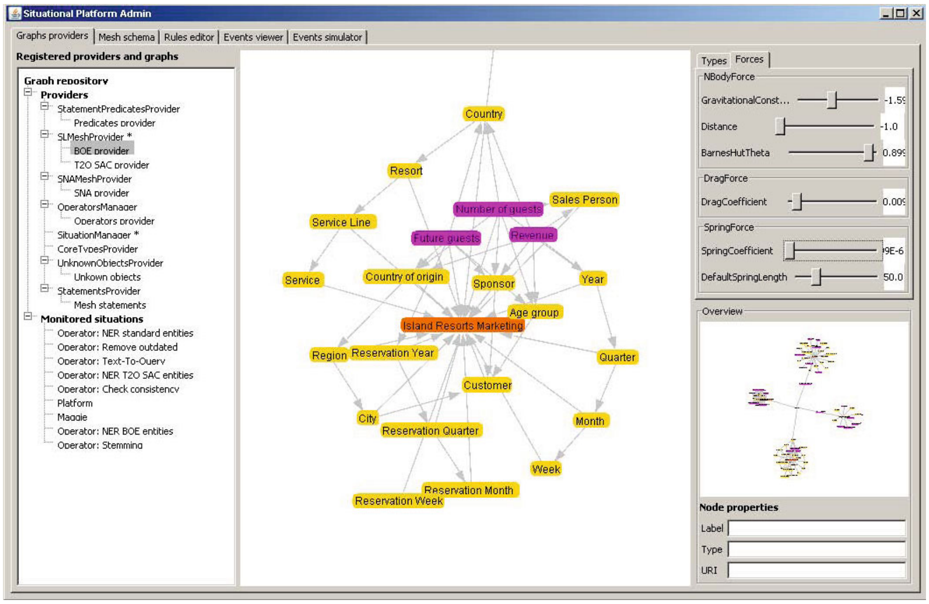
The repository gives an access to data from registered providers. Figure 2 illustrates the graph viewer of the platform administration UI. It also exposes the complete RDFS schema obtained by merging provider-specific schemas in the core repository schema. Our implementation leverages interesting functionalities coming with semantic web technologies, e.g., merging RDF graphs (e.g., to aggregate custom schemas defined by providers) and SPARQL-querying. We also use simple RDFS reasoning at the schema level, e.g., for subclass and subproperty relationships.

Extensibility and Operations. The extensibility of the framework is an important aspect of the platform. First, providers which create and populate graphs can be plugged in the repository to contribute more entities and relations. In future work, we will focus on improving storage and querying scalability with a distributed database approach, e.g., using the Apache Cassandra project. This strategy is enabled using a factory pattern, which allows the implementation of custom storage adapters.

Operators are themselves treated as modules and dependencies between them allow a good division of tasks, encouraging reusability. Operators – in particular those providing recommendations – can leverage the user's situation semantics to perform their specific operations. However, since graphs are a very general representation, operators can still implement other recommendation techniques like content-based matching or collaborative filtering. Content-based methods can leverage attributes associated to nodes and relations to represent various things like keywords, title, etc. Besides, a graph can easily be represented as a matrix, which opens the door for more statistical processing, for instance collaborative filtering techniques.

4.2 Text-To-Query: An Example of Recommendation Operator

T2Q is a system that dynamically generates structured queries to be executed on a corporate data warehouse, related to a text document. T2Q leverages models of business domains (defining measures and dimensions) in a data warehouse to automatically create dictionaries of entities. These dictionaries are used at runtime to perform entity extraction, and results can be combined to form meaningful queries. The complete recommendation process has been presented in more details in [20], and we illustrate here more specifically the operator-oriented implementation.



Registered providers and graphs

- Graph repository
  - Providers
    - StatementPredicatesProvider
      - Predicates provider
    - SLMeshProvider *
      - BOE provider
      - T2O SAC provider
    - SNAMeshProvider
      - SNA provider
    - OperatorsManager
      - Operators provider
    - SituationManager *
    - CoreTypesProvider
    - UnknownObjectsProvider
      - Unkown objects
    - StatementsProvider
      - Mesh statements
  - Monitored situations
    - Operator: NER standard entities
    - Operator: Remove outdated
    - Operator: Text-To-Query
    - Operator: NER T2O SAC entities
    - Operator: Check consistency
    - Platform
    - Maggie
    - Operator: NER BOE entities
    - Operator: Stemming

NBodyForce

GravitationalConst...	-1.59
Distance	-1.0
BarnesHutTheta	0.899

DragForce

DragCoefficient	0.009

SpringForce

SpringCoefficient	9E-6
DefaultSpringLength	50.0

Node properties

Label	
Type	
URI	

Fig. 2. Graph viewer showing some objects of the data warehouse and (functional) dependencies between them

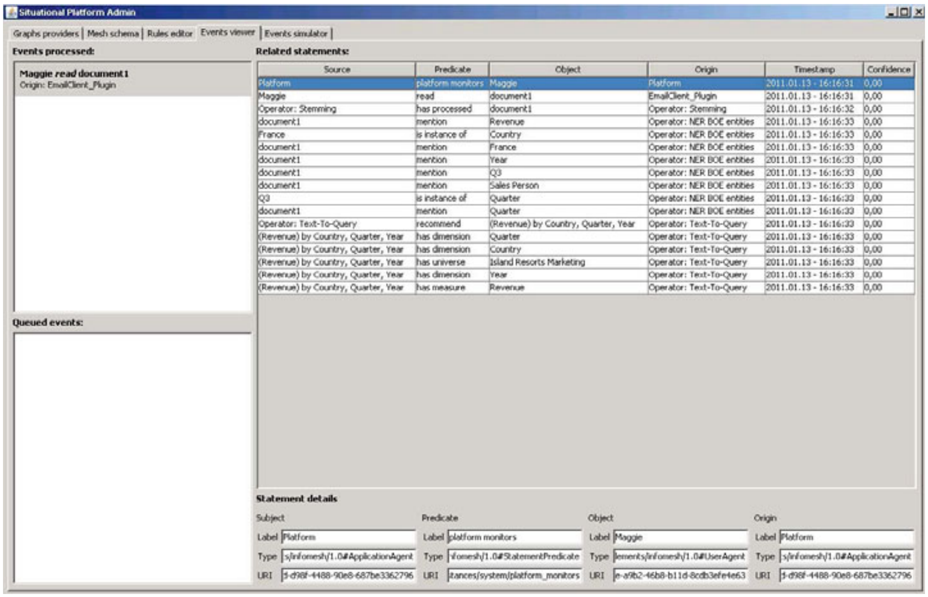
From the provider point-of-view, T2Q bases on a representation of data warehouse objects defined in business domains models. A provider has been developed to populate a graph describing measures and dimensions of a business domain, and their mutual dependencies. The graph viewer of Figure 2 displays these objects and dependencies. This provider requires users credentials to a Business Intelligence platform, so authentication and security rights can be applied. Then, at runtime, recommendation operations can be divided and implemented with the following operators, listed in Table 1:

(a) Stemming. Stemming is applied to the text to increase recall of entity extraction. We used the stemming technology provided by the BusinessObjects Text Analysis SDK. This SDK also provides a Part-Of-Speech tagger but we do not leverage it in our solution.

(b) NER and mapping. Named Entity Recognition is applied to the stemmed text. The BusinessObjects Text Analysis platform exposes entity extraction services allowing standard and custom dictionaries to be used. Extracted entities are mapped to objects of the data warehouse the user can see.

(c) Query recommendation. Data warehouse objects are combined using their semantics (hierarchies and functional dependencies) to propose meaningful queries and visualizations.

These operations produce a certain number of statements, as illustrated in Figure 3. The left column lists processed and queued events. When a processed



Events processed:

Maggie *read* document1
Origin: EmailClient_Plugin

Queued events:

Related statements:

Source	Predicate	Object	Origin	Timestamp	Confidence
Platform	platform monitors	Maggie	Platform	2011.01.13 - 16:16:31	0,00
Maggie	read	document1	EmailClient_Plugin	2011.01.13 - 16:16:31	0,00
Operator: Stemming	has processed	document1	Operator: Stemming	2011.01.13 - 16:16:32	0,00
document1	mention	Revenue	Operator: NER BOE entities	2011.01.13 - 16:16:33	0,00
France	is instance of	Country	Operator: NER BOE entities	2011.01.13 - 16:16:33	0,00
document1	mention	France	Operator: NER BOE entities	2011.01.13 - 16:16:33	0,00
document1	mention	Year	Operator: NER BOE entities	2011.01.13 - 16:16:33	0,00
document1	mention	Q3	Operator: NER BOE entities	2011.01.13 - 16:16:33	0,00
document1	mention	Sales Person	Operator: NER BOE entities	2011.01.13 - 16:16:33	0,00
Q3	is instance of	Quarter	Operator: NER BOE entities	2011.01.13 - 16:16:33	0,00
document1	mention	Quarter	Operator: NER BOE entities	2011.01.13 - 16:16:33	0,00
Operator: Text-To-Query	recommend	(Revenue) by Country, Quarter, Year	Operator: Text-To-Query	2011.01.13 - 16:16:33	0,00
(Revenue) by Country, Quarter, Year	has dimension	Quarter	Operator: Text-To-Query	2011.01.13 - 16:16:33	0,00
(Revenue) by Country, Quarter, Year	has dimension	Country	Operator: Text-To-Query	2011.01.13 - 16:16:33	0,00
(Revenue) by Country, Quarter, Year	has universe	Island Resorts Marketing	Operator: Text-To-Query	2011.01.13 - 16:16:33	0,00
(Revenue) by Country, Quarter, Year	has dimension	Year	Operator: Text-To-Query	2011.01.13 - 16:16:33	0,00
(Revenue) by Country, Quarter, Year	has measure	Revenue	Operator: Text-To-Query	2011.01.13 - 16:16:33	0,00

Statement details

	Subject	Predicate	Object	Origin
Label	Platform	platform monitors	Maggie	Platform
Type	s/infomesh/1.0#ApplicationAgent	ifomesh/1.0#StatementPredicate	lements/infomesh/1.0#UserAgent	s/infomesh/1.0#ApplicationAgent
URI	1-d98f-4488-90e8-687be3362796	tances/system/platform_monitors	e-a9b2-46b8-b11d-8cdb3efe4e63	1-d98f-4488-90e8-687be3362796

Fig. 3. Events viewer showing statements resulting of an event processing

event is select, statements resulting of its processing are displayed in the grid at the right hand side. For each statement, the following fields/attributes are listed: source, predicate, object, origin, timestamp and confidence. Some of these statements were added (by operators) to Maggie's situation. The resulting situation graph is shown in Figure 4.

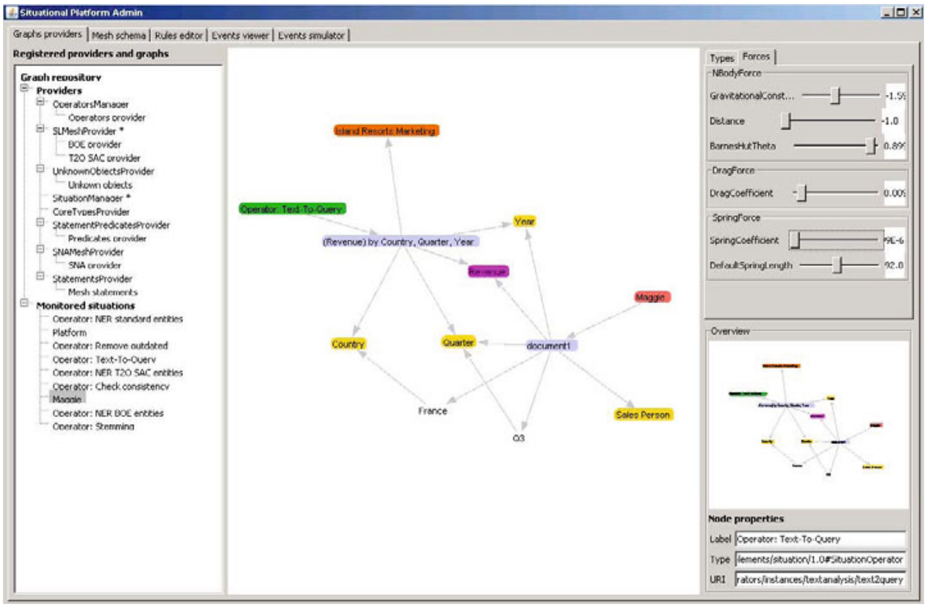
T2Q presents a first recommendation scenario using purely graph-based semantics. However, as mentioned before, the platofrm allows more operators to be developed and integrated so diversified recommendations can be produced. These recommendations may also leverage content-based or collaborative filtering techniques. Moreover, the modular approach enables reusability of previously implemented operations.

5 Related Work

This section presents some related works in the following research areas: context-awareness, situation modeling, personalization and recommender systems.

5.1 Context-Awareness and Situation Modeling

Research around context-awareness and pervasive or ubiquitous systems has been very active. These systems are meant to provide information and services to users, adapted to their context. Several architectures have been presented for developing context-aware systems [8][21]. These systems leverage different types of context models. Some used simple key-value pairs which lack semantics,


Situational Platform Admin
Graphs providers | Mesh schema | Rules editor | Events viewer | Events simulator
Types | Forces
Island Resorts Marketing
Operator: Text-To-Query
(Revenue) by Country, Quarter, Year
Year
Revenue
Maggie
Country
Quarter
document1
France
Sales Person
Q3
Overview

Registered providers and graphs

- Graph repository
  - Providers
    - OperatorsManager
      - Operators provider
    - SLMeshProvider *
      - BOE provider
      - T2O SAC provider
    - UnknownObjectsProvider
      - Unkown objects
    - SituationManager *
    - CoreTypesProvider
    - StatementPredicatesProvider
      - Predicates provider
    - SNAMeshProvider
      - SNA provider
    - StatementsProvider
      - Mesh statements
  - Monitored situations
    - Operator: NER standard entities
    - Platform
    - Operator: Remove outdated
    - Operator: Text-To-Query
    - Operator: NER T2O SAC entities
    - Operator: Check consistency
    - Maggie
    - Operator: NER BOE entities
    - Operator: Stemming

NBodyForce
GravitationalConst... -1.5
Distance -1.0
BarnesHutTheta 0.89
DragForce
DragCoefficient 0.00
SpringForce
SpringCoefficient 9E-6
DefaultSpringLength 92.0

Node properties
Label Operator: Text-To-Query
Type lements/situation/1.0#SituationOperator
URI rators/instances/textanalysis/text2query

Fig. 4. Graph of Maggie's situation after the event 'Maggie *read* document1' and activated operators

others, like [13], relied on object models which may be hard to share between client applications. Finally, more recent works have proposed ontology-based models which enable a uniform representation (e.g., using RDF) and improve reasoning [22][16].

The notion of context can be very broad [14], but Dey defines it as any information characterizing the state of an entity that impacts the interaction between the application and the user [7]. The type of device and geographical position are often used parameters, as in [14]. However, beyond these simple parameters, the previous definition highlights the importance of capturing interactions.

Heckmann et al. reconciliated user, context and resources models into the notions of situation [10] and situated interaction [9]. They introduced situation statements which we leverage in our solution. In our work, we try to go beyond the static view of a situation at a given point in time and we elaborate on dynamic aspects.

5.2 Personalization and Recommender Systems

Personalization plays an important role in recommender systems. To provide user-dependent recommendations, these systems need to capture some knowledge about the current user, his hobbies, interests, goals, etc. The user profile is then a core piece of Generic User Modeling Systems, thoroughly reviewed in [15].

It is necessary for a recommender system to be able to compare two resources, two users or estimate the relatedness between a resource and the user's profile.

Most recommender systems fall either in content-based (CB) or collaborative filtering (CF) approaches [18]. The latter approach has been later refined to exploit a trust network between users, like in TruskWalker [12]. This allows narrowing the collaborative contribution to the current user's neighborhood, supposedly more relevant. On the other hand, a content-based system tries to maximize the similarity between the user's interests and the content of its suggestions [3]. Available resources can be of very different types and similarity measures have to be defined to determine how close two of them are. Designed measures are thus very specific to the chosen representation for the information content. In our work, we address this heterogeneity issue with the graph repository, exposing a common meta-model.

Based on the underlying recommendation method, different techniques have been considered to introduce context sensitivity in recommender systems. From the CF perspective, Adomavicius proposed the extension of the user-item space with additional context dimensions, leading to multi-dimensional approaches [1]. In CB systems, items are represented through a set of descriptive features. The user context can help determine the most interesting features [17]. Most importantly, our approach brings a unified model towards situation and resources modeling, adapted to multiple recommendation techniques.

6 Conclusion and Future Work

In this paper, we have presented a modular framework enabling recommendations for situational applications in corporate environments. The situation platform bases on a homogeneous representation of varied resources in a graph repository, populated by secure providers. This platform dynamically maintains situation graphs, interpreting events thanks to a set of operators. Additional operators can easily be implemented and registered to the platform to perform varied tasks. In particular, recommendation operators benefit from dynamic and homogeneous situation models to enable personalized suggestions.

The proposed framework was first experimented by implementing Text-To-Query, a recommender system suggesting queries and visualizations related to a text. This experimentation illustrated the importance of the framework extensibility and modularity, encouraging new attempts to diversify recommendations and refine situation models. In future work, we will focus on pursuing the experimentation by extending recommendation use cases. In particular, it seems promising to consider the interpretation of situation semantics in conjunction with statistical approaches (e.g., content-based or collaborative-filtering techniques). In order to reach the scalability required with diversified scenarios, we will also focus our work on extending the graph repository storage with a distributed database, e.g., using the Apache Cassandra project.

Acknowledgements

This work was supported by SAP BusinessObjects Academic Research Center (ARC).

References

1. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems 23, 103–145 (2005)
2. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. 17(6), 734–749 (2005)
3. Alag, S.: Collective Intelligence in Action. Manning Publications (2008)
4. Beer, T., Rasinger, J., Höpken, W., Fuchs, M., Werthner, H.: Exploiting e-c-a rules for defining and processing context-aware push messages. In: Paschke, A., Biletskiy, Y. (eds.) RuleML 2007. LNCS, vol. 4824, pp. 199–206. Springer, Heidelberg (2007)
5. Behrends, E., Fritzen, O., May, W., Schenk, F.: Combining eca rules with process algebras for the semantic web. In: Eiter, T., Franconi, E., Hodgson, R., Stephens, S. (eds.) RuleML, pp. 29–38. IEEE Computer Society, Los Alamitos (2006)
6. Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): Adaptive Web 2007. LNCS, vol. 4321. Springer, Heidelberg (2007)
7. Dey, A.K.: Understanding and using context. Personal and Ubiquitous Computing 5(1), 4–7 (2001)
8. Gu, T., Pung, H.K., Zhang, D.: A service-oriented middleware for building context-aware services. J. Network and Computer Applications 28(1), 1–18 (2005)
9. Heckmann, D.: Distributed user modeling for situated interaction. In: Cremers, A.B., Manthey, R., Martini, P., Steinhage, V. (eds.) GI Jahrestagung (1). LNI, vol. 67, pp. 266–270. GI (2005)
10. Heckmann, D.: Situation modeling and smart context retrieval with semantic web technology and conflict resolution. In: Roth-Berghofer, et al. [19], pp. 34–47
11. Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., von Wilamowitz-Moellendorff, M.: Gumo - the general user model ontology. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 428–432. Springer, Heidelberg (2005)
12. Jamali, M., Ester, M.: TrustWalker: a random walk model for combining trust-based and item-based recommendation. In: Elder IV, J.F., Fogelman-Soulié, F., Flach, P.A., Zaki, M.J. (eds.) KDD, pp. 397–406. ACM, New York (2009)
13. Jrad, Z., Aufaure, M.A., Hadjouni, M.: A contextual user model for web personalization. In: Weske, M., Hacid, M.-S., Godart, C. (eds.) WISE Workshops 2007. LNCS, vol. 4832, pp. 350–361. Springer, Heidelberg (2007)
14. Kirsch-Pinheiro, M., Villanova-Oliver, M., Gensel, J., Martin, H.: Context-aware filtering for collaborative web systems: adapting the awareness information to the user's context. In: Haddad, H., Liebrock, L.M., Omicini, A., Wainwright, R.L. (eds.) SAC, pp. 1668–1673. ACM, New York (2005)
15. Kobsa, A.: Generic user modeling systems. In: Brusilovsky, et al. [6], pp. 136–154
16. Liu, C.-H., Chang, K.-L., Chen, J.J.-Y., Hung, S.C.: Ontology-based context representation and reasoning using owl and swrl. In: CNSR, pp. 215–220. IEEE Computer Society, Los Alamitos (2010)
17. Loizou, A., Dasmahapatra, S.: Recommender systems for the semantic web. In: ECAI 2006 Recommender Systems Workshop (2006), http://eprints.ecs.soton.ac.uk/12584/
18. Micarelli, A., Gasparetti, F., Sciarrone, F., Gauch, S.: Personalized search on the world wide web. In: Brusilovsky, et al. [6], pp. 195–230

19. Roth-Berghofer, T.R., Schulz, S., Leake, D.B. (eds.): MRC 2005. LNCS (LNAI), vol. 3946. Springer, Heidelberg (2006)
20. Thollot, R., Brauer, F., Barczynski, W.M., Aufaure, M.A.: Text-to-query: dynamically building structured analytics to illustrate textual content. In: EDBT 2010: Proceedings of the 2010 BEWEB Workshop, pp. 1–8. ACM, New York (2010)
21. Wan, K., Alagar, V.S., Paquet, J.: An architecture for developing context-aware systems. In: Roth-Berghofer, et al. [19], pp. 48–61 (2005)
22. Wang, X., Zhang, D., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using owl. In: PerCom Workshops, pp. 18–22. IEEE Computer Society, Los Alamitos (2004)

Storage and Use of Provenance Information for Relational Database Queries

Zhifeng Bao[1], Henning Koehler[2], Xiaofang Zhou[2], and Tok Wang Ling[1]

[1] School of Computing, National University of Singapore
{baozhife,lingtw}@comp.nus.edu.sg
[2] University of Queensland
{henning,zxf}@itee.uq.edu.au

Abstract. In database querying, provenance information can help users understand where data comes from and how it is derived. Storing the provenance data is critical in the sense that, the storage cost should be as small as possible and of fine granularity, and it should support the user query on provenance tracking efficiently as well. In this demo, we have implemented a relational database system prototype which can support SQL-like query while supporting provenance data recording during query execution. In particular, we propose a tree structure to store provenance information and further propose various reduction strategies to optimize its storage cost; we support the functionality of provenance data tracking at tuple level for user queries in a visualized way.

1 Introduction

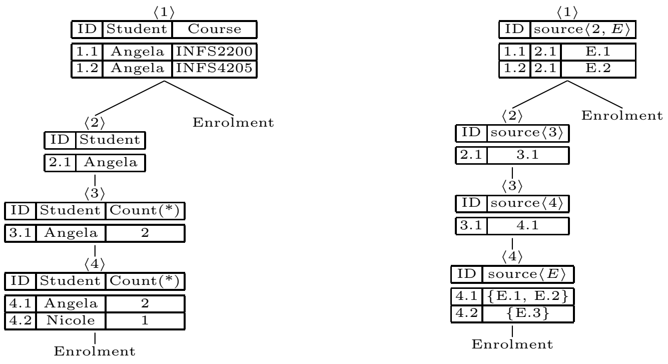
In relational database, data provenance for a user query can describe how an output result is derived from the data in source tables, and tracking of provenance can help update views, explain unexpected results and assist with data integration. The granularity of provenance data is usually a tuple or an attribute value in tuple. The size of provenance data can often easily exceed the size of actual data [1]; although disk space becomes cheaper, data itself grows as well. If we compute provenance only when requested rather than storing it, the cost can be expensive when no good inverse function is found, and it may require intermediate query results to be stored [2].

In this work, we mainly address the issue of efficient provenance storage in the context of relational database queries: the storage cost is as compact as possible, while the computational overhead for provenance tracking and optimization is reasonably small. First, we develop a *provenance tree* data structure for storing provenance information, which closely resembles the query tree. Redundant storage of information about the derivation process is avoided as tuples which possess the same derivation process (with different inputs) are grouped together. Although we address provenance for DB queries here, our solutions are applicable to dealing with arbitrary operator trees. Based on the provenance tree, in order to further reduce the storage cost, we propose several optimization strategies, which select good choices of nodes where provenance information should be stored for a given query tree. By incorporating the above techniques, we implemented a relational database prototype that supports both the evaluation of SQL-like queries and the maintenance of provenance data of user queries.

2 Provenance Tree Storage

When considering provenance for database queries, the granularity of provenance information can be data values, tuples or tables. For most database queries, value-provenance can be derived from tuple-provenance in a straight-forward and efficient manner, while it is time costly to derive tuple-provenance from table-provenance by re-executing the query or using inversion [2]. Therefore, we focus on how to store the tuple-level provenance information.

A common approach for storing tuple-level provenance is to store transformation functions as part of the provenance information. However, this can generate significant storage overhead if multiple instances share the same transformation, which is the case for many scenarios, and in particularly for database queries. We address this problem by using a tree-structure with nested data types for storing provenance information, which matches the query structure.



⟨1⟩

ID	Student	Course
1.1	Angela	INFS2200
1.2	Angela	INFS4205

⟨2⟩

ID	Student
2.1	Angela

⟨3⟩

ID	Student	Count(*)
3.1	Angela	2

⟨4⟩

ID	Student	Count(*)
4.1	Angela	2
4.2	Nicole	1

⟨1⟩

ID	source⟨2, E⟩	
1.1	2.1	E.1
1.2	2.1	E.2

⟨2⟩

ID	source⟨3⟩
2.1	3.1

⟨3⟩

ID	source⟨4⟩
3.1	4.1

⟨4⟩

ID	source⟨E⟩
4.1	{E.1, E.2}
4.2	{E.3}

Fig. 1. Intermediate Query Results and Provenance Tree

Given a query tree, we associate a *provenance table* with the root and some of the intermediate nodes. Each provenance table contains a set of provenance tuples of a fixed provenance type. A *provenance type* is constructed from a base type *tupleID*, whose domain contains references to either a provenance tuple or a tuple in the base table. E.g. the provenance type corresponding to a basic aggregation operator should be a set of tupleIDs. The provenance tables together with a mapping of output data tuples to provenance tuples form a *provenance tree*. Figure 1 shows the intermediate results of a query tree during execution and the corresponding provenance tree, for people to get a rough idea about how the provenance tree looks like, the full example can be found in [3].

After the initial provenance tree is built, we can traverse the tree in a top-down manner to remove the tuples that have never been referenced by any output tuple. We call such provenance tree as *redundancy-free* provenance tree.

There are two basic approaches for storing provenance information: (1) *Store-final* approach, which uses a single provenance table for the final result referencing base tuples directly. (2) *Store-all* approach, which is another extreme in that it stores a provenance table for every intermediate result, i.e. every node in the query tree. Both store-final and store-all can be arbitrarily poor, interested readers can refer to [3] for detailed examples.

Instead of storing provenance information at every node of the query tree, we propose two reduction rules to store it only at some of them, based on the tree resulting from the store-all approach.

Rule I: Whenever all tuples in a provenance table are referenced at most once, we copy the provenance information rather than referencing it.

Rule II: Whenever we would reference a provenance table of type tupleID (i.e. referenced tuples contain only a single reference), we copy the provenance information instead.

While these reduction rules are easy and fast to implement, the resulting reduced provenance tree may not be optimal in term of storage space. Therefore, we further propose a dynamic programming solution which runs in polynomial time to optimally decide which node in the initial provenance tree should be materialized. Please refer to [3] for details about the above reduction strategies.

3 System Architecture

The architecture of our system is shown in Figure 2. It consists of three core parts: (1) Data storage unit, which takes the job of loading/storing the database data and basic indices. (2) query evaluation engine, processing a SQL-like data query on database. (3) Provenance data manager, which further contains two units: the provenance storage unit that builds and stores the provenance tree on the fly at the time of query processing; the provenance query engine which provides user the facility to answer provenance tracking query from user side.

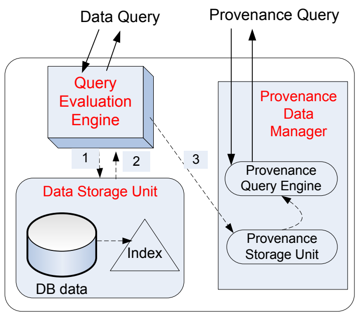
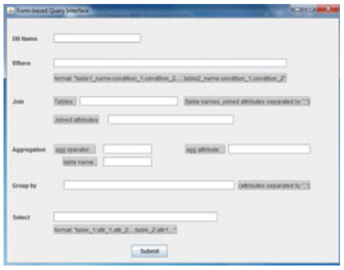


Fig. 2. System Architecture



Form-based Query Interface
DB Name
Where
format "table1_name condition_1,condition_2... table2_name condition_1,condition_2"
Join
Tables
table names, joined attributes separated by ";"
Joined attributes
Aggregation
agg operator
agg attribute
table name
Group by
attributes separated by ";"
Select
format "table_1 attr_1,attr_2... table_2 attr1..."
Submit

Fig. 3. Form-based Query Interface

4 System Features

Our system is a database system prototype supporting provenance data storage and tracking besides the traditional data storing and querying. We describe the features of our system by going through an empirical flow about how it works.

First, we have a data loader to parse and load raw data into database, based on the database schema given. The system will provide the ER diagram to users for them to learn the schema before issuing a query.

Second, we have implemented a query evaluation engine, which accepts SQL-like-syntaxed queries, where most operators such as join, aggregation, group by, selection, projection are efficiently supported. The query format follows the flow of evaluating a query tree in a bottom-up fashion. Since learning SQL (or similar) query syntax have been a tough job for novice user, we additionally provide user a *form-based* query interface, where user can express their query intention by specifying the database, table, attributes and associated operators on these attributes (see Figure 3). Then the query translator will translate the form-based query into our SQL-like syntax followed by query evaluation and result return, as shown in Figure 4.

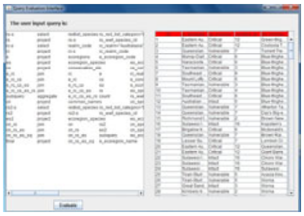


Fig. 4. Query Interface

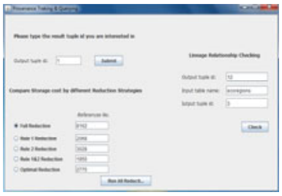


Fig. 5. Provenance Storage & Tracking

Fig. 6. Tracking Results

Third, at the end of query evaluation, the *redundancy-free* provenance tree is built, and the system offers user different options for tree reduction, such as store-final, reduction by rule 1, rule 2, and optimal reduction, as shown in Figure 5. The user can try any of them to see the effects of different reduction strategies by comparing the size of resulted provenance tree.

Fourth, our system achieves a novel visualization of the *provenance usability* facility. We provide user an efficient processing on two common types of provenance tracking queries (in Figure 5): (i) For any output tuple of the query that user is interested in, the system will return an instance of the provenance tree in a visualized way, showing the user how this output tuple is derived from a particular (set of) data tuples in database (through which kind of query operators) in the order of query tree evaluation. (ii) User can specify several output tuples and source data tuples, and would like to see whether there is a lineage relationship between them. For the above two types of queries, the provenance tree instance returned only keeps track of the tuple ids of intermediate tables, while the user may also want to view the intermediate results in each step. Therefore, we also maintain a tree instance that keeps the intermediate results during query evaluation. As a result, both the provenance tree instance and data tree instance w.r.t the provenance query is returned, as shown in Figure 6.

References

1. Chapman, A., Jagadish, H.V., Ramanan, P.: Efficient provenance storage. In: SIGMOD Conference, pp. 993–1006 (2008)
2. Cui, Y., Widom, J.: Practical lineage tracing in data warehouses. In: ICDE, pp. 367–378 (2000)
3. Koehler, H., Bao, Z., Zhou, X., Sadiq, S.: Provenance trees: Optimizing relational provenance storage. submitted to SIGMOD (2011), http://www.comp.nus.edu.sg/~baozhife/provenancetree/provenance_longversion.pdf

MRQSim: A Moving Range Query Simulation Platform in Spatial Networks*

Yu Gu, Na Guo, Chuanwen Li, and Ge Yu

Northeastern University, China
{guyu,guona,lichuanwen,yuge}@ise.neu.edu.cn

Abstract. In this paper, we present our system MRQSim to efficiently conduct the continuous range query in the simulated road network environment when the query object moves. We propose two kinds of query types and consider complex simulation setups. The relative system architecture and demonstration are illustrated. Because incremental maintenance methods based on valid intervals are adopted and the potential uncertainty of objects is considered, MRQSim can obtain high efficiency with good adaptability to complex running scenarios.

1 Introduction

In recent years, location-based services (LBS) are extensively offered in a wide range of applications including intelligent transportation systems, self-help travel services and digital battlefield. Due to the large size of spatio-temporal data, how to process the queries efficiently has been a hot research topic in the field of LBS and spatio-temporal data management[1]. Especially, as the directions and trajectories of mobile objects are usually restricted by an underlying spatial network in practice, such as railway, highway or waterway route network environment, many researches turn to focus on the query in the spatial network [2]. Compared to the query execution in the snapshot manner offered by most car navigation systems, the continuous range query is also quite useful in practical applications. Furthermore, unlike the kNN query [3], how to efficiently process the continuous range query in spatial networks has not been sufficiently studied. Therefore, our proposed system MRQSim is targeted at solving this problem.

In detail, the moving range query in spatial networks is defined as: when the query object q moves along the pre-defined paths, given a set of spatial objects O and the query radius r, find all the objects with the distance from q no more than r periodically. According to different applications, we define two types of range queries in spatial networks namely NDMR (Network Distance Moving Range) and EDMR (Euclidean Distance Moving Range). Although the movement of the query object must be constrained by the road network, NDMR utilizes the road network distance as the metric while EDMR utilizes the Euclidean distance to determine the range of queries. In the detailed implementation, the incremental

* This research was partially supported by the National Natural Science Foundation of China under Grant No.61003058, No.60773220 and No.60933001.

maintenance strategy based on split nodes is adopted. Furthermore, due to the privacy protection or the limited positioning precision, there may be widespread location information uncertainty. We extend the algorithms in MRQSim to adapt to the uncertainty. In the next sections, we will further illustrate the system architecture and interface demonstration of MRQSim.

2 System Architecture

The system MRQSim is composed of three major components.

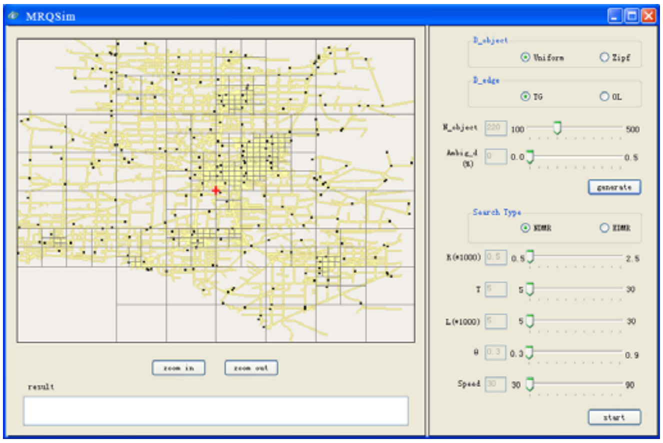
Simulation Engine. The simulation engine is responsible for the road network simulation, the target object simulation and the query object movement simulation. The detailed parameters can be flexibly set up by users to reflect different scenarios. Specifically, we offer two real road network data sets downloaded from the website http://www.maproom.psu.edu/dcw to ensure the effectiveness of our algorithms, which are separately collected from Oldenburg City in Germany(OL)(including 7304 edges and 6104 nodes) and San Joaquin City in USA(TG)(including 23873 edges and 18262 nodes). The engine can be easily replaced by the data collection module in the real-life applications.

Processing Engine. The processing engine is the core system module which executes the major query optimization algorithms. In the detailed implementation for accurate positions, we propose the concept of split nodes and classify the split nodes into four categories according to the judgements of the extreme point, the non-extreme point, the vertex point and the edge point. When the query objects enters a new indexing partition region, the split nodes will be computed on line for those target objects bounded in the query range. The efficient indexing, merging and compressing techniques are further adopted. Split nodes will split road segments into valid intervals in which the query result remains unchanged. Consequently, the new result can be obtained by incremental maintenance according to the information recorded for relative split nodes. Partial detailed techniques can be referred in [4]. Furthermore, considering the scenario when the position uncertainty exists, the position is represented by a segment instead of a point. Correspondingly, the framework of split nodes will be extended to split intervals. We modify the algorithms to efficiently execute the probabilistic range query. Because different split node or split interval computation methods are utilized by NDMR and EDMR, the processing engine needs to choose from four execution strategies namely NDMR(accurate), NDMR(uncertain), EDMR(accurate) and EDMR(uncertain) according to different query requests.

User Interface. We offer flexible and friendly user interface including the input control panel, the scenario display panel and the result output panel. The input control panel offers the function to set up the input parameters of the simulation engine. The runtime map information will be timely updated in the scenario display panel with the split nodes or split intervals indicated. The result objects will be continuously output during each query period.

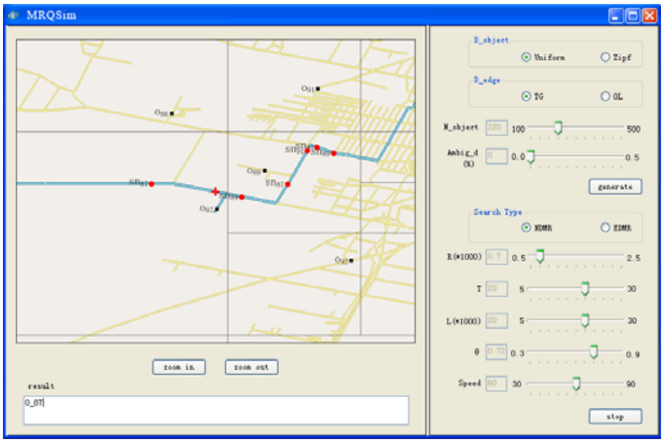
3 System Demonstration

In the demonstration, users need to first set up the input control panel. After specifying the road network data set, the distribution and number of target objects and the ambiguity degree(accurate when it is zero), a map will be generated which is illustrated in Figure 1. Furthermore, the information such as the query type, the query range and the moving object speed can be flexibly offered by the users.



D_object: [✓ Uniform] Zipf
D_edge: [✓ TG] OL
N_object 220 100 500
Ambig_d (%) 0 0.0 0.5
Search Type: [✓ NDMR] EDMR
R(*1000) 0.5 0.5 2.5
T 5 5 30
L(*1000) 5 5 30
θ 0.3 0.3 0.9
Speed 30 30 90

Fig. 1. Environment simulation interface display

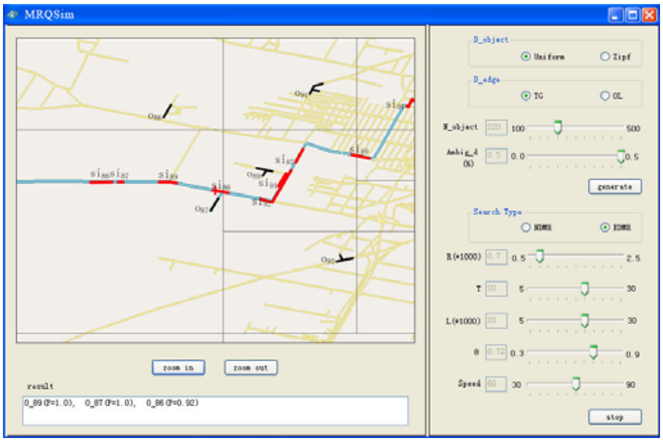


result: 0_87

D_object: [✓ Uniform] Zipf
D_edge: [✓ TG] OL
N_object 220 100 500
Ambig_d (%) 0 0.0 0.5
Search Type: [✓ NDMR] EDMR
R(*1000) 0.7 0.5 2.5
T 20 5 30
L(*1000) 20 5 30
θ 0.72 0.3 0.9
Speed 60 30 90

Fig. 2. NDMR(accurate) query interface display

After pressing the start button, according to different ambiguity levels and query types, corresponding strategies in the processing engine will be executed. Also, the split nodes and split intervals are displayed and updated in the display panel. The route where the objects have passed will be indicated by the blue



D_object: [✓ Uniform] Zipf
D_edge: [✓ TG] OL
N_object 220 100 500
Ambig_d (%) 0.5 0.0 0.5
Search Type: NDMR [✓ EDMR]
R(*1000) 0.7 0.5 2.5
T 20 5 30
L(*1000) 20 5 30
θ 0.72 0.3 0.9
Speed 60 30 90

result
O_89(P=1.0), O_87(P=1.0), O_86(P=0.92)

Fig. 3. EDMR(uncertain) query interface display

color and the result will be returned in the timely fashion. As the examples, Figure 2 and Figure 3 illustrate the runtime layout of NDMR(accurate) and EDMR(uncertain) separately.

4 Conclusion and Future Work

MRQSim is designed for efficient moving range query in the tunable simulated road networks. We offer the flexible control function and visualized scenario interface. In the detailed implementation, split nodes and split intervals are separately computed for the accurate and uncertain position situation, and thus incremental maintenance can be conducted. In the future work, given the map and real-time moving object information, our system is expected to be extended to real-life monitoring scenarios with high efficiency and robustness.

References

1. Wu, K.L., Chen, S.K., Yu, P.S.: Incremental processing of continual range queries over moving objects. IEEE Transactions on Knowledge and Data Engineering 18(11), 1560–1575 (2006)
2. Kolahdouzan, M.R., Shahabi, C.: Voronoi-based k nearest neighbor search for spatial network databases. In: Proceedings of VLDB, pp. 840–851 (2004)
3. Nutanong, S., Zhang, R., Tanin, E., Kulik, L.: The v*-diagram: a query-dependent approach to moving knn queries. PVLDB 1(1), 1095–1106 (2008)
4. Guo, N., Gu, Y., Wang, Y.Q., Yu, G.: Valid interval: an effective technique for continuous range query in spatial networks. In: Proceedings of International Conference on Emerging Databases, pp. 222–227 (2010)

DWOBS: Data Warehouse Design from Ontology-Based Sources

Selma Khouri[1] and Ladjel Bellatreche[2]

[1] National High School of Computer Science, Algiers, Algeria
s_khouri@esi.dz
[2] LISI/ENSMA Poitiers University, Futuroscope, France
bellatreche@ensma.fr

Abstract. In the past decades, data warehouse (DW) applications were built from traditional data sources. The availability of domain ontologies creates the opportunity for sources to explicit their semantics and to exploit them in many applications, including in data warehousing. In this paper, we present DWOBS, a case tool for ontological-based DW design based on domain ontologies. It takes as inputs (i) a set of sources referencing OWL ontology and (ii) a set of decisional requirements formulated using Sparql syntax on that ontology. DWOBS gives a semantic multidimensional model of the DW to be designed.

1 Introduction

Ontologies have been advocated by software engineering community, where they are used to solve several problems during different stages of software engineering lifecycle: requirement engineering, modeling, model transformations, maintenance, and software comprehension [3]. In the field of data warehousing (DW) development, a design methodology is not really useful unless a case tool supporting it is provided. Two main components are required to design a DW: *data sources* and *users requirements*. Ontologies contribute largely to explicit the sense of the concepts used by these components. Several research efforts have recognized ontologies as the key to ensure an automatic integration of heterogeneous data sources [1]. Their presence generates large amount of ontological instances usually stored in ontology-based databases (OBDB). Several architectures supporting OBDB were proposed by academicians (OntoDB, Jena, Sesame, etc.) [4] and commercial editors (Oracle, IBM SOR, etc.). Different other studies propose the use of ontologies to assist designers to identify users' requirements, to unify the requirements by detecting their inconsistencies, and offer a formal specification of the requirements model. The spectacular development of domain ontologies and their similarity with conceptual models of OBDBs motivate us to propose an ontology-based tool, named DWOBS, to design DWs. The main objective of this tool is to assist designers to exploit these ontologies in building their multidimensional schemes. DWOBS gives designers similar functionalities offered by classical database design tools such as: *PowerAMC*, *Rational Rose*, etc. The use of ontologies in the development of DWs represents a challenging issue. Note that they contributed in various phases of DW lifecycle: ETL, semantic optimization, etc., but none design tool exists. DWOBS is, to our knowledge, the first graphical tool for multidimensional DW design dedicated for OBDB. It is based on a comprehensive methodology presented in [5]. One of its main characteristics is the consideration of both sources and designer requirements defined at the ontological level. It supposes the existence of a set of OBDBs that reference an integrated 'global' ontology (GO) described in OWL and a set of decisional requirements (Fig 1). It gives DW designers the ability to express their decisional requirements on the domain ontology by means of *Sparql* language. The projection of the decisional requirements on GO generates a local ontology (LO) that can be seen as a view of the GO. The generated LO represents the conceptual model of the DW. It offers: (i) a formal and expressive representation (based on description logics) and (ii) reasoning capabilities. Having a LO has several advantages: (i) it specializes the GO, since this former is usually large (like the *Cyc* ontology used in our scenario) which makes its exploitation more difficult, (ii) it facilitates the reasoning process (existing reasoners are efficient only on small ontologies) and (iii) it is well adapted to satisfy visualization constraints which is an important aspect for DW designers.

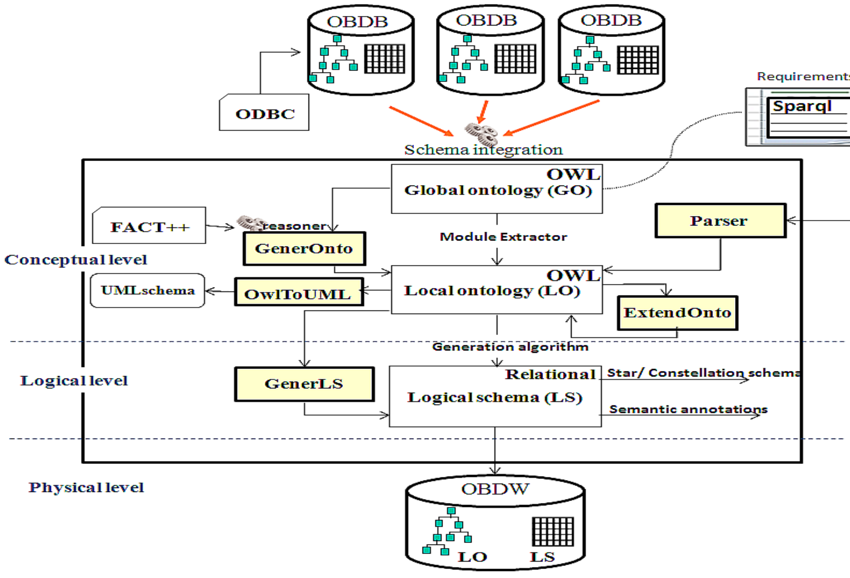


Fig. 1. DWOBS architecture

2 System Description

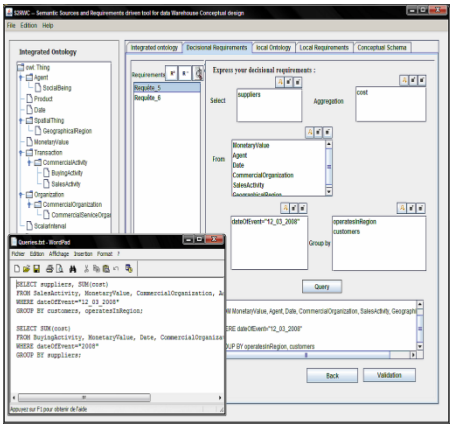
DWOBS is a graphical tool developed in Java. The OBDBs that participate in the construction of the DW reference an existing integrated ontology formalized in OWL. Note that DWOBS is not an ETL tool; it simply loads the OBDBs via ODBC and displays their contents to the user. The GO integrating the selected OBDBs is identified by its URI, and its information is displayed on the same way as Protégé editor. Classes are presented as a hierarchy. The selection of a class displays its information (Concept name, description, properties, super-classes and instances). Access to all ontologies is made through the OWL API. Requirements are expressed using the Sparql language syntax. Designers can either express the requirements through an interface allowing them to select the different classes and properties used in the requirements or by choosing a text file containing the requirements (Fig 2). The tool checks the syntax of each requirement in the case. The signature of each requirement is given as follows:

SELECT ontological_properties, Aggregation_Fonction (ontological_properties)
FROM ontological classes
WHERE Selection condition (ontological_properties)
GROUP BY ontological properties (that can be grouped)

A parser analyzes these requirements in two phases (Fig 1): (i) in the first one, it projects the decisional requirements on the integrated ontology to generate a LO. This LO contains all the terms used in the requirements and is generated by *GenerOnto* module as an OWL file using the OWL module extractor (plugin available in Protégé editor) ensuring the logical completeness of the extracted ontology. Some statistics of this ontology are presented (number of entities extracted, logical axioms and ontology URI). *GenerOnto* accesses this LO and displays it to the user in the same way as the GO. *ExtendOnto* module offers different interfaces to extend the LO (by importing, adding, specializing or renaming classes and properties) so that it covers all the application objectives. The designer may also express new *sparql* requirements on the extended LO. Fact++ reasoner is invoked to classify the LO classes taxonomy and to check its consistency. *OwlToUml* module converts the LO to an UML conceptual model by translating owl constructors to UML constructors using a set of rules defined based on [2], and displays it as an UML class diagram. The goal of this translation is to present to users a conceptual model easy to read and understand so that they can validate it. (ii) The second parsing phase analyzes the decisional requirements expressed on the GO and LO to identify the multidimensional role of each class and property according to their occurrence in the Sparql clauses (for example properties in the Aggregation and the Group by clause are candidates to represent respectively measures and dimension attributes…). *GenerLS* module accesses the LO and annotates the identified concepts by their multidimensional role (fact, dimension, measure, attribute or hierarchy). It also generates the *logical star or constellation schema* corresponding to the multidimensional concepts identified (Fig 3). Dimensions hierarchies classes are connected to their facts through *(1:n) relationships* (detected by the presence of *functional ObjectProperties*). Is-a relationships are used to deduce new facts and generalized dimensions. This multidimensional model is defined semantically i.e., a formal link between the LO and the DW model is saved. The multidimensional model is presented in two ways (Fig 3): (i) as a tree whose nodes are the identified facts, their measures, their respective dimensions and the dimensions attributes, (ii) graphically as a relational star schema that can be printed. The designer can modify this model graphically by adding and deleting classes and properties. He is aided by a set of recommendations obtained from the ontology constraints (used to specify cardinalities and to validate the obtained schema by adding classes or deleting redundant classes). The DW logical schema and the LO schema can be used after that for the construction of an Ontology-Based Data Warehouse (OBDW).

3 Demonstration

We demonstrate DWOBS using two OBDBs referenced by Cyc ontology (www.cyc.com/opencyc). These OBDBs are stored in Postgres DBMS. We adapt a fragment of this ontology to represent a domain of commercial transactions. The decisional requirements used analyze transactions costs according to regions and customers and analyze purchase costs by suppliers. DWOBS offers designers the possibility to explore different classes of the ontology, and express their requirement on local ontology (LO) (Fig. 2). By the means of useful interfaces, they may extend the LO, by specializing some concepts. New requirements using the specialized concepts are also expressed. Two fact tables and seven dimensions (with their measures and attributes) corresponding to this scenario are visually shown (Fig. 3). Multidimensional stereotypes are used to facilitate the *readability* of the schema. Once satisfied, the designer has the possibility to save the generated schema and to enrich it if necessary. Due to the lack of space, a video of our demonstration is given at this url: http://www.lisi.ensma.fr/members/bellatreche/DASFAADEMO.



```
SELECT suppliers, SUM(cost)
FROM SalesActivity, MonetaryValue, CommercialOrganization, A
WHERE dateOfEvent="12_03_2008"
GROUP BY customers, operatesInRegion;

SELECT SUM(cost)
FROM BuyingActivity, MonetaryValue, Date, CommercialOrganiza
WHERE dateOfEvent="2008"
GROUP BY suppliers;
```

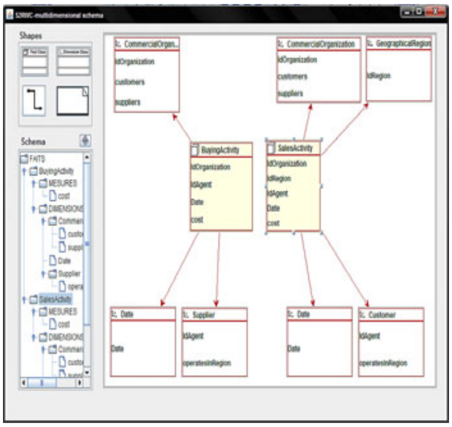


Fig. 2. Expressing Sparql query on the ontology **Fig. 3.** Generated multidimensional model

References

1. Bellatreche, et al.: Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. Computers in Industry 57(8-9) (2006)
2. Calvanese, D., Lenzerini, M., Nardi, D.: Description logics for conceptual data modeling. Logics for Databases and Information Systems, 229–264 (1998)
3. Gašević, D., Kaviani, N., Milanović, M.: Ontologies and Software Engineering. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, 2nd edn. Springer, Heidelberg (2009)
4. Jean, S., et al.: OntoDB: It is time to embed your domain ontology in your database. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 1119–1122. Springer, Heidelberg (2007)
5. Khouri, S., Bellatreche, L.: A methodology and tool for conceptual designing a data warehouse from ontology-based sources. In: DOLAP 2010, pp. 19–24 (2010)

AUCWeb: A Prototype for Analyzing User-Created Web Data

Weining Qian, Feng Chen, Juan Du, Weiming Zhang, Can Zhang, Haixin Ma, Peng Cai, Minqi Zhou, and Aoying Zhou

Institute of Massive Computing, East China Normal University, Shanghai, P.R. China
{wnqian,mqzhou,ayzhou}@sei.ecnu.edu.cn

Abstract. In this demonstration, we present a prototype system, called AUCWeb, that is designed for analyzing user-created web data. It has novel features in that 1) it may utilize external resources for semantic annotation on low-quality user-created content, and 2) it provides a descriptive language for definition of analytical tasks. Both internal mechanism and the usage of AUCWeb for building advanced applications are to be shown in the demonstration.

1 Introduction

User-created content is an important kind of resource on the Web. Along with the growth of Web 2.0 applications, more and more information are generated and spreaded via the user-created content. Meanwhile, traditional web applications, such as online forums or news groups also contain a large number of pieces of information that are created by users.

Thus, a natural question to ask is that *how to utilize the information in user-created content for advanced applications effectively and efficiently?* However, a little more study on this problem will show that this is not an easy task. There are several challenges.

- First, user-created content are often informal in terms of representation. Some pieces of information are short or just phrases instead of sentences. Thus traditional natural language processing based methods may fail.
- Second, user-created content are more than web pages. For example, a web page may contain posts from several users, while posts from one user may be one several web pages. The content, along with the relationships between pieces of information should be considered together in analytical tasks.
- Last but not the least, there may exist variant data analysis tasks on the same data set, which may share some common analytical operators. How to provide a easy-to-use yet effective tool that may reuse those operators and existing analytical results is a big challenge.

In this demonstration, we present AUCWeb, a prototype system for user-created web data analysis. AUCWeb has two novel features. First, it takes a best-effort approach to data analysis. The raw data along with the analytical

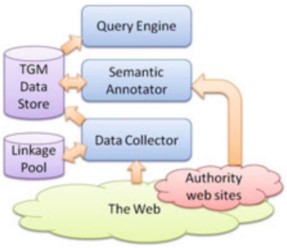


Fig. 1. AUCWeb architecture

Fig. 2. A snapshot of Shanghai Expo 2010 microblog data analysis

results, including semantic annotations, discovered external links, and statistics on term frequency, and etc. are all stored and managed. Further analysis can be conducted both on raw data and partial results.

AUCWeb is not a closed system. It utilizes external web resources, e.g. news, wikipedia, and online statistics, for semantic annotation, since user-created content are often of low quality and not self-contained.

The second feature of AUCWeb is that it provides a SQL-like descriptive language for defining analytical tasks. The underlying data model of AUCWeb is a graph with tags. It can be used to represent both content and relationships between pieces of information. It can also be used to represent partial analytical results. Analyzing tasks in descriptive language are translated to operators on the graph, and executed by the query processing engine.

The demonstration contains two parts. First, the internal of the system, including semantic annotation and analytical query translation are to be demonstrated. In the second part, two applications based on AUCWeb are to be presented to show how AUCWeb can be used for user-created content analysis.

2 System Overview

The architecture of AUCWeb is shown in Figure 1. It contains three modules: *Data Collector*, *Semantic Annotator*, and *Query Engine*. Data Collector crawls web pages containing user-created content, parses their structures and identifies *pagelets*, which are pieces of information each of which is created by one user at one moment. Furthermore, Data Collector identifies the relationships, such as *reply-to*, or *comment*, or *following*, etc. between pagelets.

Semantic Annotator identifies *semantic entities* in user-created content. There are several differences between this module and a traditional information extraction module. First, not all types of semantic entities are predefined. Thus events or emerging hotspots that are previous unknown can be identified. Second, each semantic entity is tagged with a set of attributes and keywords, so that its relationships to other entities are defined. Last but not the least, external resources are used in annotation. Since it is expensive to access external resources in terms of latency, this module only visits them when semantics cannot be determined.

The Query Engine accepts user queries, transforms them into internal query plans, and execute them via operators. Design and implementation details are introduced in [2]. We give a brief introduction to the implementation here.

2.1 Data Collecting and Preparation

Data Collector is actually a customized crawler. It is different to traditional web crawlers in that it crawls web pages *while* analyzing them. Thus, just after a web page is crawled, pagelets are extracted based on the structure of the page, and stored in XML forms, in which author, timestamp, title, content, and relationships to other pagelets are tagged. Meanwhile, the linkage pool is updated. The techniques for pagelet extraction are introduced in [3]. Since this is not the focus of our demonstration, we omit the details here.

2.2 Semantic Annotation with External Resources

Semantic Annotator is in responsible for identifying *semantic entities*. A semantic entity is *a term with fixed context in different appearances*. *Context* means people, location, time, and terms that are highly related to a specific term.

Semantic Annotator first analyzes the linguistic and statistical features of terms to identify semantic entities candidates [1]. Then, it generates a template of the form: <*term, time, location, people, related-term(s)*> for each candidate.

Semantic Annotator tries to fill in the template by using information within the pagelet. For those slots that cannot be filled in, Semantic Annotator automatically generates a set of queries and send them to external resources, such as online news sites, Wikipedia, and online statistics sites (e.g. Google Trends). The results returned from external resources are used to fill in the template.

2.3 Data Analysis with Descriptive Language

AUCWeb uses a data model called Tagged Graph Model (TGM). A TGM graph is a binary (V, E) in which V is a set of vertices and E is a set of edges. Each vertex $v \in V$ is of the form $(id, \{attr_i\})$. Here, id is the identifier of the vertex, and $attr_i$'s are attributes. Each $attr_i$ is a triple $(name, type, value)$. Each edge $e \in E$ is of the form $(id_s, id_d, \{attr_i\})$, where id_s and id_d are source and destination, and $attr_i$'s are attributes as those of vertices.

A SQL-like query language is defined on TGM. AUCWeb supports five types of operators over the graph that implementing the query language. They are *wrappers*, *filters*, *merges*, *groupings*, and *Aggregations*. Queries are translated to a tree-structured query plans on operators. AUCWeb uses a column-bases storage with indexing for management of TGM data.

3 Demonstration Outline

Our demonstration includes the following two parts:

AUCWeb **internals:** We will show how user-created data are collected, organized, and managed in the system. There are three focuses on this part of demonstration:

- We will show how the pagelets are analyzed and managed in the system, so that pieces of information and their relationships can be handled.
- We will show the process of external-resource-based semantic annotation. The external resources that are used and the templates that are filled by data from those resources are to be shown.
- We will illustrate the process of analytical query processing. Queries in descriptive language, their corresponding query plans on operators over the graph, and query results are to be shown. Users are also encouraged to pose their own queries to experience how AUCWeb works.

Applications on AUCWeb**:** In this part, we will show two applications built on AUCWeb.

- The first application is an online forum trends analysis toolkit. User may analyze trends of statistics on terms given query conditions on locality, time, topic, people, and etc. This application will show how semantic annotations can be used in advanced analytical tasks.
- The second application is for Expo 2010 micro-blog data visualization. A snapshot is shown in Figure 2. We will show how various visualization requirements are represented in our descriptive language. We will also show how partial analytical results can be utilized to facilitate visual data analysis in terms of effectiveness and efficiency.

The demonstration will be shown over real-life data, which are about hundreds of gigabytes, on one server. We will crawl some and part of web sites, and use these data as backups for offline scenario.

Acknowledgment

This work is partially supported by National Science Foundation of China under grant number 60833003 and 61070051, and National Basic Research (973 program) under grant number 2010CB731402.

References

1. Cai, P., Luo, H., Zhou, A.: Semantic entity detection by integrating crf and svm. In: Chen, L., Tang, C., Yang, J., Gao, Y. (eds.) WAIM 2010. LNCS, vol. 6184, pp. 483–494. Springer, Heidelberg (2010)
2. Qian, W., Zhou, M., Zhou, A.: A best-effort approach to an infrastructure for chinese web related research. In: Proceedings of the 1st International Workshop on Intelligence Science and Intelligent Data Engineering, IScIDE 2010 (2010)
3. Zhang, C., Zhang, J.: Inforce: Forum data crawling with information extraction. In: Proceedings of the 4th International Universal Communication Symposium, IUCS 2010 (2010)

Blending OLAP Processing with Real-Time Data Streams

João Costa[1], José Cecílio[2], Pedro Martins[2], and Pedro Furtado[2]

[1] Polytechnic of Coimbra
jcosta@isec.pt
[2] University of Coimbra, Coimbra, Portugal
{jcecilio,pmon,pnf}@dei.uc.pt

Abstract. CEP and Databases share some characteristics but traditionally are treated as two separate worlds, one oriented towards real-time event processing and the later oriented towards long-term data management. However many real-time data and business intelligence analysis do need to confront streaming data with long-term stored one. For instance, how do current power consumption and power grid distribution compare with last year's for the same day?

StreamNetFlux is a novel system that recently emerged to the market designed to integrate CEP and database functionalities. This blending allows the processing of queries that need such capabilities with top efficiency and scalability features.

1 Introduction

Lately, CEP engines have been gaining ground in commercial enterprises with event processing needs, including the IBM System S[1] stream processing core that provides scalable distributed runtime execution, a SPADE language and compiler[2] for stream processing applications and both optimization and fault-tolerance capabilities built-in; Tibco BusinessEvents[3] uses a UML-based state model, a rules engine based on the industry-standard RETE protocol, and events capture and processing functionality. Truviso[4] offers data analysis using standard SQL, and results can trigger actions such as alerts to decision makers or events in other systems, or be delivered to an end-user over a standard web browser. Queries are continuous and the system can be run in a distributed fashion across many applications. Coral8 and Aleri [5] are an engine and platform, respectively, for streaming event data, with an SQL extension to process data. The Coral8 Engine uses a Continuous Computation Language (CCL), and the Aleri Streaming Platform has visual dataflow authoring, event driven integration and implements scalability support. The StreamBase [6] server is programmed with either a StreamSQL processing language or a visual query language, and the engine offers a large number of operators to use on streams that include merging, filtering, statistics computation, thresholding and pattern matching. In Oracle CEP [7], applications are designed based on stream sources, processors represented in XML with CQL-like queries, stream sink beans that process output data and channels linking parts together. Esper [8] is a Java-based CEP that allows users to re-use Java capabilities and add powerful event and pattern handling in an event processing language syntax (EPL).

Typically, CEP and databases are two distinct entities. This approach increases complexity and creates performance problems in many practical applications when data analyses require database data together with CEP streams.

2 Streamnetflux System

StreamNetFlux CEP-DB system, illustrated in figure 1, integrates both functionalities into a single, scalable and efficient engine. StreamNetFlux users pose queries and StreamNetFlux manages memory, machines, databases and CEP engines with top efficiency and scalability.

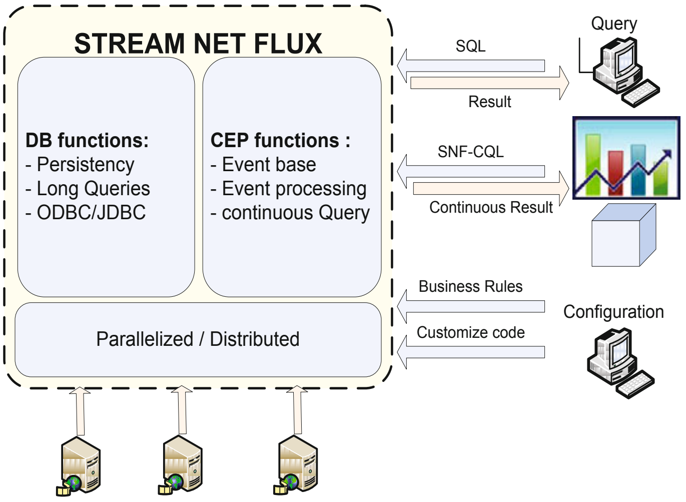


Fig. 1. StreamNetFlux Architecture

StreamNetFlux implements common features of database systems, including persistence, DB storage mechanisms, transaction management, recovery and also an ODBC/JDBC interface. The persistent storage uses a hybrid memory-disk organization and techniques to offer top efficiency and scalability. Complex event and stream processing allows data analysis to be always up-to-date.

Users can pose StreamNetFlux statements and application code with embedded StreamNetFlux statements. These include continuous StreamNetFlux queries and customized data analysis code. By offering DB functionalities and user transparent capabilities, StreamNetFlux allows applications to use the system without disrupting regular application functionalities.

StreamNetFlux ease of use, real-time processing, scalability and efficiency was evaluated with a massive volume of high-rate data produced by an energy power grid infrastructure. Besides power grid data, it also processes data from energy producers, from major energy enterprises and thousands of Micro-generation-producers (e.g. producing from solar power or window turbine).

The distribution of electricity requires that energy be produced as required by the consumers, since it cannot be efficiently stored for later usage, except for a limited and restricted time period. When required, some inactive power plants need to be activated to produce additional energy to assure the energy consumption. To reduce the energy lost, while electricity transverses the power lines and substations, this additional energy should be produced from power plants nearby the consumers.

To prevent power blackouts, it's crucial to have continuous monitorization of the power grid infrastructure and the evaluation of the energy consumption, energy generation and interconnecting links and power sub-stations capacity.

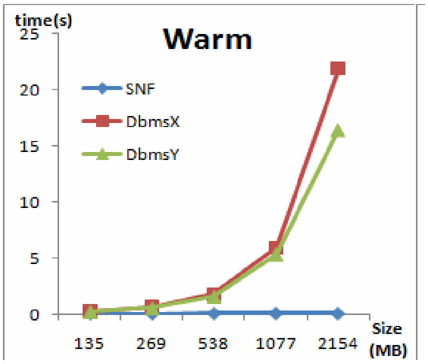


Fig. 2. Performance Summary

For operational purposes, it is important to monitor the usage of the transmission lines, in order to assess when and where abnormal patterns occur and to take preventive or corrective actions (e.g. add additional capacity). Marketing decisions can also be made based on the available data. Alerts and actions may be triggered when there is excess or shortage of some indicator. The scope of alerts, reports and analysis may be drilled up, down, or across different perspectives. Sub-stations and energy power plants capacity can also be monitored for detecting abnormal pattern usage behavior.

StreamNetFlux, even under high-loads, delivers results in the ms range (figure 2), whereas evaluated DBMS engines take too much time to obtain the same results, returning them in a discrete manner and with a highly inefficient way to integrate new data. Such result discretization (time lag between query re-execution against the data, including the new recent one) is unacceptable for critical, high-demanding applications, like energy power grid applications or telecommunications.

Stream engines, for real data analysis, can deliver fast performance results, but only for a small subset of the recent data, limited by the window size. They are unsuited for performing broader analysis, which requires not only recent data contained within the window size limits, but also other data that relies outside the window scope. Having a tool such as StreamNetFlux that allows users to use both streaming and past data is a very important improvement.

StreamNetFlux was designed with easy-of-use considerations, with reduced time-to-learn curve and a fast time to market. It allows users to specify operators and data processors, to define computations, filters, data aggregations and dataflows between them. It also allows the definition of event based actions, or rules, triggering conditions and actions to be performed.

3 Demonstration Roadmap

In the demonstration, we will be using a power grid data schema, with information from the power grid infrastructure, from energy generators (including major energy producers and micro-producers) and also information collected from power meters at consumers, to evidence the main features of StreamNetFlux: ease of use, real-time processing, scalability and efficiency.

The demonstration consists of the following steps:

- **Setting up:** using a power grid data schema, we show how to setup through a set of simple drag and drop and dataflows steps. We also show how it can be setup through a command line console.
- **Running:** we demonstrate how queries are seamlessly posed against Stream NetFlux and how the engine executes business rules. We will demonstrate alerts, reporting and analysis queries, some of those correlating streaming data with stored persistent data. For instance, to: compute the national grid or the sub-station usage and detect variations (e.g. > than) in comparison with the same week day of previous years; trigger Alerts when the variation is greater than a given threshold; raise Alarms when link usage falls below a certain level.
- **Visualization:** show how to explore and visualize data.

The StreamNetFlux is a disruptive product that works with streaming data and stored data, while simultaneously providing scalability and unusual ease of programming and querying. In this demo, we show how the system is able to do that. The technology behind StreamNetFlux has already spurred patent requests and a spin-off company dedicated to the developing of applications for industrial markets such as telecommunications and energy efficiency.

References

[1] IBM Research Exploratory Stream Processing Systems, `http://domino.research.ibm.com/comm/research_projects.nsf/pages/esps.index.html`
[2] Gedik, B., Andrade, H., Wu, K., Yu, P.S., Doo, M.: SPADE: the system s declarative stream processing engine. In: SIGMOD, pp. 1123–1134. ACM, Vancouver (2008)
[3] TIBCO Business Events, `http://www.tibco.com/software/`
[4] Truviso Data Analysis, `http://truviso.com/products/`
[5] Aleri & Coral8, `http://www.aleri.com/products/aleri-cep`
[6] Streambase, `http://www.streambase.com/`
[7] Complex Event Processing, `http://www.oracle.com/technologies/soa/complex-event-processing.html`
[8] Esper Complex Event Processing, `http://esper.codehaus.org`

AutoBayesian: Developing Bayesian Networks Based on Text Mining

Sandeep Raghuram[1], Yuni Xia[1], Jiaqi Ge[1], Mathew Palakal[1], Josette Jones[1], Dave Pecenka[2], Eric Tinsley[2], Jean Bandos[2], and Jerry Geesaman[2]

[1] Indiana University - Purdue University Indianapolis, USA
[2] My Health Care Manager, Inc.

Abstract. Bayesian network is a widely used tool for data analysis, modeling and decision support in various domains. There is a growing need for techniques and tools which can automatically construct Bayesian networks from massive text or literature data. In practice, Bayesian networks also need be updated when new data is observed, and literature mining is a very important source of new data after the initial network is constructed. Information closely related to Bayesian network usually includes the causal associations, statistics information and experimental results. However, these associations and numerical results cannot be directly integrated with the Bayesian network. The source of the literature and the perceived quality of research needs to be factored into the process of integration. In this demo, we will present a general methodology and toolkit called AutoBayesian that we developed to automatically build and update a Bayesian network based on the casual relationships derived from text mining.

1 Introduction

A Bayesian network (BN) is a directed acyclic graph whose arcs denote a direct causal influence between parent nodes (causes) and children nodes (effects). A BN is often used in conjunction with statistical techniques as a powerful data analysis and modeling tool. While it can handle incomplete data and uncertainty in a domain, it can also combine prior knowledge with new data or evidence [1].

There are two approaches to construct a BN: knowledge-driven and data-driven. The knowledge-driven approach involves using an expert's domain knowledge to derive the causal associations; and the data driven approach derives the mappings from data which can then be validated by the expert [2]. Data-driven approach has gained much popularity in recent years due to its automated nature and its potential to bring new insights to human being.

2 Demonstration

In this demo, we will show AutoBayesian, a data driven tool developed to build Bayesian network based on the casual relationships derived from text mining [3]. It was developed using Microsoft SQL Server 2009 Express edition and a Bayesian network development tool called NETICA. AutoBayesian system has been tested in geriatrics health care. Figure 1 shows a sample Bayesian network derived based on the text mining data. This BN is for fall risk evaluation and management for senior

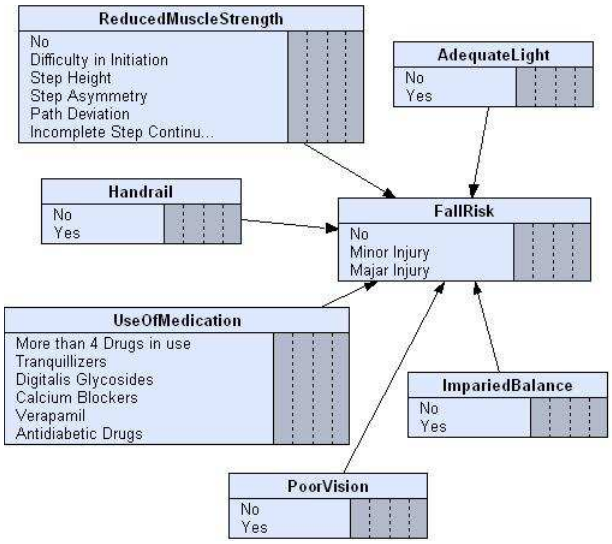


Fig. 1. A Sample Bayesian Network Derived from Text

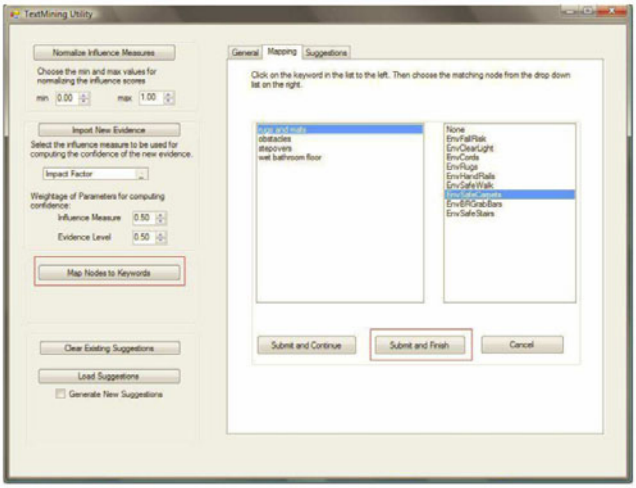
patients. In the demo, we will show step by step how AutoBayesian builds a Bayesian network from text mining information and how it interactively updates the BN when new evidences are observed.

2.1 Derive Confidence Measure

By using existing text mining techniques, causal associations can be extracted from geriatrics health care literature. After the probabilities have been extracted and assessed, we will determine how much confidence we have in the causal associations mined from text. The confidence measure is a score we associate with every causal mapping in the BN based on the confidence we have in asserting that relationship. It quantifies the confidence placed in the causal relationship uncovered by automated methods. In this respect, the two most important parameters we consider are the journal's influence measure and the evidence level of the causal relationship itself. The confidence measure is then computed as a weighted average of the journal's influence measure and the evidence level of the evidence [4].

2.2 Integrate the Causal Mapping with Bayesian Network

Mapping the mined noun phrases to a node in the existing BN is a semantic classification problem and can be solved using information retrieval and/or classification techniques. Using K- nearest neighbor (K-NN) technique, the new noun phrase can be searched in a space containing all the node names. Another method involves use of vector representation of the names of the nodes in the BN. The new noun phrases are also converted into a vector and compared to all the existing vectors to find a match. For a domain which has a large training data, machine learning techniques such as Weight-normalized Complement Naive Bayes (WCNB) will be used. The process of mapping noun phrases to nodes in a BN has to be highly interactive. Therefore, we also provide an interface so that expert can choose to build the mapping between noun phrases to nodes in a BN, as shown in Figure 2. The system will show two lists, one contains the unmapped keywords and the other contains the available nodes in the BN. The expert can manually build a mapping by choosing a keyword and the corresponding mapping node in the BN and then submit it.



Choose the min and max values for normalizing the influence scores

min 0.00 max 1.00

Select the influence measure to be used for computing the confidence of the new evidence.

Impact Factor

Weightage of Parameters for computing confidence:

Influence Measure 0.50

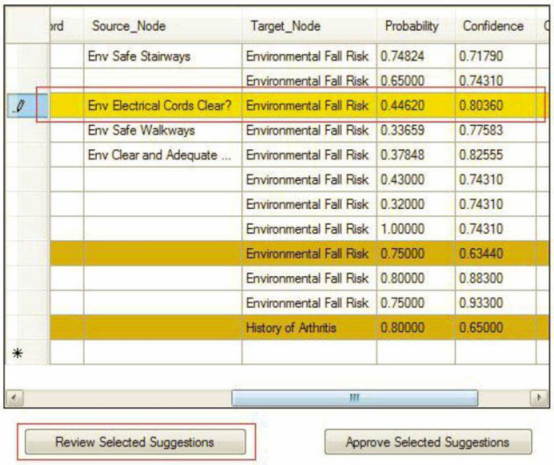
Evidence Level 0.50

Click on the keyword in the list to the left. Then choose the matching node from the drop down list on the right.

rugs and mats
obstacles
stepovers
wet bathroom floor

None
EnvFallRisk
EnvClearLight
EnvCords
EnvRugs
EnvHandRails
EnvSafeWalk
EnvSafeCarpets
EnvBRGrabBars
EnvSafeStairs

Fig. 2. Mapping keywords to Nodes in Bayesian Network



rd	Source_Node	Target_Node	Probability	Confidence
	Env Safe Stairways	Environmental Fall Risk	0.74824	0.71790
		Environmental Fall Risk	0.65000	0.74310
	Env Electrical Cords Clear?	Environmental Fall Risk	0.44620	0.80360
	Env Safe Walkways	Environmental Fall Risk	0.33659	0.77583
	Env Clear and Adequate ...	Environmental Fall Risk	0.37848	0.82555
		Environmental Fall Risk	0.43000	0.74310
		Environmental Fall Risk	0.32000	0.74310
		Environmental Fall Risk	1.00000	0.74310
		Environmental Fall Risk	0.75000	0.63440
		Environmental Fall Risk	0.80000	0.88300
		Environmental Fall Risk	0.75000	0.93300
		History of Arthritis	0.80000	0.65000

Fig. 3. Evidence to be reviewed

Our system consolidates all the evidences and writes out the result into a database table. It identifies all the unique triplets based on the nodes mapped to them and computes the probability and confidence as discussed earlier. The nodes representing cause-effect relation are also written out with the result. If the evidence is new and has no associated representation in the Bayesian Networks, then the triplet along with its probability and confidence is written out as it is but the fields representing the cause-effect nodes are left null to indicate that it is a new causal association. The generated suggestions are then displayed on the screen for review by the expert. For causal associations already existing in the BN, the previous probability and confidence is displayed to facilitate comparison with the newer values. For causal associations which induce loops in the BN, a message is displayed indicating the same. Once the suggestions are generated and displayed on the screen, the expert can choose to automatically accept the suggestions, or to review them by selecting the interesting suggestion, as shown in Figure 3. The system then displays this suggestion as part of the appropriate Bayesian Network. When both the nodes and the link between them exist, only the conditional probability table needs to be updated.

Figure 4 demonstrates a case when both nodes exist in the network but are not linked causally. The system will create the link if it does not induce any loops in the network. Figure 4 shows the BN after applying the new evidence. As shown in the Figure, the new evidence linking the client's gender and arthritis is applied to the BN.

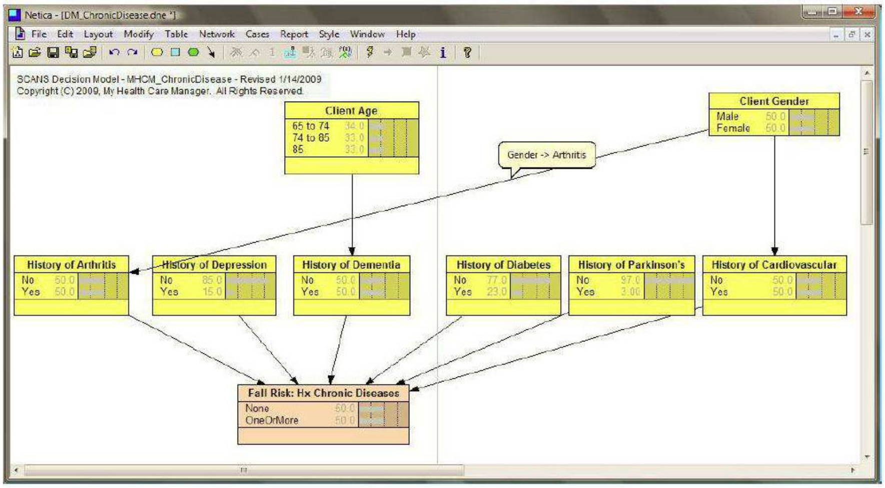


Fig. 4. Bayesian Network after Adding a New Link

References

1. Nadkarni, S., Shenoy, P.: A causal mapping approach to constructing bayesian networks. Decision Support Systems 38, 259–281 (2004)
2. Heckerman, D.: Bayesian networks for data mining. Data Mining and Knowledge Discovery (1996)
3. Raghuram, S., Xia, Y., Palakal, M., Jones, J., Pecenka, D., Tinsley, E., Ban-dos, J., Geesaman, J.: Bridging text mining and bayesian networks. In: Proc. of the Workshop on Intelligent Biomedical Information Systems (2009)
4. Capezuti, E., Zwicker, D., Mezey, M., Fulmer, T.: Evidence-based geriatric nursing protocols for best practice. Springer, Heidelberg (2008)

Classify Uncertain Data with Decision Tree

Biao Qin[1], Yuni Xia[1], Rakesh Sathyesh[1], Jiaqi Ge[1], and Sunil Probhakar[2]

[1] Indiana University Purdue University Indianapolis
{biaoqin,yxia,sathyesr,jiaqge}@cs.iupui.edu
[2] Purdue University West Lafayette
sunil@cs.purdue.edu

Abstract. This demo presents a decision tree based classification system for uncertain data. Decision tree is a commonly used data classification technique. Tree learning algorithms can generate decision tree models from a training data set. When working on uncertain data or probabilistic data, the learning and prediction algorithms need handle the uncertainty cautiously, or else the decision tree could be unreliable and prediction results may be wrong. In this demo, we will present DTU, a new decision tree based classification and prediction system for uncertain data. This tool uses new measures for constructing, pruning and optimizing decision tree. These new measures are computed considering uncertain data probability distribution functions. Based on the new measures, the optimal splitting attributes and splitting values can be identified and used in the decision tree. We will show in this demo that DTU can process various types of uncertainties and it has satisfactory classification performance even when data is highly uncertain.

1 Introduction

Classification is one of the most important data mining techniques. It is used to predict group/class membership for data instances. In many applications, data contains inherent uncertainty. A number of factors contribute to the uncertainty, such as the random nature of the physical data generation and collection process, measurement and decision errors, and data staling. As data uncertainty widely exists, it is important to develop data mining techniques for uncertain and probabilistic data. In this demo, we will show a tool called DTU – Decision Tree for Uncertain Data [1,2], which generates decision-tree based classifier with uncertain data. In DTU, data uncertainty model is incorporated into every step of the tree learning and prediction procedure to achieve higher classification accuracy.

2 Demonstration

The DTU tool is implemented based on the open source data mining tool WEKA. We have also extended the Arff Viewer in Weka so that it can display uncertain data in a proper tabular format as shown in table 1.A dataset can have both uncertain numerical attributes and uncertain categorical attributes. Table 1 shows such an example. Among all the attributes, A_i is an Uncertain Numerical Attribute(UNA) whose precise

Table 1. An uncertain dataset

ID	A_i	…	A_j	class
1	110-120	…	(V_1:0.3; V_2:0.7)	No
2	100-120	…	(V_1:0.3; V_2:0.7)	No
3	60-85	…	(V_1:0.3; V_2:0.7)	No
4	110-145	…	(V_1:0.3; V_2:0.7)	No
5	110-120	…	(V_1:0.3; V_2:0.7)	Yes
6	50-80	…	(V_1:0.3; V_2:0.7)	No
7	170-250	…	(V_1:0.3; V_2:0.7)	No
8	85-100	…	(V_1:0.3; V_2:0.7)	Yes
9	80-100	…	(V_1:0.3; V_2:0.7)	No
10	120-145	…	(V_1:0.3; V_2:0.7)	Yes
11	105-125	…	(V_1:0.3; V_2:0.7)	Yes
12	80-95	…	(V_1:0.3; V_2:0.7)	No

value isunavailable. We only know the range of the A_i of each tuple. A_j is an Uncertain Categorical Attribute (UCA). It can be either V_1or V_2, each with associated probability.

2.1 Decision Tree for Uncertain Data

The core issue in a decision tree induction algorithm is to decide the method of records being split. Each step of the tree-grow process needs to select an attribute test condition to divide the records into smaller subsets. A decision tree algorithm must provide a method for specifying the test condition for different attribute types as well as an objective measure for evaluating the goodness of each test condition.

There are many measures that can be used to determine the best way to split the records. These measures are usually defined in terms of the class distribution of the records before and after splitting. Widely used splitting measures such as information entropy and Gini index are all based on the purity of the nodes and choose the split those results in the highest node purity. These measures are not applicable to uncertain data. For example, for data in Table I, if the splitting condition is $A_i < 115$, it cannot be determined whether instances 1, 2, 4, 5, and 11 belong to the left or right node. Our approach is that when the cutting point of an attribute lies within the uncertain interval of an instance, the instance is split into both branches and the weights of being in both branches are calculated according to the probability distribution function $f(x)$. When the *pdf* (probability distribution function)$f(x)$is unavailable, we can use domain knowledge to find the appropriate *pdf* or assume commonly used distribution such as uniform or Gaussian distribution. An example of such a split is shown in figure 1.

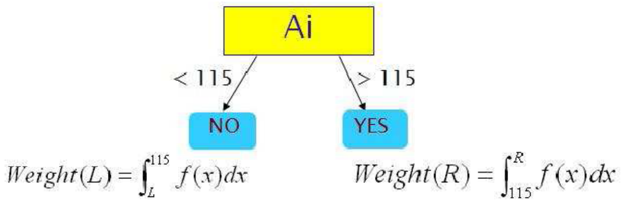

Ai
< 115
> 115
NO
YES
$Weight(L) = \int_L^{115} f(x)dx$
$Weight(R) = \int_{115}^R f(x)dx$

Fig. 1. Uncertain numerical data split

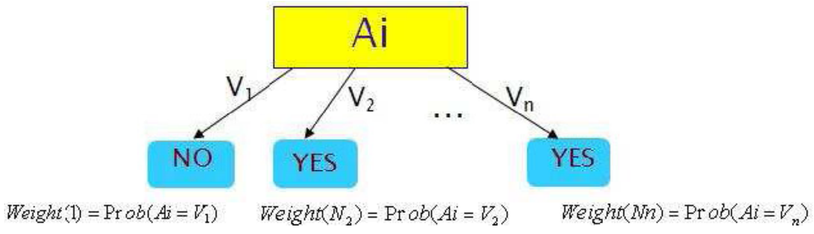

Ai
V_1
V_2
...
V_n
NO
YES
YES
$Weight(1) = \Pr ob(Ai = V_1)$
$Weight(N_2) = \Pr ob(Ai = V_2)$
$Weight(Nn) = \Pr ob(Ai = V_n)$

Fig. 2. An uncertain categorical Data

Splitting based on an UCA *A* is an n-ary split, assume attribute *A* has *n* possible values a_i, (1≤i≤ n), If an uncertain categorical attribute is identified as the best splitting attribute, a branch is created for each known value of the test attribute, and the data are partitioned accordingly. For each value a_l of the attribute, the instance is put into all of the branches with the weight equal to the probability of the attribute to be a_i, as shown in figure 2.

Figure 3 shows the uncertain decision tree for an uncertain glass dataset and the decision tree determines the type of glass based on the oxide content. In case an uncertain data instance covers a test point, it is split into both branches according to the cumulative probability distributions; our visualization routine highlights those leaf nodes in red. The leaf nodes indicate the class type of the node *Ci*, followed by two real values *x/y*. *x* is the total probabilistic cardinality of the node, that is, the total number of instance fall in that node, and *y*is the number of false positives, that is, the number of instance fall in that node but not in class *Ci*. Since both *x* and *y* are calculated according to probability distribution function, they are floating-point numbers instead of integers, which is different from traditional decision tree. Detailed algorithm for uncertain decision tree can be found in [1, 2].

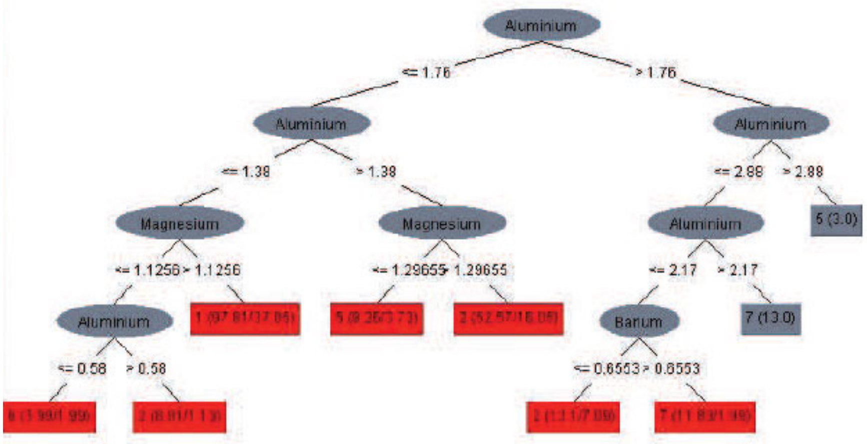
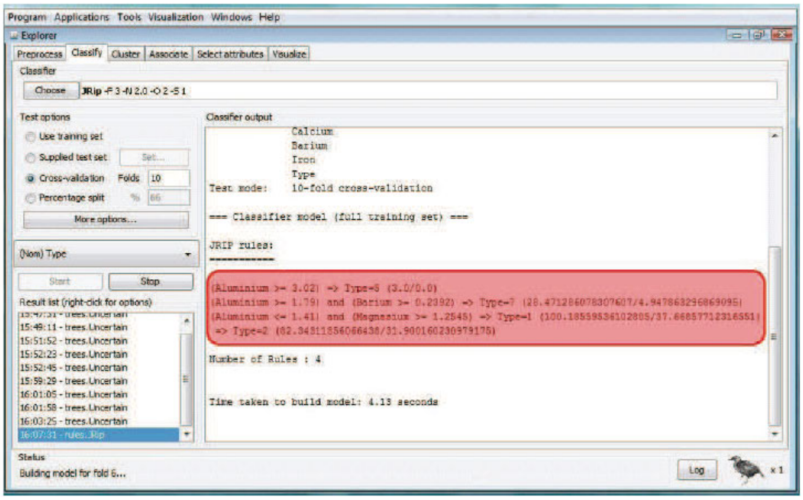


Fig. 3. An uncertain decision tree



Classifier: JRip -F 3 -N 2.0 -O 2 -S 1

Test options: Use training set / Supplied test set / [✓ Cross-validation] Folds 10 / Percentage split % 66

(Nom) Type

Result list (right-click for options)

```
15:49:11 - trees.Uncertain
15:51:52 - trees.Uncertain
15:52:23 - trees.Uncertain
15:52:45 - trees.Uncertain
15:59:29 - trees.Uncertain
16:01:05 - trees.Uncertain
16:01:58 - trees.Uncertain
16:03:25 - trees.Uncertain
16:07:31 - rules.JRip
```

Classifier output

```
              Calcium
              Barium
              Iron
              Type
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

JRIP rules:
===========

(Aluminium >= 3.02) => Type=6 (3.0/0.0)
(Aluminium >= 1.79) and (Barium >= 0.2392) => Type=7 (28.471286078307607/4.947863296869095)
(Aluminium <= 1.41) and (Magnesium >= 1.2545) => Type=1 (100.18559536102805/37.66857712316551)
 => Type=2 (82.34311856066438/31.900160230979175)

Number of Rules : 4


Time taken to build model: 4.13 seconds
```

Status: Building model for fold 6...

Fig. 4. Rule-based Classifier for Uncertain Data

2.2 Comparison with Other Classifiers

In the demo, we will compare the DTU with the traditional decision tree. We will show the difference in splitting condition selection and tree structures when applied to uncertain data. We will also demonstrate that although DTU takes slightly more time in training, it significantly outperforms the traditional decision tree in accuracy on uncertain data sets.

We will also compare DTU with a rule based uncertain classifier uRule [5]. uRule extracts a set of rules of the form *R*: *Condition* => *y* based on uncertain data. The rules show the relationships between the attributes of a dataset and the class label, as shown in figure 4. The red shade area highlights all the uncertain classification rules. We will compare the performance of DTU and uRule on various uncertain data sets with different distributions.

References

1. Qin, B., Xia, Y., Li, F.: DTU: A decision tree for uncertain data. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 4–15. Springer, Heidelberg (2009)
2. Tsang, S., Kao, B., Yip, K.Y., Ho, W.-S., Lee, S.D.: Decision treesfor uncertain data. In: ICDE (2009)
3. Singh, S., Mayfield, C., Prabhakar, S., Shah, R., Hambrusch, S.: Indexing categorical data with uncertainty. In: ICDE, pp. 616–625 (2007)
4. Agrawal, P., Benjelloun, O., Sarma, A.D., Hayworth, C., Nabar, S., Sugihara, T., Widom, J.: Trio: A system for data, uncertainty, andlineage. In: VLDB (2006)
5. Qin, B., Xia, Y., Prabhakar, S., Tu, Y.: A rule-based classification algorithmfor uncertain data. In: Proc. the IEEE Workshop on Managementand Mining of Uncertain Data, MOUND (2009)

StreamFitter: A Real Time Linear Regression Analysis System for Continuous Data Streams

Chandima Hewa Nadungodage[1], Yuni Xia[1], Fang Li[2], Jaehwan John Lee[3], and Jiaqi Ge[1]

[1] Department of Computer & Information Science, Indiana University-Purdue University, Indianapolis, USA
{chewanad,yxia,jge}@cs.iupui.edu

[2] Department of Mathematical Sciences, Indiana University-Purdue University, Indianapolis, USA
fli@math.iupui.edu

[3] Department of Electrical & Computer Engineering, Indiana University-Purdue University, Indianapolis, USA
johnlee@iupui.edu

Abstract. In this demo, we present the StreamFitter system for real-time linear regression analysis on continuous data streams. In order to perform regression on data streams, it is necessary to continuously update the regression model while receiving new data. In this demo, we will present two approaches for on-line, multi-dimensional linear regression analysis of stream data, namely Incremental Mathematical Stream Regression (IMSR) and Approximate Stream Regression (ASR). These methods dynamically recompute the regression model, considering not only the data records of the current window, but also the synopsis of the previous data. Therefore, the refined parameters more accurately model the entire data stream. The demo will show that the proposed methods are not only efficient in time and space, but also generate better fitted regression functions compared to the traditional sliding window methods and well-adapted to data changes.

Keywords: Linear regression, Data streams, Data modeling.

1 Introduction

With the emergence of network and sensor technology in recent times, there is a growing need of data stream management and mining techniques for collecting, querying, and analyzing data streams in real time [1]. Regression analysis is a widely used technique for the modeling and analysis of the relationship between dependent variables and independent variables [4]. In the recent years there is a focus on applying regression analysis techniques to model and predict the behavior of the stream data. In this context, it is required to continuously update the regression model as new data streams in, on the other hand, it is impossible to scan the entire data set multiple times due to the huge volume of the data. Therefore, it is necessary to incrementally reconstruct the regression model using one-scan algorithms. One widely used approach is to consider the current window of data to construct the regression model. Although this approach is efficient in terms of time and space, it has poor performance when accuracy of the model is considered [3].

In this demo, we will present StreamFitter system which will facilitate real-time regression analysis on continuous data streams. It demonstrates two new methods for on-line, multi-dimensional linear regression analysis of stream data namely IMSR (Incremental Mathematical Stream Regression) and ASR (Approximate Stream Regression). Both methods use a window based approach to prevent multiple scans of the entire data set, nevertheless they are able to preserve the accuracy of the regression model by maintaining a synopsis of previous data.

2 Linear Regression Analysis on Data Streams

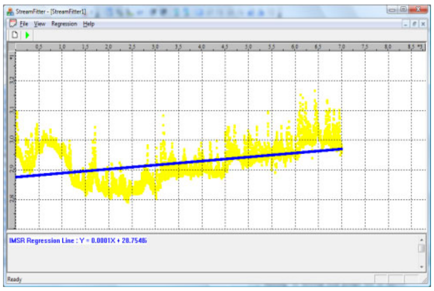
We implemented the StreamFitter System using C++ language. In this demo, we will show and compare three regression approaches: IMSR regression, ASR regression and the original Sliding Window (SW) regression approach, which only consider the records of the current window. We will use real stream data such as the financial data from Tokyo Stock Exchange (TSE) [5] and sea surface temperature (SST) data from Tropical Atmosphere Ocean project (TAO) [6].

2.1 Incremental Mathematical Stream Regression (IMSR)

Assume the window size is n data records and the number of independent variables (IVs) is p. Thus the values of IVs is a $n*p$ matrix, denoted by X. Values of dependent variable is a $n*1$ vector, denoted by y. Values of regression coefficients is $p*1$ vector, denoted by β. This β can be calculated using Ordinary Least Squares (OLS) method as $\beta = (X'X)^{-1}(X'y)$. Hence the regression parameters for the kth window will be calculated as $\beta_k = (X_k'X_k)^{-1}(X_k'y_k)$. However, this calculation is based only on the records in current window. In order to improve the accuracy of the model previous data records should also be considered. Therefore, we should maintain an effective synopsis of pervious data tuples which can be used later.

If values of the IVs for the 1st window is $X1$, values of the IVs for the 2nd window is $X2$, and the total data set available so far is X, it can be proved that $(X'*X) = (X1'*X1 + X2'*X2)$. Furthermore, $X'*X$ is always a matrix of $p*p$, which does not increase in size as more data streams in. Similarly, if values of the dependent variable for the 1st window is $y1$, values of the dependent variable for the 2nd window is $y2$, and the total data set available so far is y, it can be proved that $(X'*y) = (X1'*y1 + X2'*y2)$. $X'*y$ is always a vector of size $p*1$. Therefore to calculate refined β values for a particular window, we only have to maintain the sum of $X'X$ products for previous windows and sum of $X'y$ products for previous windows. Let us refer the sum of $X'X$ products as M and sum of $X'y$ products as V, then the refined parameter vector for the kth window can be computed as $\beta_k = \left(M + X_k'X_k\right)^{-1}\left(V + X_k'y_k\right)$.

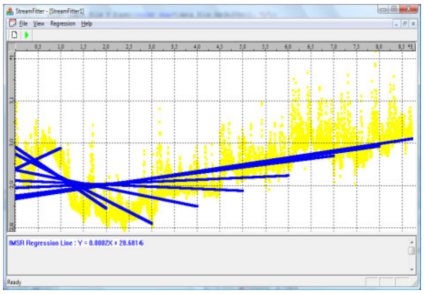
Figure 1 shows IMSR on SST measurements gathered hourly from January 2000 to December 2000, by a moored buoy located in the tropical pacific [6]. There were around 8700 records and we used a window size of 1000 records. Data points are plotted in yellow and IMSR regression line is plotted in blue. Figure 2 shows how IMSR regression line has dynamically adjusted over the time, as the new data streamed in.



IMSR Regression Line : Y = 0.0001X + 28.75486

Ready

Fig. 1. IMSR regression over SST data



IMSR Regression Line : Y = 0.0002X + 28.68146

Ready

Fig. 2. Adjustment of IMSR regression

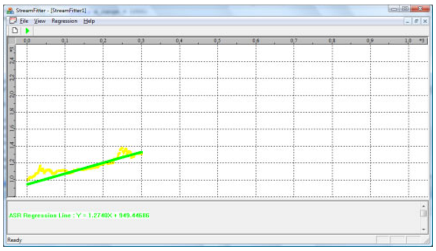
2.2 Approximate Stream Regression

The second regression method we will present is the approximate stream regression (ASR), which refines the values of the parameter vector β for a particular window considering the previous data records. The Brown's Simple Exponential Smoothing or Exponential Weighted Moving Average (EWMA) [2] has been widely used method for time series prediction for a long time. We use this EWMA method to refine the β vector for the current window, considering the β vector calculated from the previous windows. For example, suppose we know the values of the β vector for the (k-1)th window, using that we can calculate the value of the β vector for the kth window as $\beta_k = (1-\alpha) * \beta_k' + (\alpha) * \beta_{k-1}$. [1] Expanding this equation we get, $\beta_k = (1-\alpha)\beta_k' + \cdots + (1-\alpha)(\alpha)^j\beta_{k-j} + \cdots + (\alpha)^k\beta_1$. Refined value is a weighted combination of the previous values and the current value. By varying the smoothing factor α, we can adjust the importance of the historical data. The smaller α, the more weight is assigned to the current data and the less historical data will affect the regression function parameters.

In this demo, we will show that ASR performs well for non-stationary data when the smoothing factor is adjusted to give more weight to the recent samples. Figures 3-5 show ASR regression on daily REIT index of TSE [5] from April 2003 - Feb 2010. There were around 1700 records in this data set, we used a window size of 100 records. Data points are plotted in yellow, and ASR regression line is plotted in green. It is visible how the ASR regression line has dynamically adjusted over the time, as the new data streams in. As visible in figure 5, TSE REIT index has drastically dropped after mid 2007. So the nature of the data stream has significantly changed. ASR is capable of adjusting to this kind of fluctuations in the data stream as it gradually discards the historical records. Here we used α = 0.25, therefore the historical records were rapidly discarded favoring the recent samples. Figure 6 shows a comparison of IMSR and ASR with traditional sliding window (SW) method. It is visible that both IMSR (blue) and ASR (green) has produced better fitted regression lines compared to SW (red).

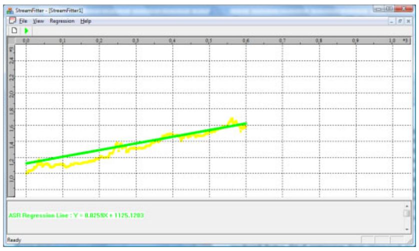
---

[1] **α** – The smoothing factor, 0 ≤ α ≤1. **β'_k** - Parameter vector for the kth window calculated considering only the data records of kth window. **β_{k-1}** - Parameter vector for the (k-1)th window calculated considering all data records seen up to (k-1)th window. **β_k** - Refined parameter vector for the kth window calculated considering all data records seen up to kth window.



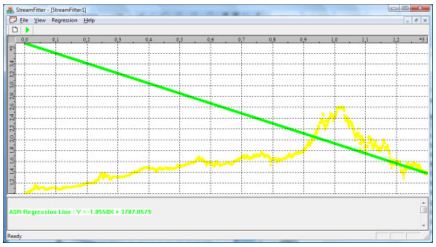
ASR Regression Line : Y = 1.2740X + 949.44686

Fig. 3. ASR at time t



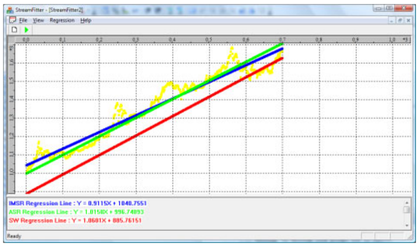
ASR Regression Line : Y = 0.8259X + 1125.1203

Fig. 4. ASR at time t+3n



ASR Regression Line : Y = -1.8550X + 3787.0579

Fig. 5. ASR at time t+kn



IMSR Regression Line : Y = 0.9115X + 1040.7551
ASR Regression Line : Y = 1.0150X + 996.74893
SW Regression Line : Y = 1.0601X + 885.76151

Fig. 6. Comparison of IMSR, ASR, and SW

3 Conclusion

In the demo we will show StreamFitter, a real time linear regression analysis tool for stream data. We will show how the data streams are loaded or received by StreamFitter, how the regression functions are generated and how they are adjusted as new data streams in. We will also compare and visualize various techniques in terms of function fitness and adaptiveness of concept drift.

References

[1] Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and Issues in Data Stream Systems. In: PODS (March 2002)

[2] Brown, R.G., Meyer, R.F.: The fundamental theorem of exponential smoothing. Operations Research 9(5), 673–685 (1961)

[3] Keogh, E., Chu, S., Hart, D., Pazzani, M.: Segmenting Time Series: A Survey And Novel Approach. In: Data Mining in Time Series Databases. World Scientific Publishing Company, Singapore (2004)

[4] Berk, R.A.: Regression Analysis: A Constructive Critique. Sage Publications, Thousand Oaks (2004)

[5] Tokyo Stock Exchange, `http://www.tse.or.jp/english/market/topix/data/index.html`

[6] Tropical Atmosphere Ocean project, `http://www.pmel.noaa.gov/tao/index.shtml`

Challenges in Managing and Mining Large, Heterogeneous Data

Haibo Hu[1], Haixun Wang[2], and Baihua Zheng[3]

[1] Department of Computer Science, Hong Kong Baptist University,
Kowloon Tong, Hong Kong SAR, China
haibo@comp.hkbu.edu.hk
[2] Microsoft Research Asia
Beijing Sigma Center, Hai Dian District, Beijing, China 100190
haixunw@microsoft.com
[3] School of Information Systems, Singapore Management University,
80 Stanford Rd, Singapore 178902
bhzheng@smu.edu.sg

Abstract. Success in various application domains including sensor networks, social networks, and multimedia, has ushered in a new era of information explosion. Despite the diversity of these domains, data acquired by applications in these domains are often voluminous, heterogeneous and containing much uncertainty. They share several common characteristics, which impose new challenges to storing, integrating, and processing these data, especially in the context of data outsourcing and cloud computing.

Some challenges include the following. First, autonomous data acquisition gives rise to privacy and security issues. Therefore, data management and mining must be elastic and privacy-conscious. Second, data is often dynamic and the trend in the data is often unpredictable. This calls for efficient incremental or cumulative algorithms for data management and mining. Load balancing and other real-time technologies are also indispensable for the task. Third, data repositories are distributed. Thus, gathering, coordinating, and integrating heterogeneous data in data management and mining will face unprecedented challenges.

This panel session gives researchers of different background and expertise an opportunity to address these challenging issues together. The main topics of this panel session target the themes in the interdisciplinary domains spreading across database, web, wireless data management, social networking, multimedia, and data outsourcing.

Managing Social Image Tags: Methods and Applications

Aixin Sun and Sourav S. Bhowmick

School of Computer Engineering, Nanyang Technological University, Singapore
{axsun,assourav}@ntu.edu.sg

Abstract. Social tags describe images from many aspects including the visual content observable from the images, the context and usage of images, user opinions and others. We present an overview of a tutorial on social image tags management - an approach to solve the management of tags associated with social images on the Web. Social image tags management defines a set of techniques for measuring effectiveness of tags in describing its annotated resources, discovering relationships between tags and how such knowledge is useful for various tag-based social media management applications such as tag recommendation, tag disambiguation and tag-based browsing systems. This tutorial offers an introduction to these issues and a synopsis of the state-of-the-art.

1 Tutorial Overview

With the advances in digital photography and social media sharing web sites, a huge number of multimedia content is now available online. Most of these sites enable users to annotate web objects including images with free tags. A key consequence of the availability of such tags as meta-data is that it has created a framework that can be effectively exploited to significantly enhance our ability to understand social images. Such understanding paves way to the creation of novel and superior techniques and applications for searching and browsing social images contributed by common users. The objective of this tutorial is to provide a comprehensive background on state-of-the-art techniques for managing tags associated with social images. To the best of my knowledge, this tutorial has not been presented in any major data management conference.

The tutorial is structured as follows. In the first part, we provide a comprehensive understanding of social image tags. We present a brief survey on studies related to motivation behind tagging and impact of various tagging systems that are used by users to create tags. We shall use *Flickr* as an example tagging system to illustrate various concepts. In the second part, we shall describe state-of-the-art techniques for measuring effectiveness of tags in describing its annotated resources (social images). Specifically, we shall describe techniques that enable us to quantitatively measure a tag's ability to describe the image content of social images. Note that this issue is one of the most fundamental problem in multimedia analysis, search, and retrieval. The third part of the tutorial is devoted to describing state-of-the-art techniques for discovering relationships between tags and how such knowledge is useful for various tag-based social media management applications such as tag recommendation, tag disambiguation and tag-based browsing systems. We conclude by identifying potential research directions in this area.

This tutorial is intended for both engineers and researchers. The former will learn about solutions to use and hard problems to avoid. The latter will get a snapshot of the research field and problems that are worth tackling next.

2 Full Description of the Tutorial

The tutorial consists of the following topics.

– **Section 1: Tagging Motivation, Tag Types and Tagging System Taxonomy**
  In this section, we begin by presenting various motivation behind tagging and give an overview of existing tagging systems that enable users to create tags. We discuss different categories of tags and taxonomy of tagging systems, which helps us to understand the impact of various dimensions in a tagging system on the resultant tags associating with their annotated resources.
– **Section 2: Measuring Effectiveness of Tags**
  This section focuses on quantitative measurement of effectiveness of tags in describing its annotated resource. We shall discuss it from two aspects, namely *local descriptiveness* and *global descriptiveness*. *Local descriptiveness* quantifies how effectively a tag describes its annotated resource. We use existing research on *tag relevance* and *re-ranking* as reported in recent literature for our discussion on local descriptiveness. *Global descriptiveness*, on the other hand, is used to discover implicitly developed consensus on describing annotated resources from tags. We use the concept of tag *visual-representativeness* as an example to illustrate this concept.
– **Section 3: Tag Relationships and Applications**
  This section describes state-of-the-art techniques to discover various relationships between tags (e.g., co-occurrence, Flickr distance, and others) and advanced social media data management applications that are mainly based on tag relationships (e.g., tag recommendation, tag disambiguation, event detection, and ontology induction). We shall also give a brief overview of resource browsing systems supported by tags. This will include a live demo of *i*AVATAR, a state-of-the-art tag-based social images browsing system developed by us.
– **Section 4: The Road Ahead**
  We expose potential research issues in analysis and managing social image tags. We also throw some open questions for the audience to ponder about.

3 Speakers

Sourav S Bhowmick and Aixin Sun have published several papers in the area of social image tags management. One of their papers was nominated for the best paper award at the SIGMM Workshop on Social Media (in conjunction with ACM MM 2009). They have also developed a state-of-the-art tag-based social image browsing system called *i*AVATAR which was recently demonstrated in VLDB 2010 held at Singapore. Biographies of Sourav and Aixin can be found at `www.ntu.edu.sg/home/assourav` and `www.ntu.edu.sg/home/axsun/`, respectively.

Web Search and Browse Log Mining: Challenges, Methods, and Applications

Daxin Jiang

Microsoft Research Asia
djiang@microsoft.com

Abstract. Huge amounts of search log data have been accumulated in various search engines. Currently, a commercial search engine receives billions of queries and collects tera-bytes of log data on any single day. Other than search log data, browse logs can be collected by client-side browser plug-ins, which record the browse information if users' permissions are granted. Such massive amounts of search/browse log data, on the one hand, provide great opportunities to mine the wisdom of crowds and improve search results as well as online advertisement. On the other hand, designing effective and efficient methods to clean, model, and process large scale log data also presents great challenges.

In this tutorial, I will focus on mining search and browse log data for search engines. I will start with an introduction of search and browse log data and an overview of frequently-used data summarization in log mining. I will then elaborate how log mining applications enhance the five major components of a search engine, namely, query understanding, document understanding, query-document matching, user understanding, and monitoring and feedbacks. For each aspect, I will survey the major tasks, fundamental principles, and state-of-the-art methods. Finally, I will discuss the challenges and future trends of log data mining.

Keywords: Search and browse logs, log data summarization, log mining applications.

1 Tutorial Description

1. **Introduction**
   (A) *Introduction of search and browse logs*: general formats of search and browse logs, differences between the two types of logs;
   (B) *Overview of log mining applications*: a taxonomy of log mining applications, five popular areas (query understanding, document understanding, query-document matching, user understanding, and monitoring and feedbacks) and important tasks;
   (C) *Frequently-used summarizations of search and browse logs*: query histogram, click-through bipartite, click patterns, and session patterns; construction algorithms, data pre-processing techniques.

2. **Query Understanding by Log Mining**
   (A) *Similar Query Finding*: query expansion, query substitution, and query suggestion; using click-through data to cluster queries; using session data to find similar queries;
   (B) *Query categorization*: categorizing queries into navigational, informational, or transactional ones; classifying queries into a pre-defined set of topics; identifying localizable queries.
3. **Document Understanding by Log Mining**
   (A) *Representing the content of documents*: term-level annotation, query-level annotation, URL annotation by tag data and query logs;
   (B) *Representing the importance of documents*: leveraging user browse patterns to identify pages/sites favored by users.
4. **Query-Document Matching by Log Mining**
   (A) *Mining preference pairs*: observations from eye-tracking experiments, generating training examples from search log data for learning to rank search results;
   (B) *Sequential click models*: user clicks as implicit feedback, position bias in user clicks for Web search, different assumptions about the user behavior on browsing search results, various click models, effectiveness and efficiency.
5. **User Understanding by Log Mining**
   (A) *Personalization*: different user profiles: click-based profiles, term-based profiles, and topic-based profiles;
   (B) *Contextualization*: summarizing context profiles from log data, models for context-aware search.
6. **Monitoring and feedbacks**
   (A) *Monitoring search engine status*: metrics to monitor the status of search engines;
   (B) *Predicting user satisfaction*: sequential patterns, Markov chain models, layered Bayesian networks.
7. **Summary:** *Challenges and Future directions.* Three layers of a log mining system, interpreting users' search intent, specially designed programming languages for log mining, privacy-preserving release of log data.

2 Short Biography

Daxin Jiang, Ph.D., Researcher, Microsoft Research Asia. Daxin Jiang's research focuses on information retrieval and log data mining. He received Ph.D. in computer science from the State University of New York at Buffalo. He has published extensively in prestigious conferences and journals, and served as a PC member of numerous conferences. He received the Best Application Paper Award of SIGKDD'08 and the Runner-up for Best Application Paper Award of SIGKDD'04. Daxin Jiang has been working on development of Microsoft search engines, including Live Search and Bing. Daxin Jiang's publication list can be found at http://research.microsoft.com/en-us/people/djiang/.

Searching, Analyzing and Exploring Databases

Yi Chen[1], Wei Wang[2], and Ziyang Liu[1]

[1] Arizona State University
{yi,ziyang.liu}@asu.edu
[2] University of New South Wales, Australia
weiw@cse.unsw.edu.au

1 Introduction

Keyword based search, analysis and exploration enables users to easily access databases without the need to learn a structured query language and to study possibly complex data schemas. Supporting keyword based search, analysis and exploration on databases has become an emerging hot area in database research and development due to its substantial benefit. Researchers from different disciplines are working together to tackle various challenges in this area.

This tutorial aims at outlining the problem space of supporting keyword based search, analysis and exploration on databases, introducing representative and state-of-the-art techniques that address different aspects of the problem, and discussing further challenges and potential future research directions. The tutorial will provide the researchers and developers a systematic and organized view on the techniques related to this topic.

A tutorial with similar topic was given in SIGMOD 2009 [1][1], and was very well received. Since the research interest in keyword search on structured data is ever increasing and there are plenty of new techniques since then, this tutorial will be updated to incorporate the new findings in this area, which covers query processing, type ahead search, query suggestion, personalization, result comparison, faceted search, etc.

2 Tutorial Outline

2.1 Search Result Definition, Generation, Ranking and Evaluation

The first task in keyword search is to define query results which *automatically* gather *relevant* information that is generally fragmented and scattered across multiple places.

Query Result Definition. A query result on a graph data model is commonly defined as a subtree of the data graph where no node or edge can be removed without losing connectivity or keywords contained in the subtree. Since finding the smallest result, which is the group Steiner tree, is NP-hard, variations and relaxation of the definition have been proposed in order to attain reasonable efficiency. For example, when the data is modeled as a tree, lowest common ancestor (LCA) is a common form to define the query results. Furthermore, besides the data that match query keywords, studies have been performed on identifying data that do not match keywords or on the paths connecting keyword nodes, but are *implicitly* relevant.

[1] http://www.public.asu.edu/~ychen127/keyword_sigmod09_tutorial.pptx

Ranking Functions. It is almost always the case that not all results of a keyword search are equally relevant to a user. Various ranking schemes have been proposed to rank the results so that users can focus on the top ones, which are hopefully the most relevant ones. Many ranking schemes are used in existing works, which consider both the properties of data nodes (e.g., TF*IDF, node weight, and page-rank style ranking, etc.) and the properties of the whole query result (e.g., number of edges, weights on edges, size normalization, redundancy penalty, etc.).

Result Generation and Top-k Query Processing. Algorithms for query result generation and efficient top-k query processing have been developed, and will be introduced in this tutorial. For example, encoding, indexing schemes as well as materialized views have been exploited for processing keyword search on XML. For keyword search on relational databases and graphs, many approaches are based on candidate network (CN) generation, and there are also dynamic programming, heuristics-based approaches (e.g., backward exploration), indexed search approaches, etc., for generating top-k query results.

Evaluation. We will present and discuss evaluation frameworks for keyword search engines. One type of evaluation framework is based on empirical evaluation using benchmark data, such as INEX (INitiative for the Evaluation of XML Retrieval), a benchmark for XML keyword search. Another type of evaluation is evaluating an approach based on a set of axioms that capture broad intuitions.

2.2 Query Suggestion and Result Analysis

To improve search quality and users' search experience, various techniques have been proposed.

Result Snippets. Result snippets should be generated by most Web search engines to compensate the inaccuracy of ranking functions. Generating snippets for structured search results have also been studied. The principle of result snippets is orthogonal to that of ranking functions: letting the user quickly judge the relevance of query results by providing a brief quotable passage of each query result.

Result Clustering and Comparison. Since many keyword queries are ambiguous, it is often desirable to cluster query results based on their similarity, so that the user can quickly browse all possible result types choose the sets of results that are relevant. Besides, techniques have been developed to automatically generate comparison tables for user selected results to help users analyze and differentiate results and get useful insight from the comparison.

Query Cleaning and Suggestion. User issued keyword queries may not be precise. Query cleaning involves semantic linkage and spelling corrections of query keywords, as well as segmentation of nearby query keywords so that each segment corresponds to a high quality data term. Query suggestion is another way of helping user issue queries, which is through suggesting related queries given the query initially submitted by the user. Query suggestion when searching structured data has recently been studied.

2.3 Other Exploration Methods

Query Form Based Database Access. Since structured query languages are highly expressive but difficult to learn, and keyword queries are easy to use but lack the expressive power, a natural idea is to strike a good balance between the two. Existing attempts include generating a large set of query forms and selecting the relevant forms based on user's keyword query, or directly generating a small set of query forms based on either a sample query workload or properties of the schema and data, etc.

Faceted Navigation. A faceted navigation approach generates a navigation tree, either based on the user's keyword query or directly for the entire data collection, in which each level is a classification of the query result. It helps user narrow down the browse scope and find the relevant results quickly. The main challenge is to select the optimal classification at each level of the tree to minimize the expected navigation cost.

2.4 Open Challenges and Future Directions

We will discuss open problems and possible directions for future research. For example, there are many other types of structured data besides normal trees and graphs, whose structures can be exploited to provide high-quality search results. There are many opportunities for supporting keyword search on data models like data warehouses, spatial and multimedia databases, workflows, and probabilistic databases, as well as data extracted from text documents (e.g. parse tree databases). Furthermore, techniques that enable users to seamlessly access vast collections of heterogeneous data sources are also in demand.

Besides, few existing work on searching structured data connects the quality of search results with user needs. In this aspect, there are much we can learn from the Information Retrieval field. In particular, having user involvement during the search process, such as analysis of query log and user click-through streams, will be helpful to provide personalized search experience. Nonetheless keyword search on structured data poses unique challenges on analyzing user preferences.

To summarize, this tutorial presents the state-of-the-art approaches for keyword based search, analysis and exploration on databases with an emphasis on introducing the different modules of a keyword search engine, the research challenges and the state-of-the-art of each module, as well as their relationships. We hope this tutorial will effectively help the audience get a big picture of this research topic.

Acknowledgement

Yi Chen is supported by NSF CAREER award IIS- 0845647 and NSF grant IIS-0740129, IIS-0915438 and IBM faculty award. Wei Wang is supported by ARC Discovery Grants DP0987273 and DP0881779.

References

1. Chen, Y., Wang, W., Liu, Z., Lin, X.: Keyword Search on Structured and Semi-Structured Data. In: SIGMOD Conference, pp. 1005–1010 (2009)

Author Index