

Wook-Shin Han · Mong Li Lee
Agus Muliantara · Ngurah Agus Sanjaya
Bernhard Thalheim · Shuigeng Zhou (Eds.)

LNCS 8505

Database Systems for Advanced Applications

19th International Conference, DASFAA 2014
International Workshops: BDMA, DaMEN, SIM³, UnCrowd
Bali, Indonesia, April 21–24, 2014, Revised Selected Papers

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Zürich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

For further volumes:

<http://www.springer.com/series/7409>

Wook-Shin Han · Mong Li Lee
Agus Muliantara · Ngurah Agus Sanjaya
Bernhard Thalheim · Shuigeng Zhou (Eds.)

Database Systems for Advanced Applications

19th International Conference, DASFAA 2014
International Workshops: BDMA,
DaMEN, SIM³, UnCrowd
Bali, Indonesia, April 21–24, 2014
Revised Selected Papers

Editors

Wook-Shin Han
Pohang University of Science
and Technology (POSTECH)
Pohang
Korea, Republic of (South Korea)

Mong Li Lee
National University of Singapore
Singapore
Singapore

Agus Muliantara
Ngurah Agus Sanjaya
Udayana University
Badung
Indonesia

Bernhard Thalheim
Institut für Informatik
Christian-Albrechts-Universität zu Kiel
Kiel
Germany

Shuigeng Zhou
Fudan University
Shanghai
China

ISSN 0302-9743 ISSN 1611-3349 (electronic)
ISBN 978-3-662-43983-8 ISBN 978-3-662-43984-5 (eBook)
DOI 10.1007/978-3-662-43984-5
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014934170

LNCS Sublibrary: SL3 – Information Systems and Applications, incl. Internet/Web, and HCI

© Springer-Verlag Berlin Heidelberg 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

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

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

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

Preface

Welcome to Bali and the workshops held in conjunction with the 19th International Conference on Database Systems for Advanced Applications (DASFAA 2014).

The objective of the workshops associated with DASFAA 2014 is to give participants the opportunity to present and discuss emerging hot topics related to database systems and applications. For this end, we selected the following four workshops:

1. The Second International DASFAA Workshop on Big Data Management and Analytics (BDMA 2014)
2. The Third International Workshop on Data Management for Emerging Network Infrastructure (DaMEN 2014)
3. The Third International Workshop on Spatial Information Modeling, Management and Mining (SIM³ 2014)
4. DASFAA Workshop on Uncertain and Crowdsourced Data (UnCrowd 2014)

This proceedings contain selected workshop papers. We would like to express our sincere thanks to the hard work of the individual workshop organizers for organizing their workshops, handling the paper submissions, reviewing, and selecting workshop papers to achieve a set of excellent programs.

April 2014

Wook-Shin Han
Ngurah Agus Sanjaya
Shuigeng Zhou

DASFAA 2014 Workshop Organizers

Workshop Co-chairs

Wook-Shin Han

Ngurah Agus Sanjaya

Shuigeng Zhou

POSTECH, South Korea

University Udayana, Indonesia

Fudan University, China

Publication Co-chairs

Mong Li Lee

Agus Muliantara

Bernhard Thalheim

National University of Singapore, Singapore

University Udayana, Indonesia

Christian-Albrechts-University, Kiel, Germany

Second International DASFAA Workshop on Big Data Management and Analytics (BDMA)

Workshop Co-organizers

Laurent d'Orazio

Bonghee Hong

Cyrus Shahabi

Ge Yu

Blaise Pascal University, France

Pusan National University, Korea

University of Southern California, USA

Northeastern University, China

Program Committee

Mirna Adriani

Hung Ngo Ba

Peter Baumann

Yeow Wei Choong

Jrme Darmont

Xiaoyong Du

Hong Gao

Le Gruenwald

Wook-Shin Han

Tan Hanh

Universitas Indonesia, Indonesia

Cantho University, Vietnam

Jacobs University Bremen, Germany

HELP University, Malaysia

Université de Lyon, France

Renmin University, China

Harbin Institute of Technology, China

University of Oklahoma, USA

KyungPook National University, Korea

Posts and Telecommunications Institute of

Technology Ho Chi Minh City, Vietnam

Kyung Hee University, Korea

University of Southern California, USA

Kangwon National University, Korea

Pusan National University, Korea

Byeong-Soo Jeong

Seon Ho Kim

Jinho Kim

Joonho Kwon

Dominique Laurent	University of Cergy Pontoise, France
Sang-goo Lee	Seoul National University, Korea
Sangjun Lee	Soongsil University, Korea
SangKeun Lee	Korea University, Korea
Yoon Joon Lee	KAIST, Korea
Zhanhuai Li	Northwestern Polytechnical University, China
Mondher Maddouri	Taibah University, Tunisia
Philippe Rigaux	CNAM Paris, France
Mohan Sad Hacid	Université de Lyon, France
Stefanie Scherzinger	Hochschule Regensburg, Germany
Hyoseop Shin	Konkuk University, Korea
Ha-Joo Song	Pukyong National University, Korea
Nicolas Spyrtos	University of Paris-South, France
Maria Del Pilar Vilamil	Universidad de Los Andes, Colombia
Guoren Wang	Northeastern University, China
Jongwook Woo	California State University, USA
Jae Soo Yoo	Chungbuk National University, Korea
Aoying Zhou	Eastern China Normal University, China

Third International Workshop on Data Management for Emerging Network Infrastructure (DaMEN)

Workshop Co-organizers

Rui Chen	Hong Kong Baptist University, Hong Kong, SAR China
Minqi Zhou	East China Normal University, China

Program Committee

Gergely Acs	Inria, France
Rui Chen	Hong Kong Baptist University, Hong Kong, SAR China
Zhihong Chong	Southeast University, China
Haibo Hu	Hong Kong Baptist University, Hong Kong, SAR China
Nonman Nohanmmed	McGill University, Canada
Weining Qian	East China Normal University, China
Kai Qin	RMIT University, Australia
Hongzhi Wang	Harbin Institute of Technology, China
Fang Wei	China Mobile Research, China
Linhao Xu	IBM China Research Lab, China
Ying Yan	Microsoft Research Asia, China
Bin Yang	Aarhus University, Denmark
Rong Zhang	East China Normal University, China
Minqi Zhou	East China Normal University, China

Third International Workshop on Spatial Information Modeling, Management and Mining (SIM³)

Workshop Co-organizers

Xin Wang	University of Calgary, Canada
Jun Luo	Huawei Noahs Ark Laboratory, Hong Kong, SAR China
Jihong Guan	Tongji University, China

Program Committee

Michela Bertolotto	University College Dublin, Ireland
Elena Camossi	University College Dublin, Ireland
Christophe Claramunt	Naval Academy Research Institute, France
Haiquan Chen	Valdosta State University, USA
Ke Deng	Huawei Noah's Ark Laboratory, Hong Kong, SAR China
Georg Gartner	Vienna University of Technology, Austria
Yan Huang	University of North Texas, USA
Yoshiharu Ishikawa	Nagoya University, Japan
Bin Jiang	University of Gävle, Sweden
Songnian Li	Ryerson University, Canada
Xiang Li	East China Normal University, China
Steve Liang	University of Calgary, Canada
Eleni Mangina	University College Dublin, Ireland
Gavin Mcardle	National University of Ireland Maynooth, Ireland
Wolfgang Reinhardt	Universität der Bundeswehr München, Germany
Markus Schneider	University of Florida, USA
Ruisheng Wang	University of Calgary, Canada
Shuliang Wang	Wuhan University, China
Ling Yin	Shenzhen Institutes of Advanced Technology, CAS, China
Qiming Zhou	Hong Kong Baptist University, Hong Kong, SAR China
Danielle Ziebeline	Joseph Fourier University, France

DASFAA Workshop on Uncertain and Crowdsourced Data (UnCrowd)

Workshop Organizer

Pierre Senellart	Télécom ParisTech, France
------------------	---------------------------

Program Committee

Talel Abdessalem	Télécom ParisTech, France
Yael Amsterdamer	Tel Aviv University, Israel
Zhifeng Bao	National University of Singapore, Singapore
Bogdan Cautis	Université Paris-Sud, France
Reynold Cheng	Hong Kong University, Hong Kong, SAR China
Valter Crescenzi	Roma Tre University, Italy
Jiaheng Lu	Renmin University of China, China
Zongmin Ma	Northeastern University, China
Silviu Maniu	Hong Kong University, Hong Kong, SAR China
Paolo Merialdo	Roma Tre University, Italy
Atsuyuki Morishima	University of Tsukuba, Japan
Wilfred Ng	Hong Kong University of Science and Technology, Hong Kong, SAR China
Tuyet Trinh Vu	Hanoi University of Science and Technology, Vietnam
Huayu Wu	A*STAR, Singapore

BDMA 2014 Workshop Organizers' Message

The Second International DASFAA Workshop on Big Data Management and Analytics (BDMA 2014) took place on April 21, 2014, in Bali, Indonesia, in conjunction with DASFAA 2014, which is an annual international database conference in the Asia-Pacific region. The objective of BDMA 2014 is to create a dedicated forum to bring together researchers, practitioners, and others to present and exchange ideas, experiences, and the latest research results in big data management and analytics. BDMA 2014 provided an excellent opportunity for researchers from academia and industry as well as practitioners to showcase the latest advances in this area and to discuss future research directions and challenges on big data management and analytics. The workshop's scope includes processing, management, analytics, visualization, integration, and modeling of big data.

We solicit technical papers and position papers (just to include problem identifications and novel approaches to problem solving) that address all important aspects of information technologies for processing and analyzing big data. Topics of interest include big data analytics and visualization, big data management architectures, big data placement, scheduling, and optimization, programming models for big data processing, distributed/parallel processing for streaming big data, big data integration and interoperable big data modeling, real-time processing of streaming big data, and streaming big data applications and challenges. The workshop attracted 21 submissions from France, Korea, China, and the USA. All submissions were peer reviewed by at least two Program Committee members to ensure that high-quality papers were selected. The Program Committee selected 12 papers for inclusion in the workshop proceedings.

The Program Committee of the workshop consisted of 32 experienced researchers and experts. We would like to thank the valuable contribution of all the Program Committee members during the peer-review process. We also would like to acknowledge all the authors who submitted very interesting and impressive papers from their work.

April 2014

Laurent d'Orazio
Cyrus Shahabi
Ge Yu
Hong Bonghee

DaMEN 2014 Workshop Organizers' Message

The emerging network infrastructures such as P2P, mobile and sensor networks, and cloud computing were once lab toys. Nonetheless, they show strong potential to join the mainstream in the foreseeable future. While most network-side issues have been addressed or resolved, the data management issues that arise from the real deployment of these infrastructures are ever increasing. In particular, challenges associated with acquiring, storing, processing, and analyzing large-scale data from these heterogeneous networks call for novel data management techniques. The inherently dynamic nature of these networks further poses new research issues, such as privacy and security. This workshop aims to facilitate the collaboration between researchers in database and networking areas by presenting cutting-edge research topics and methodologies.

The Third International Workshop on Data Management for Emerging Network Infrastructure (DaMEN 2014) was held on April 21, 2014, in Bali, Indonesia, in conjunction with DASFAA 2014. The overall goal of the workshop was to bring together those in academia, researchers, and industrial practitioners from computer science, information systems, network systems, and to provide a forum for recent advances in the field of emerging network infrastructure, from the perspectives of data management.

The workshop attracted 12 submissions from Bangladesh and China. All submissions were peer reviewed by at least three Program Committee members to ensure that high-quality papers were selected. On the basis of the reviews, the Program Committee selected seven papers for inclusion in the workshop proceedings. The Program Committee of the workshop consisted of 28 experienced researchers and experts. We would like to thank the valuable contribution of all the Program Committee members during the peer-review process. Also, we would like to acknowledge the DASFAA 2014 workshop chairs for their great support in ensuring the success of DaMEN 2014. Last but not least, we appreciate all the authors who submitted very interesting and impressive papers from their recent work.

April 2014

Minqi Zhou
Rui Chen

SIM³ 2014 Workshop Organizers' Message

Nowadays, spatial data exist pervasively in various information systems and applications. The unprecedented amount of spatial data that has been amassed and that is being produced at an increasing speed calls for extensive research on spatial information modeling, management, and mining. The Third International Workshop on Spatial Information Modeling, Management and Mining (SIM3-2014) was a half-day workshop held in conjunction with DASFAA 2014. The workshop provides a forum for original research contributions and practical experiences of spatial information modeling, management, and mining. The workshop received ten submissions from Asian, North America, and Europe. Through careful review by the Program Committee, five full papers and two short papers were selected for the presentation and inclusion in the proceedings. The accepted papers are all of excellent quality and cover topics in spatial data management and mining. We grouped the seven accepted papers into two sessions. The program covered a wide range of topics including spatial queries, location-based services and applications, spatial analysis on collusion data and social networks, and sensor web. A successful workshop requires a lot of effort from many people. First, we would like to thank the authors for their contributions, and the Program Committee members for reviewing and selecting papers. In addition, we appreciate DASFAA 2014 workshop co-chairs Drs. Shuigeng Zhou, Wook-Shin Han, and Ngurah Agus Sanjaya for the excellent coordination. Finally, we would like to thank the local Organizing Committee for its wonderful arrangements.

April 2014

Xin Wang
Jun Luo
Jihong Guan

UnCrowd 2014 Workshop Organizer' Message

Crowdsourcing systems utilize human power to perform difficult tasks, such as entity resolution, search, filtering, image matching, or clustering. Typically, data obtained from crowdsourcing platforms are to be considered as uncertain, because of various levels of quality obtained by crowd workers. Modeling, reasoning on, and querying data uncertainty is the general goal of uncertain data management, which has attracted much attention from the research community.

The objective of UnCrowd 2014, the DASFAA Workshop on Uncertain and Crowdsourced Data, was to explore the connections between uncertain data management and crowdsourcing. Three research papers and three vision papers on crowdsourcing, probabilistic data, and the connections between crowdsourcing and uncertainty were accepted by the Program Committee, each article being reviewed by three to four reviewers. The UnCrowd 2014 program also featured a keynote talk by Prof. Lei Chen, from the Hong Kong University of Science and Technology, on “Crowdsourcing over Big Data, Are We There Yet?”.

As PC chair of UnCrowd 2014, I would like to thank the Program Committee for their contributions to the selection of the UnCrowd program, all authors of submitted articles, as well as Lei Chen for doing us the honor of being our keynote speaker. We are also grateful for the help of DASFAA organizers, and for the support of the French Government, who provided a scholarship for a student to attend the workshop, under the framework of the CCIPX Stic-Asia project.

April 2014

Pierre Senellart

Contents

Second International DASFAA Workshop on Big Data Management and Analytics (BDMA)

Meme Media and Knowledge Federation for Exploratory Visual Analytics of Big Data	3
<i>Yuzuru Tanaka</i>	
Online Data Clustering Using Variational Learning of a Hierarchical Dirichlet Process Mixture of Dirichlet Distributions	18
<i>Wentao Fan and Nizar Bouguila</i>	
Distributed Skyline Computation of Vertically Splitted Databases by Using MapReduce	33
<i>Md. Anisuzzaman Siddique, Hao Tian, and Yasuhiko Morimoto</i>	
Short-Term Speed Prediction on Urban Highways by Ensemble Learning with Feature Subset Selection.	46
<i>Mohammad Arif Rasyidi and Kwang Ryel Ryu</i>	
Graph Summarization Using Word Correlation Analysis on Large Set of Documents.	61
<i>Putu Y. Kusmawan and Joonho Kwon</i>	
Distributed K-Distance Indexing Approach for Efficient Shortest Path Discovery on Large Graphs	75
<i>Jihye Hong, Hyunwook Kim, Waqas Nawaz, Kisung Park, Byeong-Soo Jeong, and Young-Koo Lee</i>	
Customized Information Interface with Web Applications	89
<i>Wookey Lee, Suan Lee, and Jinho Kim</i>	
Leveraging Enterprise Application Characteristics to Optimize Incremental Aggregate Maintenance in a Columnar In-Memory Database.	102
<i>Stephan Müller, Paul Möller, and Hasso Plattner</i>	
MaiterStore: A Hot-Aware, High-Performance Key-Value Store for Graph Processing.	117
<i>Dong Chang, Yanfeng Zhang, and Ge Yu</i>	
Vertical Bit-Packing: Optimizing Operations on Bit-Packed Vectors Leveraging SIMD Instructions	132
<i>Martin Faust, Martin Grund, Tim Berning, David Schwalb, and Hasso Plattner</i>	

Efficient Streaming Detection of Hidden Clusters in Big Data
Using Subspace Stream Clustering 146
Marwan Hassani and Thomas Seidl

A Comparison of Systems to Large-Scale Data Access 161
Amin Mesmoudi and Mohand-Saïd Hacid

**Third International Workshop on Data Management
for Emerging Network Infrastructure (DaMEN)**

A Framework to Measure Storage Utilization in Cloud Storage Systems 179
Xiao Zhang, Wan Guo, Zhanhuai Li, Xiaonan Zhao, and Xiao Qin

Personalized Recommendation via Relevance Propagation on Social
Tagging Graph 192
Huiming Li, Hao Li, Zimu Zhang, and Hao Wu

Optimizing Pipelined Execution for Distributed In-Memory OLAP System . . . 204
Li Wang, Lei Zhang, Chengcheng Yu, and Aoying Zhou

Hash^{ed}-Join: Approximate String Similarity Join with Hashing. 217
Peisen Yuan, Chaofeng Sha, and Yi Sun

Minimizing Explanations of Why-Not Questions 230
Chuanyu Zong, Bin Wang, Jing Sun, and Xiao Chun Yang

HadoopM: A Message-Enabled Data Processing System on Large Clusters . . . 243
Wei Pan, Zhanhuai Li, Bo Suo, and Zhuo Wang

AntiqueData: A Proxy to Maintain Computational Transparency in Cloud. . . 256
Himel Dev, Mohammed Eunus Ali, Tanmoy Sen, and Madhusudan Basak

**Third International Workshop on Spatial Information Modeling,
Management and Mining (SIM³)**

Monitoring Query Processing in Mobile Robot Databases 271
Kento Sugiura, Arata Hayashi, Tingting Dong, and Yoshiharu Ishikawa

Efficiently Evaluating Range-Constrained Spatial Keyword Query
on Road Networks 283
Wengen Li, Jihong Guan, and Shuigeng Zhou

A Spatial-Temporal Analysis of Users’ Geographical Patterns
in Social Media: A Case Study on Microblogs 296
Chao Li, Zhongying Zhao, Jun Luo, Ling Yin, and Qiming Zhou

Solving Multiple Bichromatic Mutual Nearest Neighbor Queries
with the GPU 308
Marta Fort and J. Antoni Sellarès

A Kernel Density Method for Aggregating Boundary Collision Data
into Areal Units 317
Ge Cui, Xin Wang, and Dae-Won Kwon

Integrated Indoor Positioning with Mobile Devices for Location-Based
Service Applications 329
Bei Huang and Yang Gao

A Hybrid Scale-Out Cloud-Based Data Service for Worldwide Sensors 342
Tania Khalafbeigi, Chih-Yuan Huang, Steve Liang, and Mea Wang

DASFAA Workshop on Uncertain and Crowdsourced Data (UnCrowd)

Uncertainty in Crowd Data Sourcing Under Structural Constraints 351
Antoine Amarilli, Yael Amsterdamer, and Tova Milo

Integration of Web Sources Under Uncertainty and Dependencies
Using Probabilistic XML 360
M. Lamine Ba, Sebastien Montenez, Ruiming Tang, and Talel Abdessalem

Skill Ontology-Based Model for Quality Assurance in Crowdsourcing 376
*Kinda El Maarry, Wolf-Tilo Balke, Hyunsouk Cho, Seung-won Hwang,
and Yukino Baba*

ProbKS: Keyword Search on Probabilistic Spatial Data. 388
Feng Gao, Rohit Jain, Sunil Prabhakar, and Luo Si

Towards Mobile Sensor-Aware Crowdsourcing: Architecture,
Opportunities and Challenges 403
Jiyin He, Kai Kunze, Christoph Lofi, Sanjay K. Madria, and Stephan Sigg

Conditioning Probabilistic Relational Data with Referential Constraints 413
Ruiming Tang, Dongxu Shao, M. Lamine Ba, and Huayu Wu

Author Index 429

**Second International DASFAA
Workshop on Big Data Management
and Analytics (BDMA)**

Meme Media and Knowledge Federation for Exploratory Visual Analytics of Big Data

Yuzuru Tanaka (✉)

Meme Media Laboratory, Hokkaido University, Sapporo, Japan
tanaka@meme.hokudai.ac.jp

Abstract. This paper proposes the use of meme media and knowledge federation technologies as the basis for a generic framework of exploratory visual analytics of big data. We first propose a “coordinated multiple views” visualization framework, and then extend this to a “coordinated multiple analyses” visualization framework for the integration of clustering tools, frequent pattern mining tools, and statistical analysis tools into the framework. The webtop meme media system Webble World works as the enabling technology to implement these frameworks. Its improvisational knowledge federation capability allows us to improvisationally federate available external tools and services to work together in our framework. This provides our exploratory visual analytics framework with a large library of tools and data sources open for future extension.

Keywords: Big data analysis · Exploratory visual analytics · Coordinated multiple views · Coordinated multiple analyses · Meme media · Clinical trial · Social cyber-physical system

1 Introduction

One of the problems we are facing these days in big data R&Ds may be a big gap between the core technology R&Ds and the application R&Ds. Through the involvement both in EU FP6 and FP7 projects on the integrated IT support of clinical trials on cancers and in Japanese government-initiative project on social cyber-physical systems for optimizing social system services such as the snow plowing and removing in Sapporo City, which has 1.9 million people and the average annual snowfall of 6 m, the current author has been also facing the difficulties to fill in the gap between varieties of available data analysis methods and the goals to find out new meaningful personalized medicine or the optimized resource scheduling for the snow plowing and removing.

Each clinical trial ends up with an accumulation of patient treatment data and patient diagnosis data including DICOM images and patient genomic data. The number of patients in each clinical trial may range from hundreds to thousands. While the data size is not extremely large, their analysis is not so straight forward as applying any “planned-for” analysis scenarios. The objective there is to find out new personalized medical treatments, each of which shows the best recovery rate for some specific group of patients. We need to find out each of these specific patient groups for

which one of the candidate treatment arms after the randomization in the master trial plan may show the best recovery rate than the others. The analysis process there is inherently an exploratory repetition of the hypothesis making through the segmentation of some patient group and the hypothesis checking through some analysis and visualization of the segmented patient group. The personalized medicine has become more focused since it became obvious that the best treatment for a patient is not necessarily the best for another patient who shows the same macroscopic properties as the first one. This means that some macroscopic analysis of a patient group that is segmented with respect to their macroscopic properties does not work well.

The same is true in the analysis of how snowfalls and the snow plowing and removing may influence the traffic of each road segment. Using the average speed in each road segment at every 5 min interval for 24 h, we can characterize each road segment as a vector of 288 dimensions. The same clustering method classifies the consecutive road segments along the same major route in the central Sapporo into the same cluster for the data on the next day of the complete snow plowing and removing, but in different clusters for the data on a day with a heavy snowfall. This indicates that the influence of snow to the traffic is not uniform, and that we cannot apply macro analysis methods to obtain any meaningful knowledge about winter roads.

These two examples have convinced the current author the necessity of exploratory visual analytics.

2 Requirements for Exploratory Visual Analytics

It is well known that exploratory visual analytics [1–4] may use the “coordinated multiple views” visualization framework [5] as its basis. This framework provides more than one view for the visualization of the same database Δ from different aspects as shown in Fig. 1. Each view V_i may be a chart view, a map view, a graph representation view, or a calendar view, and shows the evaluation result $Q_i(\Delta)$ of some query Q_i associated with this view using its specific visualization scheme. Each view allows users to select a set of visualized objects by directly specifying each of them or enclosing some of them, which defines a new additional quantification condition C to quantify the objects stored in the underlying database Δ . This quantification defines a new database view Δ' defined as follows:

```
CREATE VIEW  $\Delta'$ 
AS
SELECT *
FROM  $\Delta$ 
WHERE  $C$ 
```

Each view V_j including V_i itself then immediately changes its visualization from $Q_j(\Delta)$ to $Q_j(\Delta')$, or just highlights the objects in $Q_j(\Delta')$ in the visualization of $Q_j(\Delta)$, depending on the user specification of its visualization mode. Multiple visualization views are mutually coordinated in this sense.

In exploratory visual analytics with a coordinated-multiple-views visualization system, each user may start with the original database Δ , and repetitively try different

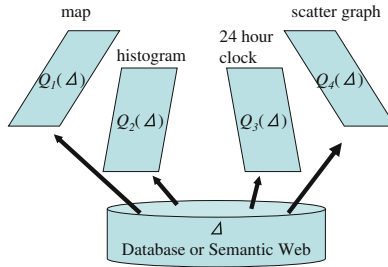


Fig. 1. A coordinated-multiple-views visualization framework.

selections of visual objects on different views for the exploration of different quantifications on database objects to find out a meaningful group of database objects. He or she may roll back the preceding visual object selection to try a different quantification through a different visualization view. Figure 2 schematically shows such a process of exploratory visual analytics.

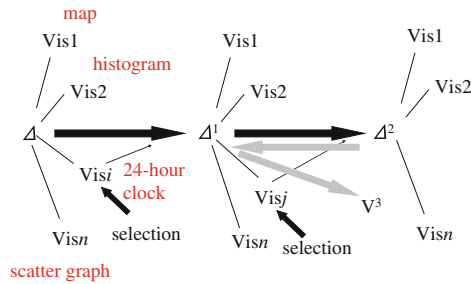


Fig. 2. A process of exploratory visual analytics.

Each view in coordinated-multiple-views visualization is, however, just a database visualization view. No analysis is actually applied in such exploratory visual analytics processes. In order to apply analysis tools to quantified sets of objects, we need to integrate these tools together with their analysis result visualizations into coordinated multiple view visualization. Many researchers emphasized the importance of integrating various analyses and visualizations [6]. To the best of our knowledge, apart from some statistical chart tools to show histograms, correlations, or heatmaps [7], no other analysis tools such as clustering and frequent pattern mining tools have ever been integrated into coordinated multiple views visualization to allow users to directly select a cluster or some of the mined frequent patterns for further quantifying database objects and for further analyzing those quantified database objects. Varieties of analysis tools are rapidly increasing these days. In order to keep our visual analytics at the current state of the art, we need to be able to make any new analysis tools available and operational in our visual analytics.

Exploratory visual analytics requires a coordinated-multiple-views visualization framework to which we can integrate any analysis tools so that their result visualizations may be also coordinated with other visualization views and analysis result

views. It also requires an open library of analysis tools that can be integrated into the framework. These requirements made us to choose meme media technologies [8] as enabling technologies to develop such a framework.

3 A Coordinated-Multiple-Views Framework Based on Meme Media

3.1 Quantification of Database Objects Through Each View

Each view V_i in a coordinated multiple view environment is a chart which stores its query Q_i , issues this query to the underlying database Δ to receive $Q_i(\Delta)$, visualizes this result in its visualization scheme, enables its user to directly specify some of the visualized objects on itself, generates the corresponding quantification condition C , and sends it to the database to modify its view to Δ' . Such a chart can be implemented as a visual object that can communicate with its underlying visual object working as a proxy object of the database as shown in Fig. 3. This underlying database proxy object should be able to accept more than one such a chart object of different types. The chart object and the database proxy visual object exchange the query, the generated quantification condition, and the retrieved result through their I/O port connection.

We like to use such a coordinated-multiple-views environment in a Web portal so that users can access the same environment from anywhere in the world. For this purpose, we use the webtop meme media system Webble World [9] as an enabling technology.

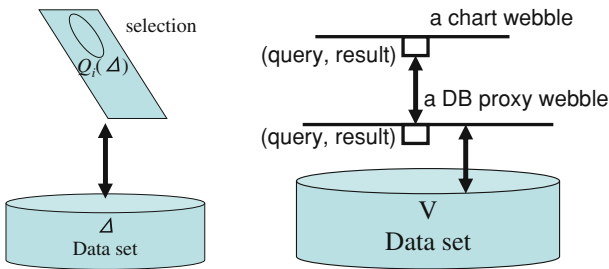


Fig. 3. Each view in a coordinated multiple view visualization.

3.2 Meme Media and Knowledge Federation as Enabling Technologies

The history of meme media research started with an idea of the synthetic media architecture IntelligentPad in 1987 and its implementation in Smalltalk 80 in 1989 [10]. This architecture was a component-based media architecture that exploits the compound document architecture to enable us to embed not only multimedia contents but also visual tools and services in documents. IntelligentPad represents both multimedia contents and functional tools/services as its visual components called pads. Each pad is a card-like visual object on a display screen, wraps some object in itself, and provides a list of I/O ports called slots for its communication with another pad.

A pad can be pasted on another pad. The former becomes a child of the latter. A child pad can connect one of its slots to one of the parent pad slots. If these slots represent variables, their connection makes the child and the parent to share the same variable for data communication between them. Each slot connection may be defined from a pad not only to its direct parent, but also to any of its ancestors. Each pad may have no more than one parent pad, but any number of child pads. In order to migrate any other tool or service into an IntelligentPad environment, we first need to wrap it as a pad.

In 1993, we reoriented our media architecture to meme media and meme pool architectures [8, 11] so that people can publish any composite pads into a world-wide repository, retrieve some composite pads from it, recombine their components in a different way to compose new composite pads, and republish them to the same repository. This architecture makes pads and their world-wide repository respectively work as memes [12] shared by people, and a meme pool. The dissemination of Web browsers after 1995 made us consider how to use the Web as a world-wide meme pool. Around the same time, our collaborators in industry released commercial versions of IntelligentPad developed in C++. Then we came across the problem of how to wrap a Web resource like Web applications or Web services into a meme media object. While the wrapping of a web service is always possible, the wrapping of an application tool may not be always possible. Once being wrapped, however, such resources can be easily combined together to interoperate with each other as pads. We can just combine these pads using the pad pasting operation and the slot connection operation. This idea opened a new vista of knowledge federation, especially improvisational knowledge federation, i.e., dynamic federation of knowledge resources over the Web [13, 14]. This idea, however, required two different system environments, i.e., the Web environment and a meme media system environment. An ideal solution to avoid the use of more than one environment was to unify these two environments, which resulted in the proposal of the Webble World system in 2010 [9].

The current version of the Webble World is a webtop meme media system based on Microsoft Silverlight plug-in technology, which restricts the wrapping of applications. We have already developed generic wrappers for any applications written in R and Octave. The development of similar generic wrappers for Python and Ruby is also possible in principle. We have also wrapped the ArcView of the ArcGIS from ESRI into a webble using the Silverlight version of the ArcView. In order to improve the cross-platform compatibility and to remove such restriction on the wrapping, we have already developed a new HTML5 version Webble World system based on the HTML 5 technology instead of the Silverlight technology. This version enables us to wrap, for example, Google Map into a webble, and to make it interoperate with other webbles in a Webble World environment. Webble World allows us to publish a page with composite webbles as a Web page, and to reuse some components of such webbles in a published Web page for constructing a new composite webble and for publishing it on another Web page. Webble World has made Web documents and the Web respectively work as meme media and a meme pool, i.e., a world wide repository for publishing, reediting, and redistributing meme media objects.

3.3 Composition of a Parallel Coordinate System

Figure 4 shows a single coordinate axis as a visualization view. Its stored query is as follows, where the attribute A denotes its associated database attribute:

```
SELECT A
FROM  $\Delta$ 
WHERE  $v1 \leq A \leq v2$ 
```

This view provides pairs of delimiters for its user to directly specify some intervals for selecting visual objects in each of these intervals. In this example, we assumed that the current selection condition is $v1 \leq A \leq v2$.

A parallel coordinate system [15] is an example of a coordinated-multiple-views system, and can be used together with other coordinated views. It consists of more than one coordinate axis, say n axes, each of which corresponds to some attribute A_i of the database. It also shows for each record over the attribute set $\{A_1, A_2, \dots, A_n\}$ that is retrieved from the database, a polyline which crosses the i -th coordinate axis at the A_i value of this record. Figure 5 shows how such a parallel coordinate system can be composed as a composite webble. It uses n single-coordinate-axis view webbles as shown in Fig. 4, and one polyline chart webble, both of which are connected directly to the underlying database proxy webble. However, each single coordinate axis view webble is put on the polyline chart webble, and defined as its child without any slot connection between them. The polyline chart webble obtains the relative location and the size of each coordinate-axis view webble when the latter is put on the former. The former can read the reserved slot of the latter storing its associated database attribute name. Using the information about these, the polyline chart webble issues the following query to the database, and draws a polyline for each of the retrieved records so that it may cross the i -th coordinate axis drawn by the i -th single-coordinate-axis view webble at the A_i attribute value of this record.

```
SELECT  $A_1, A_2, \dots, A_n$ 
FROM  $\Delta$ 
```

3.4 TOB Based on Our Coordinated-Multiple-Views Framework

Figure 6 shows the exploratory visual analytics mode of TOB (Trial Outline Builder) [16]. This is a webtop environment developed using Webble World. The upper

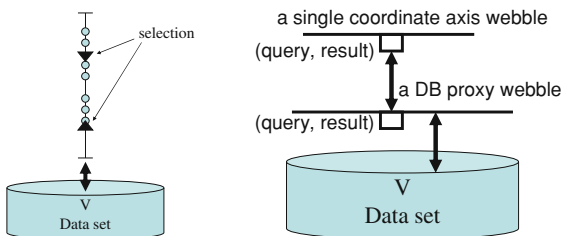


Fig. 4. A single coordinate axis as a view in a coordinated multiple views visualization.

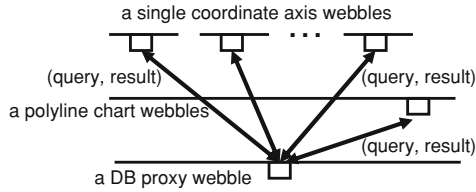


Fig. 5. The composition of a parallel coordinate system.

window shows a master clinical trial plan. The lower window shows the visual analytics environment. The upper part in this lower window shows a parallel coordinate visualization of patients with such coordinates as the hometown, the target organ of radio therapy, whether the patient has metastasis, and the selected treatment arm after the randomization. The hometown coordinate is represented by a geographical map. Such a map is just an extension of a single-coordinate-axis view. This extension, however, requires some extension of the composite webble architecture described in Fig. 5, which is beyond the scope of this paper. The bottom left chart is a life table showing the temporal decrease of the patient survival rate. The bottom right webble shows a heatmap visualization of the patients' gene expression data, where each row represents a patient, and each column represents a gene. This TOB system allows us to concurrently specify more than one selection of visual objects on each view. Visualizations corresponding to different selections may use different dedicated colors for highlighting visual objects in each view. If we separately select each of the two different treatment arms, as well as the case without applying randomization, on the leftmost single-coordinate-axis of the parallel coordinate system, the life table will show three different survival rate curves in three different colors. The case without applying randomization mainly corresponds to the case with metastasis. While we explore different patient quantifications, we may come across some specific case in which one of the two treatment arms show significantly a better survival rate than the other. This implies that the better-performance arm may become a good candidate of personalized medicine for this quantified set of patients.

3.5 Geospatial Digital Dashboard Based on Our “Coordinated Multiple Views” Framework

Figure 7 shows another “coordinated multiple views” visualization system using the Webble World technology.

This system called Geospatial Digital Dashboard was developed for the exploratory visual analytics of social cyber-physical data related to the winter road management in Sapporo City. It consists of several different views. The map view shows average taxi speed in each direction of each road segment, and the distribution of tweets with geotags. It allows us to specify a rectangular area to select only those road segments in this area. The clock view at the bottom right corner shows the population of taxis in each of the 24 h as the area of each circle around the circumference of the

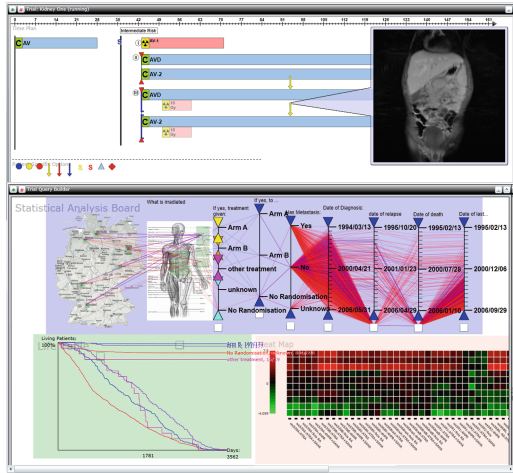


Fig. 6. Trial Outline Builder as a coordinated multiple view visualization system

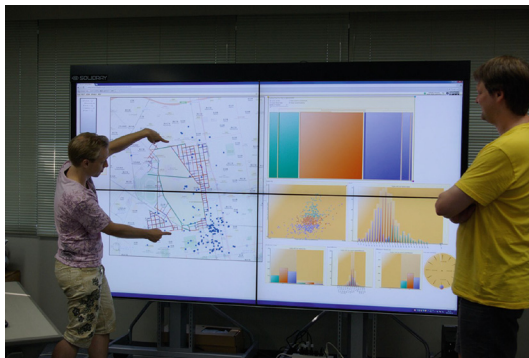


Fig. 7. Geospatial Digital Dashboard as a coordinated multiple views visualization system

clock circle. It allows us to select time intervals for selecting only those statistical probe car data in the specified time intervals. Other views include various kinds of charts such as correlation charts and histograms.

4 Coordinated-Multiple-Analyses Framework Based on Meme Media

Now we need to integrate varieties of analysis tools and their result visualizations into our coordinated-multiple-views framework. We call such an integrated framework a coordinated-multiple-analyses framework.

4.1 Integration with Clustering Tools and Their Result Visualizations

Let us first consider the integration of clustering tools and their result visualizations into the coordinated-multiple-views framework. The result of any clustering applied to objects identified by the values of some attribute A of the underlying database can be considered as a relation $\text{Cluster}(A, \text{ClusterID})$, where the values of A work as the object IDs of objects that are clustered, and ClusterID denotes the ID of each cluster. This relation $\text{Cluster}(A, \text{ClusterID})$ can be visualized in one of various visualization schemes. Each visualization needs to provide a direct manipulation operation for users to select some objects or some clusters. Such a direct selection corresponds to a quantification condition on the attribute A or ClusterID , which further quantifies the underlying database objects. Such a clustering tool with its result visualization can be easily implemented as a webble. Figure 8 shows an extended Geospatial Digital Dashboard with the integration of a clustering tool. It has two clustering visualization views in its rightmost area. Each rectangle in each of them represents a cluster. Its size is proportional to the cluster size, i.e., the number of different A attribute values in the cluster. Each of these clustering result views also shows the phylogenetic tree of clusters over these clusters. In this example, road segments are clustered in terms of the daily change of the number of taxis and the daily change of the average taxi speed in each road segment. These values are available for each 5 min interval. Therefore, the changes of these values characterize each road segment as two different vectors of 288 dimensions. For each of these two vector representations of each road segment, we clustered the road segments based on the similarity of their vector representations.

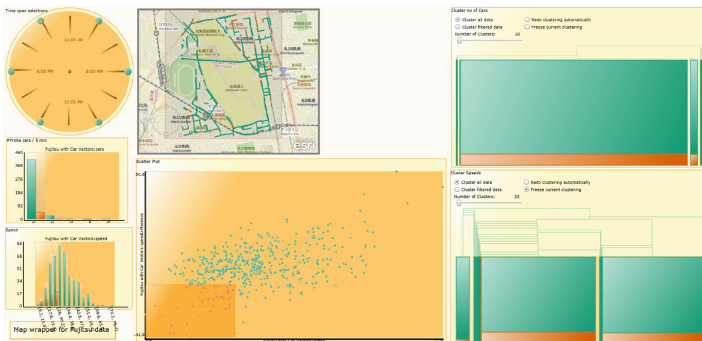


Fig. 8. Extended Geospatial Digital Dashboard with the integration of a clustering tool.

4.2 Integration with Frequent Pattern Mining Tools and Their Result Visualizations

Let us now consider the integration of frequent pattern mining tools and their result visualizations into the coordinated-multiple-views framework. Any frequent pattern mining result can be represented by two relations, $\text{Mining}(\text{Pattern}, \text{Supp}, \text{Conf})$ and $\text{Include}(A, \text{Pattern})$. The first relation lists up each frequent pattern with its support and confidence indices. The second relation tells which objects among those identified

by the attribute value of A include each of the mined frequent patterns. The first relation can be visualized in various visualization schemes to list up mined frequent patterns together with their support and confidence index values. Each visualization needs to provide a direct manipulation operation for users to change the threshold values of support and confidence indices to list up only those patterns with their support and confidence indices higher than the thresholds, and further to directly select some frequent patterns in the list. Using the second relation, this selection of some patterns is converted to the corresponding quantification condition on A attribute, which further quantifies the underlying database and changes the other coordinated visualization views. Figure 9 shows an extension of Geospatial Digital Dashboard with the integration of a frequent pattern mining tool as well as a clustering tool.

In the top right heatmap chart, each row represents each 5 min interval in a day, and each column represents a road segment. The intensity of each cell represents the average speed in the corresponding road segment during the corresponding 5 min interval. In general, a heatmap can be defined for two nominal value attributes A1, A2, and one numerical value attribute A3 that are arbitrarily selected out of the underlying database attributes. Its row and column represent the domains of A1 and A2 respectively, and each cell shows the numerical value of A3 as the color intensity. In this example, the attribute A1 is the Time attribute representing each 5 min interval, the attribute A2 is the Road Segment ID represented by the geo-locations of two end points of each road segment, and the attribute A3 is the Average Speed of taxis in this road segment at this time interval.

On the left hand side of the heatmap, we have a color gradient bar showing how different intensities are mapped to different colors. This bar allows its user to specify more than one intensity interval. If he or she specifies k intervals, the heatmap webble generates k items for each of its columns, and will interpret each row as a transaction, and each column as k items. Each item in each transaction has a binary value.

We can apply item set mining and association rule mining to this set of transactions. In Fig. 9, we specified only one intensity interval to focus on the average speed lower than 10 km/h. We wanted to focus on traffic jams on road segments. Figure 9 shows only the result of applying the association rule mining to the set of transactions.

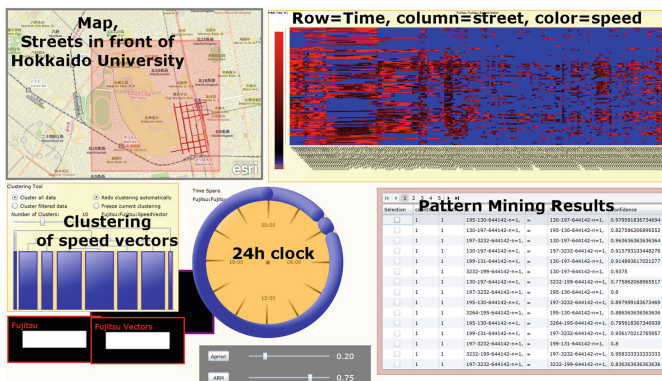


Fig. 9. Extended Geospatial Dashboard integrated with a frequent pattern mining tool.

At the bottom center, this figure shows two sliders for users to specify threshold values of the support and the confidence. This association rule mining result provides a check box in front of each mined association rule. These check boxes are used to select some of the mined patterns. This selection quantifies those transactions having one of these selected patterns in the heatmap. Unselected transactions, or rows, will be dimmed in the heatmap. The mining result webble also allows its user to select each item appearing in each selected pattern. Since each item in this example is a road segment, you may pick up a mined association rule to see how the traffic jam in some road segments may propagate to other road segments on the map view.

Figure 10 shows such a traffic jam dependency we mined using the extended Geospatial Digital Dashboard. In this example, we selected the three mined association rules marked by arrows. These three rules indicate that a traffic jam in the road segment labeled with “pre-condition” propagates along this north bound one way road to two other road segments labeled with “post-condition”. These road segments are found to be consecutive. This example shows a potentiality of the exploratory visual analytics with the Geospatial Digital Dashboard for discovering traffic jam dependency rules, and possibly for identifying those initial road segments from which traffic jams expand.

Figure 11 shows an extension of the TOB with the integration of a heatmap webble to visualize the patient gene expression intensity and an association-rule-mining webble. In this heatmap, each row represents a patient, and each column represents a gene.

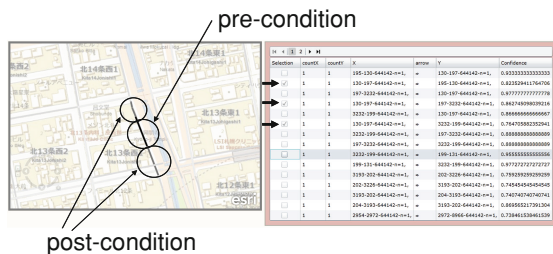


Fig. 10. Traffic jam dependency we found using the association rule mining in Fig. 9.

Users can specify k intensity intervals of the heatmap to generate k items for each gene. We specified only two intervals, the high gene-expression region and the low gene-expression region. Roughly speaking, for each gene, the two items, the high-intensity item and the low-intensity item respectively correspond to the activation and the inhibition of this gene expression in each patient. While the TOB in Fig. 6 enables us to analyze the clinical trial data only from the phenotype view of patients, this extended TOB in Fig. 11 enables us to analyze both clinical trial data and patient genomic data through repetitively changing the view between the phenotype view and the genotype view of patients. Even if a group of patients quantified only from their phenotype characteristics does not show any significant difference of the survival ratio among different candidate treatment arms, the association rule mining of their

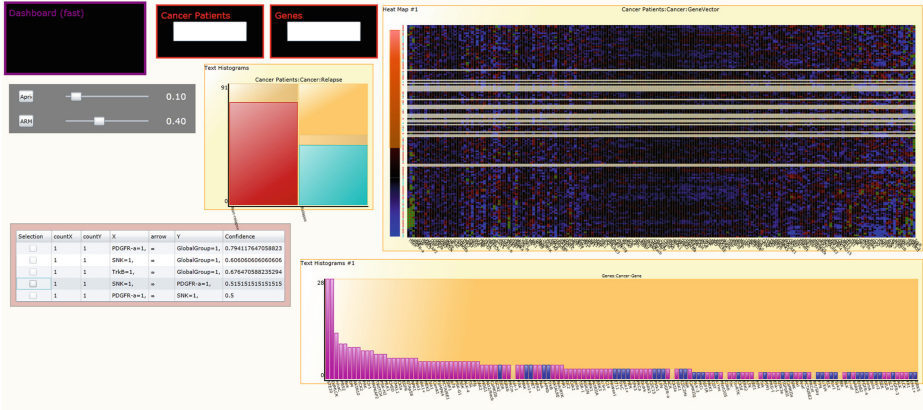


Fig. 11. An extension of the TOB with the integration of a heatmap webble to visualize the patient gene expression intensity and an association-rule-mining webble.

high and low gene expression may find out some association rules with their support and confidence higher than the thresholds, and each of them may further quantify a different subgroup of patients. If, in some of these subgroups, one candidate arm shows a significantly better survival ratio than the others, this association rule may work as a good biomarker to further segment patients for applying this personalized treatment.

4.3 Integration with Statistical Analysis Tools and Their Result Visualization

Let us now consider the integration of statistical analysis tools and their result visualizations into the coordinated-multiple-views framework. Any statistical analysis specifies the group-by attributes and, for each group of records, some aggregate function to calculate the aggregate value. The result can be represented as a relation Stat (GBattributes, Afunction), where the attribute GBattributes is a list of attributes specified as group-by attributes, and the attribute Afunction is a derived attribute whose value is obtained by applying the specified aggregate function such as average, count, minimum, maximum, and correlation to the set of values of the specified attribute in each group. This specified attribute is called the measure attribute. This relation can be visualized in various visualization schemes. Each visualization needs to provide a direct manipulation operation to quantify the GBattributes value and the Afunction value. This quantification further quantify the values of the database attributes in GBattributes, which modifies the underlying database view. Our taxi probe car data are stored in a relation Taxi (Date, Time, RoadSegment, Speed, NumberOfCars, MaxSpeed). For GBattribute = (Date, RoadSegment) and Afunction = average (Speed × NumberOfCars), the attribute Afunction takes the value of the

average taxi traffic flow in each day. If we quantify the average taxi traffic flow to be higher than a specified threshold, we will obtain, for each day, those road segments satisfying this quantification. This quantification modifies the database view, and changes the other view visualizations and analysis visualizations in the coordinated-multiple-analyses system.

5 Similar Systems

Here we give a brief overview of other systems for visualization and exploration of data, and point out some differences between them and our system.

SpotFire [17] is a system using coordinated multiple views for exploring and visualizing data. It has many visualization tools and supports zooming, interactive query modification, details on demand, and more. It was not built for data mash-up (combining data from many sources), and it does not have components for generating new complex data from the data.

Tioga-2 [18] (now Tioga DataSplash) uses direct manipulation to visually explore database contents. Multiple visualizations can be coordinated. Tioga has fairly few visualization primitives and expects a relational databases as the data source. DEVise [19] is also a data exploration system for relational databases, and also supports coordinated multiple views of the data. It supports many types of operations on the data, but the user interaction with the visualization results is limited.

Snap-Together Visualization [20] (Snap) has many similarities with our system. Visualization components can be coordinated so that e.g. selecting data using one component is automatically reflected in other coordinated components, and further interactions in these are also reflected in the first component. There are several different types of coordination, so selections in one component can select related items in another view, and it can open up details about the selection in a third view, etc. Snap, like our system, requires components to adhere to a small component interface and it is possible to wrap existing software with interface wrappers (in for example Visual Basic) to allow pre-existing software as components in the Snap system. Two difference from our system are that Snap expects the data to come from an ODBC database and that it uses Microsoft's COM for component (process) communication.

RapidMiner [21] is an open-source system. It is a widely used prototyping system for knowledge discovery and data mining. RapidMiner supports very many data mining and machine learning algorithms. Work flows are set up using a graphical interface, and multiple views of the same data are supported. Going back and changing something in the work flow will change the visualization results. Like our system, RapidMiner hides the underlying data format from the data mining components or visualization components, and changes in a work flow are reflected in all views of this work flow. While the work flow set up process is graphical, interaction with the visualization results is not possible.

6 Conclusion

Exploratory visual analysis requires the following features.

- (1) It should support the repetition of the hypothesis making through data segmentation of a specific set of data and the hypothesis checking through data analysis and visualization of the segmented data set. The analysis result may be also used for further data segmentation.
- (2) It should provide a large library of analysis and visualization tools open for the future extension. It should be easy to improvisationally wrap external tools and services into components and to register them into the library for their future reuse in the visual analytics environment through the improvisational federation of them with other tools and services..
- (3) It should allow users to improvisationally bring external data sources provided as web services into the visual analytics environment.

The first requirement made us propose a coordinated-multiple-analyses visualization framework as an extension of the coordinated-multiple-views visualization environment. We used the webtop meme media technology as the enabling technology for these frameworks.

Exploratory visual analytics requires various analysis tools and data sources. Its system should be open for the future integration of new analysis tools and new data sources to itself. Our framework is based on the webtop meme media system *Webble World*, which enables us to improvisationally wrap both varieties of tools developed in R, Octave, Python, and Ruby, and any analysis and/or data providing web services into *webbles*. These *webbles* can be registered into the open library to increase its variety. Users can improvisationally federate any of these wrapped tools and services to work together. For example, in Fig. 7, the map view shows the geographical distribution of tweets. These tweets are obtained from the twitter service. We improvisationally wrapped the twitter service into a *webble*, and improvisationally federated this *webble* with the map view *webble* to obtain this visualization.

These days we have a huge variety of related open data sources over the Web. They can be accessed through web services. It is important for us to be able to improvisationally federate these data sources with our visual analytics environment. We can also find out a huge variety of open tools and services for data analysis and visualization. The improvisational knowledge federation capability of *Webble World* will allow us to improvisationally wrap a large portion of them into *webbles*, and to reuse them in cooperation with other data sources, tools and services in our exploratory visual analytics environment.

References

1. Thomas, J., Cook, K.: *Illuminating the Path: Research and Development Agenda for Visual Analytics*. IEEE-Press, Los Alamitos (2005)
2. Thomas, J., Kielman, J.: Challenges for visual analytics. *Inf. Vis. J.* **11**, 309–314 (2009). Special Issue: Foundations and Frontiers of Visual Analytics

3. Tukey, J.W.: *Exploratory Data Analysis*. Addison-Wesley, Reading (1977)
4. Keim, D.A., Mansmann, F., Stoel, A., Ziegler, H.: *Visual Analytics*. *Encyclopedia of Database Systems*. Springer (2009)
5. Roberts, J. C.: State of the art: coordinated and multiple views in exploratory visualization. In: 5th International Conference on Coordinated and Multiple Views in Exploratory Visualization, Zurich, pp. 61–71, July 2007
6. Keim, D.A., Mansmann, F., Thomas, J.: Visual analytics: how much visualization and how much analytics. *ACM SIGKDD Explor. Newsl.* **11**(2), 5–8 (2009)
7. Perer, A., Shneiderman, B.: Integrating statistics and visualization for exploratory power: from long-term case studies to design guidelines. *IEEE Comput. Graph. Appl.* **29**, 39–51 (2009)
8. Tanaka, Y.: *Meme Media and Meme Market Architectures: Knowledge Media for Editing, Distributing, and Managing Intellectual Resources*. IEEE Press and Wiley-Interscience, New York (2003)
9. Kuwahara, M., Tanaka, Y.: Webble world – a web-based knowledge federation framework for programmable and customizable Media objects. In: *Proceedings of the IET International Conference on Frontier Computing Theory, Technologies and Applications*, Taichung, Taiwan, IET inspec, pp 372–377, 4–6 August 2010
10. Tanaka, Y., Imataki, T.: IntelligentPad: a hypermedia system allowing functional composition of active media objects through direct manipulations. In: *Proceedings of the IFIP 11th World Computer Congress*, San Francisco, USA, pp. 541–546 (1989)
11. Tanaka, Y.: From augmentation to Meme Media. In: *Proceedings of the ED-Media 94*, Vancouver, pp. 58–63, June 1994
12. Dawkins, R.: *The Selfish Gene*. Oxford University Press, Oxford (1976)
13. Itoh, K., Tanaka, Y.: A visual environment for web application composition. In: *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*, Nottingham, pp. 184–193, August 2003
14. Tanaka, Y., Fujima, J., Ohigashi, M.: Meme Media for the Knowledge Federation Over the Web and Pervasive Computing Environments. In: Maher, M.J. (ed.) *ASIAN 2004*. LNCS, vol. 3321, pp. 33–47. Springer, Heidelberg (2004)
15. Inselberg, A., Dimsdale, B.: Parallel coordinates: a tool for visualizing multi-dimensional geometry. In: *VIS'90: Proceedings of the 1st Conference on Visualization '90*, Los Alamitos, CA, USA, pp. 361–378 (1990)
16. Sjöbergh, J., Kuwahara, M., Tanaka, Y.: Visualizing clinical trial data using pluggable components. In: *Proceedings of the 15th International Conference on Information Visualization IV 2012*, Montpelier, pp. 291–296, July 2012
17. Ahlberg, C.: Spotfire: an information exploration environment. *SIGMOD Rec.* **25**(4), 25–29 (1996)
18. Aiken, A., Chen, J., Stonebraker, M., Woodruff, A.: Tioga-2: a direct manipulation database visualization environment. In: *Proceedings of ICDE'96*, New Orleans, LA, USA, pp. 208–217 (1996)
19. Livny, M., Ramakrishnan, R., Beyer, K., Chen, G., Donjerkovic, D., Lawande, S., Myllymaki, J., Wenger, K.: DEVise: integrated querying and visual exploration of large datasets. In: *Proceedings of SIGMOD'97*, Tucson, AZ, USA, pp. 301–312 (1997)
20. North, C., Shneiderman, B.: Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In: *Proceedings of AVI'00*, Palermo, Italy, pp. 128–135 (2000)
21. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: YALE: Rapid prototyping for complex data mining tasks. In: *KDD'06: Proceedings of the 12th ACM SIGKDD*, Philadelphia, PA, USA, pp. 935–940 (2006)

Online Data Clustering Using Variational Learning of a Hierarchical Dirichlet Process Mixture of Dirichlet Distributions

Wentao Fan and Nizar Bouguila^(✉)

Concordia Institute for Information Systems Engineering,
Concordia University, Montreal, QC, Canada
{wenta_fa,nizar.bouguila}@encs.concordia.ca

Abstract. This paper proposes an online clustering approach based on both hierarchical Dirichlet processes and Dirichlet distributions. The deployment of hierarchical Dirichlet processes allows to resolve difficulties related to model selection thanks to its nonparametric nature that arises in the face of unknown number of mixture components. The consideration of the Dirichlet distribution is justified by its high flexibility for non-Gaussian data modeling as shown in several previous works. The resulting statistical model is learned using variational Bayes and is evaluated via a challenging application namely images clustering. The obtained results show the merits of the proposed statistical framework.

Keywords: Mixture models · Dirichlet distribution · Variational inference · Hierarchical Dirichlet process · Online learning · Image clustering

1 Introduction

With the ubiquity of new information technology and media, the amount of multimedia data generated everyday has increased exponentially. Handling the resulting massive data sets is a difficult problem [19, 20, 24, 33]. Fortunately, advances in statistics and computing have made available several data modeling tools and approaches in many areas such as pattern recognition, computer vision, and data mining. Among these approaches finite mixture models play a crucial role and have become fundamental tools for data analysis [9]. The efficient adoption of finite mixture models, however, presents itself serious challenges related mainly to the important model selection problem (i.e. automatic determination of the model complexity without under- or over-fitting). Thus, much recent research has been directed at data modeling using infinite mixtures rather than finite ones. Indeed, as we can see from advances in the area of machine learning, Bayesian nonparametric approaches have been widely studied and adopted recently [26, 31]. This is especially true for Dirichlet process (DP) mixtures of distributions [10, 11, 19, 25].

DP mixtures of Gaussian distributions have been largely adopted in the past. In a previous work, however, we have shown that DP mixtures of Dirichlet distributions could be a better alternative especially in the case of non-Gaussian

data [4]. A DP mixture of Dirichlet distributions can be viewed as a learning machine which estimates a given probability density function as an infinite weighted sum of Dirichlet distributions. This learning machine has been shown to be effective in several data mining and computer vision applications and has been proposed as an alternative to overcome the drawbacks of finite Dirichlet mixture models [4]. In this paper, we go a step further by taking advantage of the flexibility that hierarchical Bayesian modeling offers via the development of a hierarchical DP process mixture of Dirichlet distributions. A hierarchical DP [32] is actually a dependency model for multiple Dirichlet processes. It has been shown to be an efficient nonparametric Bayesian approach to the problem of model-based clustering of grouped data with sharing clusters [8, 30]. It is an extension to the conventional DP with a Bayesian hierarchy where the base measure for a set of Dirichlet processes is itself distributed according to a DP. Learning technique for DP-based models are generally designed to be run over already observed collections of objects. In several real applications, however, the collection grows over time which makes the use of batch learning algorithms infeasible. In this case, we should consider online learning algorithms, which allow to update the model’s parameters each time new objects are observed, by maintaining high-quality inference for new introduced data [7]. We develop then an online variational algorithm for the learning of our hierarchical DP mixture of Dirichlet distributions model. The adoption of variational Bayesian inference [1] is motivated by the fact that it has been shown to be an efficient alternative to purely Bayesian inference in the case of several nonparametric Bayesian models [13] and especially in the case of Dirichlet mixture models [14].

The paper is organized as follows. In Sect. 2 we present our hierarchical non-parametric model. In Sect. 3, an online variational approach is developed for the learning of the proposed model. Section 4 outlines the experimental setup involving the challenging problem of images categorization and presents the obtained results. The paper is concluded in Sect. 5.

2 Hierarchical DP Mixture of Dirichlet Distributions

In this section, we start by briefly reviewing Dirichlet processes and then we present in details our hierarchical model.

2.1 Dirichlet Process

The DP is a stochastic process whose sample paths are probability measures with probability one [16, 21]. Given a random distribution G , it is distributed according to a DP if its marginals follow Dirichlet distributions. More specifically, let H be a distribution over some probability space Θ and γ be a positive real number, then G is a DP with the base distribution H and concentration parameter γ , denoted as $G \sim \text{DP}(\gamma, H)$, if

$$(G(A_1), \dots, G(A_t)) \sim \text{Dir}(\gamma H(A_1), \dots, \gamma H(A_t)) \quad (1)$$

where (A_1, \dots, A_t) is the set of the finite partitions of Θ , and $\text{Dir}(\gamma H(A_1), \dots, \gamma H(A_t))$ is a finite-dimensional Dirichlet distribution with parameters $(\gamma H(A_1), \dots, \gamma H(A_t))$.

2.2 Hierarchical DP Mixture Model of Dirichlet Distributions

Hierarchical Dirichlet Process. A hierarchical DP is a distribution over a set of random probability measures over a probability space Θ . Recently, it has been shown to be an effective framework for modeling grouped data where observations are organized into groups that are allowed to remain statistically linked [30, 32]. Assuming that we have a data set which is separated into M groups. A hierarchical DP involves an indexed set of DPs $\{G_j\}$, one of each group, that share a base distribution G_0 , which is itself distributed as a DP:

$$G_0 \sim \text{DP}(\gamma, H) \quad G_j \sim \text{DP}(\lambda, G_0) \quad \text{for each } j, j \in \{1, \dots, M\} \quad (2)$$

where j is an index for each group of data. A hierarchical Dirichlet process can be represented in a more intuitive and straightforward way using two stick-breaking constructions [18, 29] containing a base-level and a group-level construction. In the base-level construction, since the base distribution G_0 is distributed according to the Dirichlet process $\text{DP}(\gamma, H)$, it can be expressed using a stick-breaking representation as

$$\beta'_k \sim \text{Beta}(1, \gamma) \quad \alpha_k \sim H \quad \beta_k = \beta'_k \prod_{s=1}^{k-1} (1 - \beta'_s) \quad G_0 = \sum_{k=1}^{\infty} \beta_k \delta_{\alpha_k} \quad (3)$$

where $\{\alpha_k\}$ are independent random variables distributed according to H , and where δ_{α_k} is an atom at α_k . The variables $\{\beta_k\}$ are known as the stick-breaking weights that satisfy $\sum_{k=1}^{\infty} \beta_k = 1$, and are obtained by recursively breaking a unit length stick into an infinite number of pieces such that the size of each successive piece is proportional to the rest of the stick. It is noteworthy that since G_0 is discrete and has a stick-breaking representation as in Eq. (3) according to the property of DP, the atoms α_k are shared among all G_j and differ only in weights. In this work, we apply the stick-breaking representation [34] to construct each group-level DP G_j :

$$\pi'_{jt} \sim \text{Beta}(1, \lambda) \quad \varpi_{jt} \sim G_0 \quad \pi_{jt} = \pi'_{jt} \prod_{s=1}^{t-1} (1 - \pi'_{js}) \quad G_j = \sum_{t=1}^{\infty} \pi_{jt} \delta_{\varpi_{jt}} \quad (4)$$

where $\delta_{\varpi_{jt}}$ is a group-level atom at ϖ_{jt} , and where $\{\pi_{jt}\}$ are the stick-breaking weights which satisfy $\sum_{t=1}^{\infty} \pi_{jt} = 1$. Since ϖ_{jt} is distributed according to the base distribution G_0 , it takes on the value α_k with probability β_k . We may also represent this using a binary latent variable $\mathbf{C}_{jt} = (C_{jt1}, C_{jt2}, \dots)$ as an indicator variable, such that $C_{jtk} \in \{0, 1\}$, $C_{jtk} = 1$ if ϖ_{jt} maps to the base-level atom α_k which is indexed by k ; otherwise, $C_{jtk} = 0$. Accordingly, we have $\varpi_{jt} = \alpha_k^{C_{jtk}}$. Consequently, group-level atoms ϖ_{jt} do not need to be explicitly represented

which further simplifies the inference process as it shall be clearer in the next section. The indicator variable \mathbf{C}_{jt} is distributed according to $\boldsymbol{\beta}$:

$$p(\mathbf{C}|\boldsymbol{\beta}) = \prod_{j=1}^M \prod_{t=1}^{\infty} \prod_{k=1}^{\infty} \beta_k^{C_{jtk}} \quad (5)$$

Since $\boldsymbol{\beta}$ is a function of $\boldsymbol{\beta}'$ according to the stick-breaking construction of the Dirichlet process as shown in Eq. (3), $p(\mathbf{C})$ can then be represented in the following form

$$p(\mathbf{C}|\boldsymbol{\beta}') = \prod_{j=1}^M \prod_{t=1}^{\infty} \prod_{k=1}^{\infty} [\beta'_k \prod_{s=1}^{k-1} (1 - \beta'_s)]^{C_{jtk}} \quad (6)$$

The prior of $\boldsymbol{\beta}'$ is a Beta distribution according to Eq. (3):

$$p(\boldsymbol{\beta}') = \prod_{k=1}^{\infty} \text{Beta}(1, \gamma_k) = \prod_{k=1}^{\infty} \gamma_k (1 - \beta'_k)^{\gamma_k - 1} \quad (7)$$

One significant application of hierarchical DP is its consideration as a non-parametric prior over the factors for grouped data. More specifically, let i indexes the observations within each group j , we assume that each variable θ_{ji} is a factor corresponding to an observation X_{ji} , and the factors $\boldsymbol{\theta}_j = (\theta_{j1}, \theta_{j2}, \dots)$ are distributed according to G_j , for each j . Thus, we can have the likelihood in the following form

$$\theta_{ji}|G_j \sim G_j \quad X_{ji}|\theta_{ji} \sim F(\theta_{ji}) \quad (8)$$

where $F(\theta_{ji})$ denotes the distribution of the observation X_{ji} given θ_{ji} , the prior for the factors θ_{ji} is the base distribution H of G_0 . This setting forms the definition of a *hierarchical DP mixture model*, where each group is associated with a mixture component, and the components are shared among these mixture models due to the sharing of atoms α_k among all G_j . Moreover, since each factor θ_{ji} is distributed according to G_j , it takes the value ϖ_{jt} with probability π_{jt} . Next, we introduce a binary latent variable $\mathbf{Z}_{ji} = (Z_{ji1}, Z_{ji2}, \dots)$ as an indicator variable. That is, $Z_{jit} \in \{0, 1\}$, we have $Z_{jit} = 1$ if θ_{ji} is associated with component t and maps to the group-level atom ϖ_{jt} ; otherwise, $Z_{jit} = 0$. Thus, we have $\theta_{ji} = \varpi_{jt}^{Z_{jit}}$. Since ϖ_{jt} also maps to the base-level atom α_k , we then have $\theta_{ji} = \varpi_{jt}^{Z_{jit}} = \alpha_k^{C_{jtk}Z_{jit}}$. The indicator variable \mathbf{Z}_{ji} is distributed according to $\boldsymbol{\pi}$ as

$$p(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{j=1}^M \prod_{i=1}^N \prod_{t=1}^{\infty} \pi_{jt}^{Z_{jit}} \quad (9)$$

According to the stick-breaking construction of the Dirichlet process in Eq. (4), $\boldsymbol{\pi}$ is a function of $\boldsymbol{\pi}'$. Then, we have

$$p(\mathbf{Z}|\boldsymbol{\pi}') = \prod_{j=1}^M \prod_{i=1}^N \prod_{t=1}^{\infty} [\pi'_{jt} \prod_{s=1}^{t-1} (1 - \pi'_{js})]^{Z_{jit}} \quad (10)$$

As shown in Eq. (4), the prior distribution of $\boldsymbol{\pi}'$ is a Beta:

$$p(\boldsymbol{\pi}') = \prod_{j=1}^M \prod_{t=1}^{\infty} \text{Beta}(1, \lambda_{jt}) = \prod_{j=1}^M \prod_{t=1}^{\infty} \lambda_{jt} (1 - \pi'_{jt})^{\lambda_{jt} - 1} \quad (11)$$

The Hierarchical Infinite Dirichlet Mixture Model. We focus on a specific form of hierarchical DP mixture model where each observation within a group is drawn from a mixture of Dirichlet distributions. Since DP mixture models are often considered as infinite mixture models, we refer to the proposed model as the hierarchical infinite Dirichlet mixture model. The consideration of Dirichlet mixtures is motivated by their superior performance in modeling proportional data (i.e. normalized histograms) that are naturally generated by many applications [3, 6, 14]. Although the Dirichlet distribution is a multivariate distribution which is often used as a conjugate prior to the multinomial distribution in Bayesian statistics, it will be considered as parent distribution to model the data directly in this work. Furthermore, since we adopt the hierarchical DP mixture model framework, the problem of determining the number of mixture components is avoided by assuming that there is a countably infinite number of components.

Now let us consider a data set \mathcal{X} containing N random vectors and separated into M groups. We suppose that each vector $\mathbf{X}_{ji} = (X_{ji1}, \dots, X_{jiD})$ is represented in a D -dimensional space and is drawn from a hierarchical infinite Dirichlet mixture model. Then, the corresponding likelihood function of the proposed model with latent variables can be written as

$$\begin{aligned} p(\mathcal{X}|\mathbf{Z}, \mathbf{C}, \boldsymbol{\alpha}) &= \prod_{j=1}^M \prod_{i=1}^N \prod_{t=1}^{\infty} \prod_{k=1}^{\infty} \text{Dir}(X_{jit}|\boldsymbol{\alpha}_k)^{Z_{jit}C_{jtk}} \\ &= \prod_{j=1}^M \prod_{i=1}^N \prod_{t=1}^{\infty} \prod_{k=1}^{\infty} \left[\frac{\Gamma(\sum_{l=1}^D \alpha_{kl})}{\prod_{l=1}^D \Gamma(\alpha_{kl})} \prod_{l=1}^D X_{jil}^{\alpha_{kl}-1} \right]^{Z_{jit}C_{jtk}} \end{aligned} \quad (12)$$

Next, we need to place a prior distribution over the parameter $\boldsymbol{\alpha}$. In our case, conjugate prior is preferred since it greatly simplifies the mathematics in the learning process. Since $\boldsymbol{\alpha}$ is positive and the formal conjugate prior for the Dirichlet distribution is intractable, a Gamma distribution $\mathcal{G}(\cdot)$ is adopted to approximate the conjugate prior with an assumption that the Dirichlet parameters are statistically independent [14]:

$$p(\boldsymbol{\alpha}) = \mathcal{G}(\boldsymbol{\alpha}|\mathbf{u}, \mathbf{v}) = \prod_{k=1}^{\infty} \prod_{l=1}^D \frac{v_{kl}^{u_{kl}}}{\Gamma(u_{kl})} \alpha_{kl}^{u_{kl}-1} e^{-v_{kl}\alpha_{kl}} \quad (13)$$

where \mathbf{u} and \mathbf{v} are positive hyperparameters.

3 Online Variational Model Learning

First, we propose a batch variational inference method for learning the proposed hierarchical infinite Dirichlet mixture model based on a natural gradient method. Then, an online extension is proposed to account for large-scale or streaming data. The consideration of Variational inference [1] is motivated by the excellent results that it has provided when applied to finite Dirichlet mixtures [14]. In order to simplify notations, in this section, we define $\Omega = (\mathbf{Z}, \Lambda)$ as the set of latent and unknown random variables where $\Lambda = (\mathbf{C}, \boldsymbol{\pi}', \boldsymbol{\beta}', \boldsymbol{\alpha})$.

3.1 Batch Variational Inference

The goal of variational inference is to find an appropriate approximation, in terms of Kullback-Leibler (KL) divergence, $q(\Omega)$ for the true posterior distribution $p(\Omega|\mathcal{X})$. This problem can be tackled by adopting a factorization assumption for restricting the form of $q(\Omega)$ which is known as *mean field theory* [1]. Moreover, we adopt a truncation technique proposed in [2] to truncate the variational approximations of base and group levels at K and T , such that

$$\beta'_K = 1, \quad \sum_{k=1}^K \beta_k = 1, \quad \beta_k = 0 \text{ when } k > K \quad (14)$$

$$\pi'_{jT} = 1, \quad \sum_{t=1}^T \pi_{jt} = 1, \quad \pi_{jt} = 0 \text{ when } t > T \quad (15)$$

Notice that the truncation levels K and T are variational parameters which can be freely initialized and will be optimized automatically during the learning process. By adopting the truncated stick-breaking representation and the factorization assumption, the approximated posterior distribution $q(\Omega)$ can be fully factorized into disjoint distributions as

$$q(\Omega) = q(\mathbf{Z})q(\mathbf{C})q(\boldsymbol{\pi}')q(\boldsymbol{\beta}')q(\boldsymbol{\alpha}) \quad (16)$$

The approach that we consider for deriving our optimization solutions is based on a gradient method [28] and that can be easily extended to online settings as we shall see in the next section. The idea of the gradient-based variational inference approach is that, since the model has conjugate priors, the functional form of the factors in the variational posterior distribution is known. Thus, the lower bound $\mathcal{L}(q)$ can be considered as a function of the parameters of these distributions by taking their general parametric forms. The optimization of variational factors is then obtained by maximizing the lower bound with respect to these parameters. In our case, the functional form for each variational factor is the same as its conjugate prior distribution, namely Discrete for \mathbf{Z} and \mathbf{C} , Beta for $\boldsymbol{\beta}'$ and $\boldsymbol{\pi}'$, and Gamma for $\boldsymbol{\alpha}$. Therefore, the parametric forms for these variational posterior distributions can be defined as the following

$$q(\mathbf{Z}) = \prod_{j=1}^M \prod_{i=1}^N \prod_{t=1}^T \rho_{jit}^{Z_{jit}} \quad q(\mathbf{C}) = \prod_{j=1}^M \prod_{t=1}^T \prod_{k=1}^K \vartheta_{jtk}^{C_{jtk}} \quad (17)$$

$$q(\boldsymbol{\pi}') = \prod_{j=1}^M \prod_{t=1}^T \text{Beta}(\pi'_{jt}|a_{jt}, b_{jt}) \quad q(\boldsymbol{\beta}') = \prod_{k=1}^K \text{Beta}(\beta'_k|g_k, h_k) \quad (18)$$

$$q(\boldsymbol{\alpha}) = \prod_{k=1}^K \prod_{l=1}^D \mathcal{G}(\alpha_{kl}|u_{kl}^*, v_{kl}^*) \quad (19)$$

By Maximizing the lower bound $\mathcal{L}(q)$, we obtain $\rho_{jit} = \frac{\exp(\tilde{\rho}_{jit})}{\sum_{f=1}^T \exp(\tilde{\rho}_{jif})}$, where

$$\tilde{\rho}_{jit} = \sum_{k=1}^K \langle C_{jtk} \rangle [\tilde{\mathcal{R}}_k + \sum_{l=1}^D (\bar{\alpha}_{kl} - 1) \ln X_{jil}] + \langle \ln \pi'_{jt} \rangle + \sum_{s=1}^{t-1} \langle \ln(1 - \pi'_{js}) \rangle \quad (20)$$

$$\tilde{\mathcal{R}}_k = \ln \frac{\Gamma(\sum_{l=1}^D \bar{\alpha}_{kl})}{\prod_{l=1}^D \Gamma(\bar{\alpha}_{kl})} + \sum_{l=1}^D \bar{\alpha}_{kl} [\Psi(\sum_{l=1}^D \bar{\alpha}_{kl}) - \Psi(\bar{\alpha}_{kl})] [\langle \ln \alpha_{kl} \rangle - \ln \bar{\alpha}_{kl}] \quad (21)$$

$$+ \frac{1}{2} \sum_{l=1}^D \bar{\alpha}_{kl}^2 [\Psi'(\sum_{l=1}^D \bar{\alpha}_{kl}) - \Psi'(\bar{\alpha}_{kl})] \langle (\ln \alpha_{kl} - \ln \bar{\alpha}_{kl})^2 \rangle$$

$$+ \frac{1}{2} \sum_{c=1}^D \sum_{\substack{d=1 \\ (d \neq c)}}^D \alpha_{kc} \alpha_{kd} \left[\Psi'(\sum_{l=1}^D \bar{\alpha}_{kl}) (\langle \ln \alpha_{kc} \rangle - \ln \bar{\alpha}_{kc}) (\langle \ln \alpha_{kd} \rangle - \ln \bar{\alpha}_{kd}) \right]$$

$$\vartheta_{jtk} = \frac{\exp(\tilde{\vartheta}_{jtk})}{\sum_{f=1}^K \exp(\tilde{\vartheta}_{jtf})} \quad (22)$$

$$\tilde{\vartheta}_{jtk} = \sum_{i=1}^N \langle Z_{jit} \rangle [\tilde{\mathcal{R}}_k + \sum_{l=1}^D (\bar{\alpha}_{kl} - 1) \ln X_{jil}] + \langle \ln \beta'_k \rangle + \sum_{s=1}^{k-1} \langle \ln(1 - \beta'_s) \rangle \quad (23)$$

$$a_{jt} = 1 + \sum_{i=1}^N \langle Z_{jit} \rangle, \quad b_{jt} = \lambda_{jt} + \sum_{i=1}^N \sum_{s=t+1}^T \langle Z_{jis} \rangle \quad (24)$$

$$g_k = 1 + \sum_{j=1}^K \sum_{t=1}^T \langle C_{jtk} \rangle, \quad h_k = \gamma_k + \sum_{j=1}^M \sum_{t=1}^T \sum_{m=k+1}^K \langle C_{jtm} \rangle \quad (25)$$

$$u_{kl}^* = u_{kl} + \sum_{j=1}^M \sum_{t=1}^T \langle C_{jtk} \rangle \sum_{i=1}^N \langle Z_{jit} \rangle \bar{\alpha}_{kl} [\Psi(\sum_{s=1}^D \bar{\alpha}_{ks}) - \Psi(\bar{\alpha}_{kl})]$$

$$+ \sum_{s \neq l}^D \bar{\alpha}_{ks} \Psi'(\sum_{s=1}^D \bar{\alpha}_{ks}) (\langle \ln \alpha_{ks} \rangle - \ln \bar{\alpha}_{ks}) \quad (26)$$

$$v_{kl}^* = v_{kl} - \sum_{j=1}^M \sum_{t=1}^T \langle C_{jtk} \rangle \sum_{i=1}^N \langle Z_{jit} \rangle \ln X_{jil} \quad (27)$$

where $\Psi(\cdot)$ is the digamma function. The expected values in the above formulas are defined as

$$\bar{\alpha}_{kl} = \frac{u_{kl}^*}{v_{kl}^*} \quad \langle Z_{jit} \rangle = \rho_{jit} \quad \langle C_{jtk} \rangle = \vartheta_{jtk} \quad \langle \ln \alpha_{kl} \rangle = \Psi(u_{kl}^*) - \ln v_{kl}^* \quad (28)$$

$$\langle \ln \pi'_{jt} \rangle = \Psi(a_{jt}) - \Psi(a_{jt} + b_{jt}) \quad \langle \ln(1 - \pi'_{jt}) \rangle = \Psi(b_{jt}) - \Psi(a_{jt} + b_{jt}) \quad (29)$$

$$\langle \ln \beta'_k \rangle = \Psi(g_k) - \Psi(g_k + h_k) \quad \langle \ln(1 - \beta'_k) \rangle = \Psi(h_k) - \Psi(g_k + h_k) \quad (30)$$

$$\langle (\ln \alpha_{kl} - \ln \bar{\alpha}_{kl})^2 \rangle = [\Psi(u_{kl}^*) - \ln v_{kl}^*]^2 + \Psi'(u_{kl}^*) \quad (31)$$

The batch variational inference for hierarchical infinite Dirichlet mixture model can be considered as an EM-like algorithm and is summarized in Algorithm 1.

Algorithm 1. Batch variational learning.

- 1: Choose the initial truncation levels K and T .
 - 2: Initialize the values for hyperparameters λ_{jt} , γ_k , u_{kl} and v_{kl} .
 - 3: Initialize the value of ρ_{jit} by K -Means algorithm.
 - 4: **repeat**
 - 5: *The variational E-step:*
 - 6: Estimate the expected values in Eqs. (28)–(31), use the current distributions over the model parameters.
 - 7: *The variational M-step:*
 - 8: Update the variational solutions for each factor using Eqs. (17)–(19) and the current values of the moments.
 - 9: **until** Convergence.
-

3.2 Online Variational Inference

Inspired from the online learning framework proposed in [28] and tested successfully in [34], we develop an online variational inference framework for learning our model. In contrast with batch learning algorithms, online algorithms are more efficient when dealing with large-scale or streaming data which are naturally present in many real-world applications. In our case, let r denote the amount of observed data that we currently have. Then, the current lower bound for the observed data can be calculated by

$$\mathcal{L}^{(r)}(q) = \frac{N}{r} \sum_{i=1}^r \int q(\Lambda) d\Lambda \sum_{\mathbf{Z}_i} Q(\mathbf{Z}_i) \ln \left[\frac{p(\mathbf{X}_i, \mathbf{Z}_i | \Lambda)}{q(\mathbf{Z}_i)} \right] + \int q(\Lambda) \ln \left[\frac{p(\Lambda)}{q(\Lambda)} \right] d\Lambda \quad (32)$$

where $\Lambda = (\mathbf{C}, \boldsymbol{\pi}', \boldsymbol{\beta}', \boldsymbol{\alpha})$. The main idea of the online variational inference is to successively maximize the current variational lower bound as in Eq. (32) with respect to each variational factor. Consider that we have already observed a data set $\{\mathbf{X}_1, \dots, \mathbf{X}_{(r-1)}\}$. Then, after obtaining a new observation X_r , we can maximize the current lower bound $\mathcal{L}^{(r)}(q)$ with respect to $q(\mathbf{Z}_r)$, while other variational factors remain fixed to $q^{(r-1)}(\mathbf{C})$, $q^{(r-1)}(\boldsymbol{\alpha})$, $q^{(r-1)}(\boldsymbol{\pi}')$ and $q^{(r-1)}(\boldsymbol{\beta}')$. Therefore, we can update the variational solution to $q(\mathbf{Z}_r)$ as

$$q(\mathbf{Z}_r) = \prod_{j=1}^M \prod_{t=1}^T \rho_{jtr}^{Z_{jtr}} \quad (33)$$

where $\rho_{jtr} = \frac{\exp(\tilde{\rho}_{jtr})}{\sum_{j'=1}^T \exp(\tilde{\rho}_{j'tr})}$, and $\tilde{\rho}_{jtr} = \sum_{k=1}^K \langle C_{jtk}^{(r-1)} \rangle [\tilde{\mathcal{R}}_k^{(r-1)} + \sum_{l=1}^D (\bar{\alpha}_{kl}^{(r-1)} - 1) \ln X_{jrl}] + \langle \ln \pi_{jt}^{(r-1)} \rangle + \sum_{s=1}^{t-1} \langle \ln(1 - \pi_{js}^{(r-1)}) \rangle$. In the following step, we maximize the current lower bound $\mathcal{L}^{(r)}(q)$ with respect to $q^{(r)}(\mathbf{C})$, while $q(\mathbf{Z}_r)$ is fixed and other variational factors remain at their $(r-1)$ th values. Thus, the variational factor $q^{(r)}(\mathbf{C})$ can be updated as

$$q^{(r)}(\mathbf{C}) = \prod_{j=1}^M \prod_{t=1}^T \prod_{k=1}^K (\vartheta_{jtk}^{(r)})^{C_{jtk}^{(r)}} \quad (34)$$

where the hyperparameter $\vartheta_{jtk}^{(r)}$ is defined by

$$\vartheta_{jtk}^{(r)} = \vartheta_{jtk}^{(r-1)} + \xi_r \Delta \tilde{\vartheta}_{jtk}^{(r)} \quad (35)$$

where ξ_r is the learning rate. In this work, we adopt a learning rate function introduced in [34], such that $\xi_r = (\eta_0 + r)^{-w}$, subject to the constraints $w \in (0.5, 1]$ and $\eta_0 \geq 0$. In Eq. (35), $\Delta \tilde{\vartheta}_{jtk}^{(r)}$ is the natural gradient of the hyperparameter $\tilde{\vartheta}_{jtk}^{(r)}$. The natural gradient of a hyperparameter is obtained by multiplying the gradient by the inverse of Riemannian metric, which cancels the coefficient matrix for the posterior parameter distribution. Thus, we can obtain the natural gradient $\Delta \tilde{\vartheta}_{jtk}^{(r)}$ as

$$\Delta \tilde{\vartheta}_{jtk}^{(r)} = \tilde{\vartheta}_{jtk}^{(r)} - \vartheta_{jtk}^{(r-1)} = \frac{\exp(\tilde{\vartheta}_{jtk}^{(r)})}{\sum_{f=1}^K \exp(\tilde{\vartheta}_{jtf}^{(r)})} - \vartheta_{jtk}^{(r-1)} \quad (36)$$

$$\tilde{\vartheta}_{jtk}^{(r)} = N \rho_{jtr} [\tilde{\mathcal{R}}_k^{(r-1)} + \sum_{l=1}^D (\bar{\alpha}_{kl}^{(r-1)} - 1) \ln X_{jrl}] + \langle \ln \beta_k^{(r-1)} \rangle + \sum_{s=1}^{k-1} \langle \ln(1 - \beta_s^{(r-1)}) \rangle \quad (37)$$

Next, the current lower bound $\mathcal{L}^{(r)}(q)$ is maximized with respect to $q^{(r)}(\boldsymbol{\pi}')$, $q^{(r)}(\boldsymbol{\beta}')$ and $q^{(r)}(\boldsymbol{\alpha})$:

$$q^{(r)}(\boldsymbol{\pi}') = \prod_{j=1}^M \prod_{t=1}^T \text{Beta}(\pi_{jt}^{(r)} | a_{jt}^{(r)}, b_{jt}^{(r)}) \quad (38)$$

$$q^{(r)}(\boldsymbol{\beta}') = \prod_{k=1}^K \text{Beta}(\beta_k^{(r)} | g_k^{(r)}, h_k^{(r)}) \quad q^{(r)}(\boldsymbol{\alpha}) = \prod_{k=1}^K \prod_{l=1}^D \mathcal{G}(\alpha_{kl}^{(r)} | u_{kl}^{*(t)}, v_{kl}^{*(t)}) \quad (39)$$

where the hyperparameters are given by

$$a_{jt}^{(r)} = a_{jt}^{(r-1)} + \xi_r \Delta a_{jt}^{(r)}, \quad b_{jt}^{(r)} = b_{jt}^{(r-1)} + \xi_r \Delta b_{jt}^{(r)} \quad (40)$$

$$g_k^{(r)} = g_k^{(r-1)} + \xi_r \Delta g_k^{(r)}, \quad h_k^{(r)} = h_k^{(r-1)} + \xi_r \Delta h_k^{(r)} \quad (41)$$

$$u_{kl}^{*(r)} = u_{kl}^{*(r-1)} + \xi_r \Delta u_{kl}^{*(r)}, \quad v_{kl}^{*(r)} = v_{kl}^{*(r-1)} + \xi_r \Delta v_{kl}^{*(r)} \quad (42)$$

The corresponding natural gradients can be calculated as

$$\Delta a_{jt}^{(r)} = 1 + N \rho_{jtr} - a_{jt}^{(r-1)} \quad \Delta b_{jt}^{(r)} = \lambda_{jt} + N \sum_{s=t+1}^T \rho_{jsr} - b_{jt}^{(r-1)} \quad (43)$$

$$\Delta g_k^{(r)} = 1 + \sum_{j=1}^K \sum_{t=1}^T \vartheta_{jtk}^{(r)} - g_k^{(r-1)} \quad \Delta h_k^{(r)} = \gamma_k + \sum_{j=1}^M \sum_{t=1}^T \sum_{m=k+1}^K \vartheta_{jtk}^{(r)} - h_k^{(r-1)} \quad (44)$$

$$\begin{aligned} \Delta u_{kl}^{*(t)} &= u_{kl} + N \sum_{j=1}^M \sum_{t=1}^T \vartheta_{jtk}^{(r)} \rho_{jtr} \bar{\alpha}_{kl}^{(r-1)} [\Psi(\sum_{s=1}^D \bar{\alpha}_{ks}^{(r-1)}) - \Psi(\bar{\alpha}_{kl}^{(r-1)})] \\ &+ \sum_{s \neq l}^D \bar{\alpha}_{ks}^{(r-1)} \Psi'(\sum_{s=1}^D \bar{\alpha}_{ks}^{(r-1)}) (\langle \ln \alpha_{ks}^{(r-1)} \rangle - \ln \bar{\alpha}_{ks}^{(r-1)}) - u_{kl}^{*(t-1)} \end{aligned} \quad (45)$$

$$\Delta v_{kl}^{(r)} = v_{kl} - N \sum_{j=1}^M \sum_{t=1}^T \vartheta_{jtk}^{(r)} \rho_{jtr} \ln X_{jrl} - v_{kl}^{(r-1)} \quad (46)$$

It is noteworthy that the hyperparameters of $q^{(r)}(\boldsymbol{\pi}')$, $q^{(r)}(\boldsymbol{\beta}')$ and $q^{(r)}(\boldsymbol{\alpha})$ can be updated in parallel. This online variational inference procedure is repeated until all the variational factors are updated with respect to the current arrived observation. The online variational inference for hierarchical infinite Dirichlet mixture model is summarized in Algorithm 2. The proposed online learning algorithm is much more computationally efficient than its batch counterpart. This is because the batch algorithm updates the variational factors by using the whole data set in each iteration, and thus its estimation quality is improved more slowly than in the case of the online one.

Algorithm 2. Online variational learning.

- 1: Choose the initial truncation levels K and T .
 - 2: Initialize the values for hyperparameters λ_{jt} , γ_k , u_{kl} and v_{kl} .
 - 3: **for** $r = 1 \rightarrow N$ **do**
 - 4: *The variational E-step:*
 - 5: Update the variational solution to $q(\mathbf{Z}_r)$ using Eq. (33).
 - 6: *The variational M-step:*
 - 7: Compute learning rate $\xi_r = (\eta_0 + r)^{-w}$.
 - 8: Calculate the natural gradient $\Delta \vartheta_{jtk}^{(r)}$ using Eq. (36).
 - 9: Update the variational factor $q^{(r)}(\mathbf{C})$ as shown in Eq. (34).
 - 10: Calculate the natural gradients of the remaining hyperparameters using Eqs. (43)–(46).
 - 11: Update variational factors $q^{(r)}(\boldsymbol{\pi}')$, $q^{(r)}(\boldsymbol{\beta}')$ and $q^{(r)}(\boldsymbol{\alpha})$ through Eqs. (38)–(39).
 - 12: Repeat the *E-* and *M-steps* until new data are observed.
 - 13: **end for**
-

4 Experimental Results: Online Images Categorization

4.1 Experimental Design

In this section, we evaluate the effectiveness of the proposed online hierarchical infinite Dirichlet mixture (referred to as *OnHIDM*) model through a challenging real-world application namely online images categorization. The tackled problem is a fundamental task in computer vision and has drawn significant attention during the last decade [12, 15, 17, 35]. This problem, however, remains challenging due to the difficulty of capturing the variability of appearance and shape of diverse objects belonging to the same class, while avoiding confusing objects from different classes [23]. In our experiments, we demonstrate the advantages of our *OnHIDM* model by comparing its performance with three other mixture models involving the batch hierarchical infinite Dirichlet mixture

(*BaHIDM*) model, the online hierarchical infinite Gaussian mixture (*OnHIGM*) model and the online finite Dirichlet mixture (*OnFDM*) model. To make a fair comparison, all of these models are learned using variational inference. It is noteworthy that our goals are mainly to demonstrate the advantages of using online variational inference learning framework over the batch one, and using hierarchical infinite mixture model over the finite one, as well as using Dirichlet over the Gaussian mixture. In our experiments, the testing data are supposed to arrive sequentially in an online manner except for the *BaHIDM* model. We initialize the base truncation level K to 50, and the group truncation level T to 15. The parameters w and η_0 of the learning rate are set to 0.65 and 64, respectively. The hyperparameters involved in our model are initialized as $(\lambda_{jt}, \gamma_k, u_{kl}, v_{kl}) = (0.05, 0.05, 0.1, 0.01)$. Our simulations have supported these specific choices.

4.2 Methodology and Results

We apply the proposed *OnHIDM* to the problem of online images clustering using the following methodology. First, 128-dimensional scale-invariant feature transform (SIFT) [22] descriptors¹ are extracted from each image using the Difference-of-Gaussians (DoG) interest point detectors and then normalized. Next, these features are modeled using the proposed approach. Specifically, each image \mathcal{I}_j is considered as a “group” and is therefore associated with a Dirichlet process mixture (infinite mixture) model G_j . Thus, each extracted SIFT feature vector X_{ji} from image \mathcal{I}_j is supposed to be drawn from an infinite mixture model G_j , in which mixture models can be viewed as a representation of “visual words”. A global vocabulary is constructed and is shared among all groups (images) through the introduction of the common global infinite mixture model G_0 . This setting matches the desired design of a hierarchical Dirichlet process mixture model. An important step in image categorization approaches with bag-of-visual words representation is the construction of a visual vocabulary. The majority of these approaches need to use a separate vector quantization algorithm (such as K -means) to build the visual dictionary, where the vocabulary size is normally manually selected. In our approach, the construction of the visual vocabulary is part of the hierarchical Dirichlet process mixture framework, and the size of the vocabulary (number of mixture components in the global level mixture model) can be automatically inferred from the data thanks to its Bayesian nonparametric nature. Since our goal is to determine automatically the category to which a testing image \mathcal{I}_j should be assigned, our hierarchical Dirichlet process mixture framework needs to be augmented by an indicator variable B_{jm} associated with each image (or group). B_{jm} means that \mathcal{I}_j is generated from category m and then is drawn from an other infinite mixture model which is truncated at level J . This means that we need to add a new hierarchy level to our hierarchical infinite mixture model with a sharing vocabulary among all image categories. In this

¹ Other state-of-the-art local visual descriptors may provide better results, however, this is not the focus of this work.

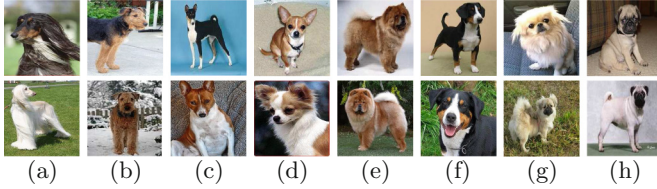


Fig. 1. Samples from the Dogs database. (a) Afghan hound, (b) Airedale, (c) Basenji, (d) Chihuahua, (e) Chow, (f) Entlebucher, (g) Pekinese, (h) Pug.

experiment, we truncate J to 20 and initialize the hyperparameter of the mixing probability of B_{jm} as 0.05. Finally, a testing image is affected to the category which has the highest posterior probability according to Bayes' decision rule.

Table 1. The average categorization accuracy rate (Acc) (%) obtained over 30 runs using different methods. The numbers in parenthesis are the standard deviation of the corresponding quantities.

Method	<i>OnHIDM</i>	<i>BaHIDM</i>	<i>OnFDM</i>	<i>OnHIGM</i>
Acc (%)	80.87 (1.19)	81.32 (1.02)	76.18 (1.54)	75.43 (1.31)

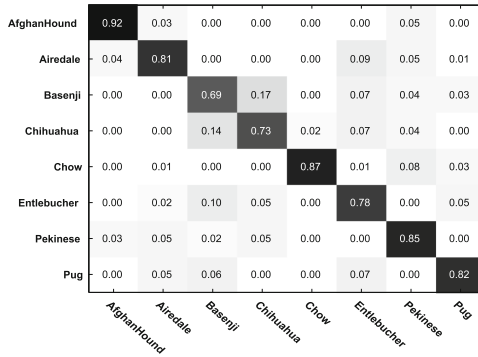


Fig. 2. Accuracy as a function of the number of images in the training set.

In our experiments, we consider a challenging public available database known as the Stanford Dogs database². This database contains 20,580 images of 120 breeds of dogs from around the world. The images are characterized by large scale, pose and light variations. The large intra-class and the small inter-class

² Database available at: <http://vision.stanford.edu/aditya86/ImageNetDogs>.

variabilities make this data set more challenging. In our experiments, we use a subset of this database consisting of 8 classes of dogs: Afghan hound (239 images), Airedale (202 images), Basenji (209 images), Chihuahua (152 images), Chow (196 images), Entlebucher (202 images), Pekinese (149 images) and Pug (200 images). Thus, we have 1,549 images in total. Sample images from each class are displayed in Fig. 1. We evaluated the categorization performance of the proposed algorithm by running it 30 times. We quantified the performance of our categorization approach using a confusion matrix as well as the rate of overall categorization accuracy. Each entry (i, j) of the confusion matrix denotes the percentage of images in category i that are assigned to category j . Figure 2 shows the confusion matrix computed by the proposed *OnHIDM* for our Dogs database. According to this matrix, the average categorization accuracy obtained by using *OnHIDM* was 80.87 % (error rate of 19.13 %). For comparison, we have also applied three other mixture-based approaches as mentioned earlier: *BaHIDM*, *OnHIGM* and *OnFDM*. The average performances of all tested approaches are given in Table 1. According to the results shown in this table, it is clear that the proposed *OnHIDM* and its batch counterpart (the *BaHIDM*) behave similarly (i.e., a Students t -test shows that the difference in performance between the *BaHIDM* and *OnHIDM* is not statistically significant: p -values between 0.1364 and 0.2237 for different runs) by providing better results than other two tested approaches. In this case, *OnHIDM* is a better choice over the *BaHIDM*, since *OnHIDM* is significantly faster, thanks to its online learning property, than the *BaHIDM*. According to our results, the *BaHIDM* required 2 h and 32 min to categorize all images while the *OnHIDM* only needed 47 min to do so on a computer with Intel’s Core i7 processor 2.00 GHz. Furthermore, the advantage of using a hierarchical infinite mixture model over a finite mixture model is clear by observing that better performance was obtained by *OnHIDM* (80.87 %) than by *OnFDM* (76.18 %) in terms of categorization accuracy rate. It is also worth mentioning that, as we can see from Table 1, the proposed *OnHIDM* (80.87 %) outperformed *OnHIGM* (75.43 %) which shows again the fact that the Dirichlet model has better modeling capability than the Gaussian for normalized data.

5 Conclusion

Nonparametric Bayesian models have been quite popular recently in many pattern recognition and computer vision problems due to their high accuracy and potential for data modeling. The success of these techniques rests largely on good choices of the distributions. This paper has presented and evaluated a hierarchical DP mixture model of Dirichlet distributions learned within a variational framework. The approach strives to achieve a high accuracy of online data clustering and has been validated through a challenging application namely images categorization. Further efficiency improvements are possible by performing several extensions such as introducing feature selection within the proposed model or considering Beta-Liouville distribution that has been shown to be a good alternative to the Dirichlet recently [5]. The consideration of the proposed model

with other learning approaches such as transfer learning [27] or its application to other challenging problems such as images annotation or objects recognition are interesting avenues for future research, also.

Acknowledgment. The completion of this research was made possible thanks to the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springle, New York (2006)
2. Blei, D.M., Jordan, M.I.: Variational inference for Dirichlet process mixtures. *Bayesian Anal.* **1**, 121–144 (2005)
3. Bouguila, N., Wang, J.H., Hamza, A.B.: Software modules categorization through likelihood and bayesian analysis of finite dirichlet mixtures. *J. Appl. Stat.* **37**(2), 235–252 (2010)
4. Bouguila, N., Ziou, D.: A dirichlet process mixture of dirichlet distributions for classification and prediction. In: Proceedings of the IEEE Workshop on Machine Learning for Signal Processing (MLSP), pp. 297–302. IEEE (2008)
5. Bouguila, N.: Infinite liouville mixture models with application to text and texture categorization. *Pattern Recogn. Lett.* **33**(2), 103–110 (2012)
6. Bouguila, N., Ziou, D.: Using unsupervised learning of a finite dirichlet mixture model to improve pattern recognition applications. *Pattern Recogn. Lett.* **26**(12), 1916–1925 (2005)
7. Bouguila, N., Ziou, D.: Online clustering via finite mixtures of dirichlet and minimum message length. *Eng. Appl. Artif. Intell.* **19**(4), 371–379 (2006)
8. Boyd-Graber, J.L., Blei, D.M.: Syntactic topic models. In: NIPS, pp. 185–192. Curran Associates, Inc. (2008)
9. Bradley, P.S., Fayyad, U., Reina, C.A.: Clustering very large databases using em mixture models. In: Proceedings of ICPR, vol. 2, pp. 76–80. IEEE (2000)
10. Carbonetto, P., Kisynski, J., de Freitas, N., Poole, D.: Nonparametric bayesian logic. In: Proceedings of UAI, pp. 85–93 (2005)
11. Caron, F., Davy, M., Doucet, A., Duflos, E., Vanheeghe, P.: Bayesian inference for linear dynamic models with dirichlet process mixtures. *IEEE Trans. Sign. Proces.* **56**(1), 71–84 (2008)
12. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision, ECCV, pp. 1–12. Springer (2004)
13. Doshi, F., Miller, K., Gael, J.V., Teh, Y.W.: Variational inference for the indian buffet process. *J. Mach. Learn. Res. Proc. Track* **5**, 137–144 (2009)
14. Fan, W., Bouguila, N., Ziou, D.: Variational learning for finite Dirichlet mixture models and applications. *IEEE Trans. Neural Netw. Learn. Syst.* **23**(5), 762–774 (2012)
15. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. *Comput. Vis. Image Underst.* **106**(1), 59–70 (2007)
16. Ferguson, T.S.: Bayesian density estimation by mixtures of normal distributions. *Recent Adv. Stat.* **24**, 287–302 (1983)

17. Frome, A., Singer, Y., Sha, F., Malik, J.: Learning globally-consistent local distance functions for shape-based image retrieval and classification. In: Proceedings of ICCV, pp. 1–8. IEEE (2007)
18. Ishwaran, H., James, L.F.: Gibbs sampling methods for stick-breaking priors. *J. Am. Stat. Assoc.* **96**, 161–173 (2001)
19. Jin, L.C., Wan, W.G., Cui, B., Yu, X.Q.: A new multimedia classification approach: Bayesian of inductive cognition algorithm based on dirichlet process. *Imaging Sci. J.* **58**(6), 331–339 (2010)
20. Jin, Y., Khan, L., Wang, L., Awad, M.: Image annotations by combining multiple evidence and wordnet. In: Proceedings of the 13th ACM International Conference on Multimedia, pp. 706–715 (2005)
21. Korwar, R.M., Hollander, M.: Contributions to the theory of Dirichlet processes. *Ann. Probab.* **1**, 705–711 (1973)
22. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**(2), 91–110 (2004)
23. Malisiewicz, T., Efros, A.A.: Improving spatial support for objects via multiple segmentations. In: Proceedings of BMVC, pp. 1–10 (2007)
24. Malisiewicz, T., Efros, A.A.: Recognition by association via learning per-exemplar distances. In: Proceedings of CVPR, pp. 1–8. IEEE (2008)
25. Nott, D.J.: Predictive performance of dirichlet process shrinkage methods in linear regression. *Comput. Stat. Data Anal.* **52**(7), 3658–3669 (2008)
26. Opper, M., Winther, O.: Gaussian processes for classification: mean-field algorithms. *Neural Comput.* **12**(11), 2655–2684 (2000)
27. Quattoni, A., Collins, M., Darrell, T.: Transfer learning for image classification with sparse prototype representations. In: Proceedings of CVPR, pp. 1–8. IEEE (2008)
28. Sato, M.: Online model selection based on the variational Bayes. *Neural Comput.* **13**, 1649–1681 (2001)
29. Sethuraman, J.: A constructive definition of Dirichlet priors. *Stat. Sin.* **4**, 639–650 (1994)
30. Teh, Y.W., Jordan, M.I.: Hierarchical Bayesian nonparametric models with applications. In: Hjort, N., Holmes, C., Müller, P., Walker, S. (eds.) *Bayesian Nonparametrics: Principles and Practice*, pp. 158–207. Cambridge University Press (2010)
31. Teh, Y.W., Görür, D., Ghahramani, Z.: Stick-breaking construction for the indian buffet process. *J. Mach. Learn. Res. Proc. Track* **2**, 556–563 (2007)
32. Teh, Y.W., Jordan, M.I., Beal, M.J., Blei, D.M.: Hierarchical Dirichlet processes. *J. Am. Stat. Assoc.* **101**(476), 1566–1581 (2006)
33. Volkmer, T., Smith, J.R., Natsev, A.: A web-based system for collaborative annotation of large image and video collections: an evaluation and user study. In: Proceedings of the 13th ACM International Conference on Multimedia, pp. 892–901 (2005)
34. Wang, C., Paisley, J.W., Blei, D.M.: Online variational inference for the hierarchical Dirichlet process. *J. Mach. Learn. Res. Proc. Track* **15**, 752–760 (2011)
35. Zhang, W., Yu, B., Zelinsky, G.J., Samaras, D.: Object class recognition using multiple layer boosting with heterogeneous features. In: Proceedings of the CVPR, pp. 323–330. IEEE (2005)

Distributed Skyline Computation of Vertically Splitted Databases by Using MapReduce

Md. Anisuzzaman Siddique^(✉), Hao Tian, and Yasuhiko Morimoto

Graduate School of Engineering, Hiroshima University, 1-7-1 Kagamiyama,
Higashi-Hiroshima 739-8521, Japan
{siddique,M124671}@hiroshima-u.ac.jp
morimoto@mis.hiroshima-u.ac.jp

Abstract. Skyline query retrieve objects that are not dominated by another object. A result of a skyline query is relatively small, does not contain less important objects, and is useful for selecting an object. In this paper, we consider a method for computing skyline query in MapReduce framework, which is a de facto standard in big data analysis. Currently, we have to be aware of data disclosure. Therefore, we propose a distributed computation method, in which each computer uses only a projected database that is vertically splitted from an original database, for computing skyline query. Since one computer can see only projected values, sensitive information in a database can be localized in the proposed method in addition to the advantage of the efficiency of MapReduce. Extensive experiments demonstrate the efficiency of proposed algorithm for synthetic datasets.

Keywords: Skyline query · MapReduce · Privacy · Sensitive database

1 Introduction

Recent computing infrastructure makes a large amount of information, many of which are stored in databases. In order to utilize the stored information, useful and efficient information retrieval methods are necessary.

In order to extract useful and relevant information from large information sources, the research community has invested considerable efforts into developing tools that facilitate the exploration of data. As a result, the skyline operator [1] and its variants such as dynamic skyline [10] and reverse skyline [11] operators have recently attracted considerable attention due to their broad applications including product or restaurant recommendations [12], review evaluations with user ratings [13], querying wireless sensor networks [14] and graph analysis [15].

A skyline query retrieves a set of skyline objects so that the user can choose objects from small number of noteworthy objects. Skyline objects in a dataset are objects that are not dominated by any other objects in the dataset. Given a m -attributes dataset DS , an object O_i is said to be in skyline of DS if there is no other object O_j ($i \neq j$) in DS such that O_j is better than O_i . If there exists

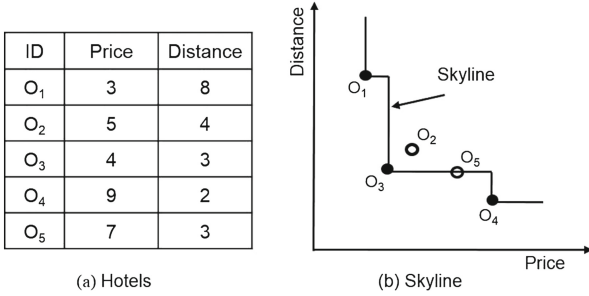


Fig. 1. Skyline example

such O_j , then we say that O_i is dominated by O_j or O_j dominates O_i . Figure 1 shows a typical example of skyline. The table in the figure is a list of hotels, each of which contains two numerical attributes: *distance* and *price*, for online booking. A user chooses a hotel from the list according to her/his preference. In this situation, her/his choice usually comes from the hotels in skyline, i.e., one of O_1, O_3, O_4 (see Fig. 1 (b)).

Motivating Example

Consider four datasets DS_1, DS_2, DS_3 , and DS_4 with two attributes a_1 and a_2 as shown in Table 1. To compute skyline on the union of these distributed datasets, one can follow straightforward approach in which she/he computes each local skyline and then compute global skyline from local skylines with any of existing conventional skyline algorithms. Assume that an user wants to choose a hotel from these distributed datasets. She/he usually chooses an object among the global skyline $\{O_{1,1}, O_{4,1}, O_{3,3}\}$. Since all other objects those are not in the global skyline are not better than one of the skyline objects. Here, the first suffix represents data source *ID* number and second one represents object *ID* in the corresponding data source. For example $O_{4,1}$ is an object of DS_4 and its *ID* in DS_4 is “1”. Conventional skyline algorithms are not suitable for such global skyline computation on distributed datasets. Most of existing skyline algorithms use single computer, which has limited scalability. In order to handle big data, some recent works utilize distributed computers, some of which use MapReduce framework.

Table 1. Database DS

Dataset1			Dataset2			Dataset3			Dataset4		
ID	a_1	a_2	ID	a_1	a_2	ID	a_1	a_2	ID	a_1	a_2
$O_{1,1}$	1	9	$O_{2,1}$	6	7	$O_{3,1}$	5	6	$O_{4,1}$	9	1
$O_{1,2}$	2	10	$O_{2,2}$	9	10	$O_{3,2}$	4	3	$O_{4,2}$	10	4
$O_{1,3}$	4	8	$O_{2,3}$	7	5	$O_{3,3}$	3	2	$O_{4,3}$	6	2
									$O_{4,4}$	8	3

In addition to the scalability, we have to be aware of privacy issues in a database. All existing distributed skyline algorithms do not pay much attentions on the privacy issue. It is always possible to pool local skyline result from all distributed datasets in one place and run global skyline computation. However, this is exactly what we do not want to do. Data privacy issues can arise in response to information from a wide range of sources. Some of them are: health-care records, criminal justice investigations and proceedings, financial institutions and transactions, residence and geographic records, etc. We address the problem how to compute skyline results without pooling the actual data. In this paper, we consider a distributed skyline algorithm on MapReduce framework. In our method, source databases are vertically splitted and are distributed. Unlike other MapReduce skyline algorithm, we can localize sensitive information in a database by the vertical splitting strategy and send projected values (in this paper we called “rank”) to compute final results.

Contributions of this paper include following three aspects:

- The skyline algorithm on distributed data using MapReduce framework is presented.
- The proposed distributed algorithm can localize sensitive information in a database.
- Extensive experiments are conducted to evaluate the efficiency and scalability of propose algorithm.

The rest of this paper is organized as follows: Sect. 2 reviews related work. Section 3 presents the notions and properties of skyline objects computation. We provide detailed examples and analysis of our algorithms in Sect. 4. We experimentally evaluate the proposed algorithms in Sect. 5 under a variety of settings. Finally, Sect. 6 concludes the paper.

2 Related Work

Our work is motivated by previous studies of skyline query processing as well as MapReduce based query processing. Those are is reviewed in the following sections.

2.1 Skyline Query Processing

Borzsonyi et al. first introduced the skyline operator over large databases and proposed three algorithms: *Block-Nested-Loops (BNL)*, *Divide-and-Conquer (D&C)*, and B-tree-based schemes [1]. BNL compares each object of the database with every other object, and reports it as a result only if any other object does not dominate it. A window W is allocated in main memory, and the input relation is sequentially scanned. In this way, a block of skyline objects is produced in every iteration. In case the window saturates, a temporary file is used to store objects that cannot be placed in W . This file is used as the input to the

next pass. *D&C* divides the dataset into several partitions such that each partition can fit into memory. Skyline objects for each individual partition are then computed by a main-memory skyline algorithm. The final skyline is obtained by merging the skyline objects for each partition. Chomicki et al. improved BNL by presorting, they proposed *Sort-Filter-Skyline (SFS)* as a variant of BNL [2]. Among index-based methods, Tan et al. proposed two progressive skyline computing methods Bitmap and Index [3]. In the Bitmap approach, every dimension value of a point is represented by a few bits. By applying bit-wise *AND* operation on these vectors, a given point can be checked if it is in the skyline without referring to other points. The index method organizes a set of d -dimensional objects into d lists such that an object O is assigned to list i if and only if its value at attribute i is the best among all attributes of O . Each list is indexed by a B-tree, and the skyline is computed by scanning the B-tree until an object that dominates the remaining entries in the B-trees is found. The current most efficient method is *Branch-and-Bound Skyline (BBS)*, proposed by Papadias et al., which is a progressive algorithm based on the *best-first nearest neighbor (BF-NN)* algorithm [4]. Instead of searching for nearest neighbor repeatedly, it directly prunes using the R*-tree structure.

Recently, more aspects of skyline computation have been explored. Chan et al. proposed k -dominant skyline and developed efficient ways to compute it in high-dimensional space [5]. Lin et al. proposed n -of- N skyline query to support online query on data streams, i.e., to find the skyline of the set composed of the most recent n elements. In the cases where the datasets are very large and stored distributedly, it is impossible to handle them in a centralized fashion [6]. Balke et al. first mined skyline in a distributed environment by partitioning the data vertically [7]. Vlachou et al. introduce the concept of extended skyline set, which contains all data elements that are necessary to answer a skyline query in any arbitrary subspace [8]. Tao et al. discuss skyline queries in arbitrary subspaces [9]. More skyline variants such as dynamic skyline [10] and reverse skyline [11] operators also have recently attracted considerable attention.

2.2 MapReduce Based Query Processing

Computing the skyline or its variants is challenging today since there is an increasing trend of applications expected to deal with *big data*. For such data intensive applications, the MapReduce [18–20] framework has recently attracted a lot of attention. MapReduce is a programming model that allows easy development of scalable parallel applications to process big data on large clusters of commodity machines. Ideally, a MapReduce system should achieve a high degree of load balancing among the participating machines, and minimize the space uses, CPU and I/O time, and network transfer at each machine. There exist some recent works on skyline computation using MapReduce [16, 17]. All of these works focus on efficient computation but did not give any idea how to preserve privacy.

In MapReduce framework, the implementation of Mappers and Reducers are completely independent of each other without communication among Mappers

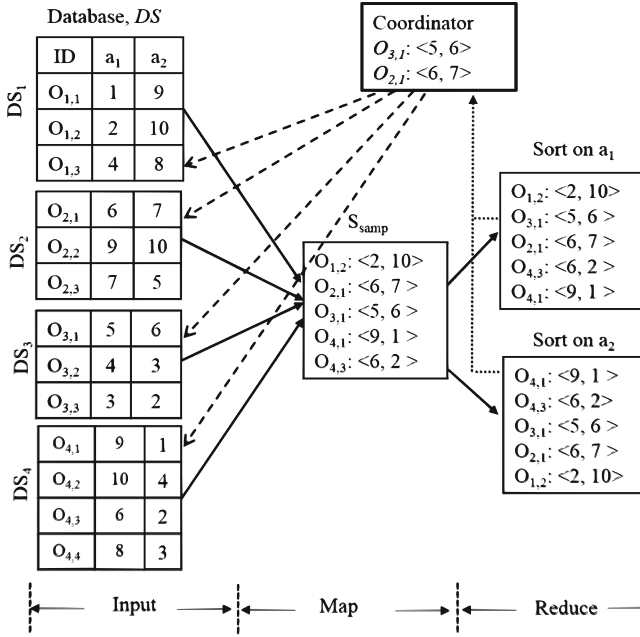


Fig. 2. TeraSort Implementation (1)

or Reducers. When processing this type of applications on MapReduce, there are large amount unpromising intermediate data to be transferred. These unpromising intermediate data will be finally abandoned anyhow, leading to the waste of disk access, network bandwidth, and CPU resources. To filter unpromising data and reduce the amount of intermediate data and the waste of time processing the unpromising data can be mitigated by applying skyline query search first.

3 Preliminaries

Given a database DS that is defined by a set of m -attributes $\{a_1, a_2, \dots, a_m\}$. The database is distributed into n datasets $\{DS_1, DS_2, \dots, DS_n\}$ on different locations. Without loss of generality, assume that each attribute has non-negative numerical values. We also assume smaller value is preferable in each attribute. We use $O_{i,j}.a_k$ to denote the k -th attribute's value of object $O_{i,j}$ where i represents datasets ID and j represent object ID in the corresponding dataset DS_i .

For objects $O_{i,j}$ and $O'_{i,j}$, an object $O_{i,j}$ is said to *dominate* another object $O'_{i,j}$ with respect to DS , denoted by $O_{i,j} \leq O'_{i,j}$, if $O_{i,j}.a_s \leq O'_{i,j}.a_s$ for all attributes ($s = 1, \dots, m$) and $O_{i,j}.a_x < O'_{i,j}.a_x$ for at least one attribute ($1 \leq x \leq m$). We call such $O_{i,j}$ as *dominant* object and such $O'_{i,j}$ as *dominated* object between $O_{i,j}$ and $O'_{i,j}$. If $O_{i,j}$ dominate $O'_{i,j}$, then $O_{i,j}$ is more preferable than $O'_{i,j}$.

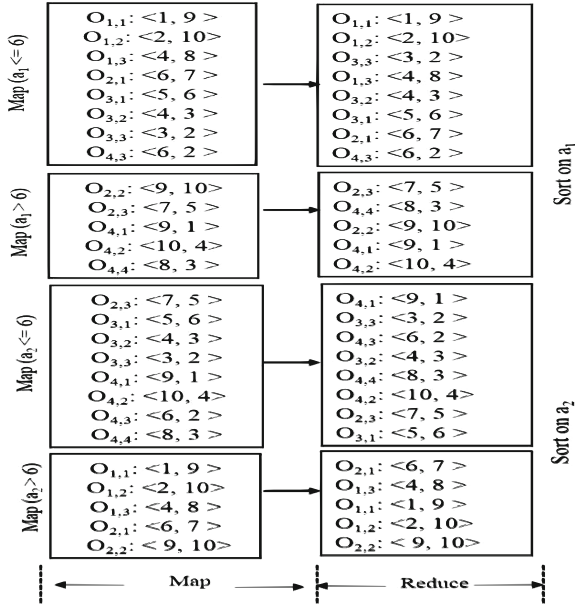


Fig. 3. TeraSort Implementation (2)

Definition. *Skyline*: An object $O \in DS$ is in skyline of DS (i.e., a skyline object in DS) if O is not dominated by any other object in DS . The skyline of DS , denoted by $Sky(DS)$, is the set of skyline objects in DS . For dataset DS , object $O_{1,1}, O_{4,1}$, and $O_{3,3}$ can dominate all other objects and they are not dominated by each other. Thus skyline query for dataset DS will retrieve $O_{1,1}, O_{4,1}$, and $O_{3,3}$ as $Sky(DS)$.

4 Distributed Skyline Using MapReduce

Proposed parallel algorithm of the skyline query on a distributed data environment consists of the following three phases.

Distributed Data Sorting Phase: We partition (map) each dataset vertically and sort each partition by TeraSort for further processing [21].

Map and Ranking Phase: In this phase, each Map function generates (*Key*, *Value*) pairs, where *Key* is the corresponding object *ID* and *Value* is the rank of corresponding attribute value. The output of this phase is (*ID*, *Rank*) pairs for each object.

Reduce and Skyline Computation Phase: Coordinator collects (*ID*, *Rank*) pairs to reduce data access between Map and Reducer. Coordinator maintains

counter for each retrieves object. When counter value becomes equal to m (attributes number), then coordinator stops $(ID, Rank)$ pair collection. The reduce function is invoked in this stage for skyline computation.

4.1 Distributed Data Sorting Phase

To compute skyline on MapReduce, we sort each dataset in a vertical fashion. We use TeraSort, which is a standard map/reduce sort. TeraSort sort the dataset using a one dimensional value as key in several computers. As a result, the dataset has been distributed into the computers in a sorted fashion. That means proposed approach split all of the attributes one by one. In other words, it splits the dataset attribute wise. Assume database DS is distributed among the machines in the MapReduce denoted by $\{M_1, M_2 \dots, M_t\}$. After successful implementation of TeraSort, all objects in M_i must precede those in M_j for any $(1 \leq i < j \leq t)$.

TeraSort

□ Round 1.

Map-shuffle

Every M_i ($1 \leq i \leq t$) randomly select samples from its local storage. It sends all the sampled objects to M_1 .

Reduce (only on M_1)

- Let S_{samp} be the set of samples received by M_1 , and sample size, $sz = |S_{samp}|$.
- Sort S_{samp} and pick boundary object $\{b_1, \dots, b_{t-1}\}$ where b_i is the $i(sz/t)$ -th object in ascending order of S_{samp} for $(1 \leq i \leq t-1)$.

□ Round 2.

Map-shuffle (Assumption: all b_i have been sent to all M_t)

Every M_i sends the objects in $(b_{j-1}, b_j]$ from its local storage to M_j , for each $(1 \leq j \leq t)$, where $b_0 = \infty$ and $b_t = \infty$ are dummy boundary objects.

Reduce:

Every M_i sorts the objects received in the previous phase.

Two dimensional TeraSort is shown in Figs. 2 and 3. Figure 2 shows that input dataset DS is distributed as DS_1, DS_2, DS_3 , and DS_4 . Initially, each dataset randomly chooses some objects $\{O_{1,2}, O_{2,1}, O_{3,1}, O_{4,1}, O_{4,3}\}$ as S_{samp} and send them to Map (sample). In reduce phase, each reduce worker sorts S_{samp} dataset according to each attribute. Objects $O_{2,1}$ and $O_{3,1}$ are chosen as boundary objects for each attribute. Coordinator distribute the boundary of a_1 and a_2 to each M with datasets DS_1, DS_2, DS_3 , and DS_4 . Figure 3 shows that every machine map its local storage according to these boundary objects. Here $\{O_{1,1}, O_{1,2}, O_{1,3}, O_{2,1}, O_{3,1}, O_{3,2}, O_{3,3}, O_{4,3}\}$ are mapped to Map ($a_1 \leq 6$), $\{O_{2,2}, O_{2,3}, O_{4,1}, O_{4,2}, O_{4,4}\}$ are mapped to Map ($a_1 > 6$), $\{O_{2,3}, O_{3,1}, O_{3,2}, O_{3,3}, O_{4,1}, O_{4,2}, O_{4,3}, O_{4,4}\}$ are mapped to Map ($a_2 \leq 6$), and $\{O_{1,1}, O_{1,2}, O_{1,3}, O_{2,1}, O_{2,2}\}$ are mapped to Map ($a_2 > 6$). Next, in the reduce phase all of these objects are sorted.

4.2 Map and Ranking Phase

In this phase, each map task independently operates a non-overlapping input dataset and calls the user-defined Map function to emit a list of *Key-Value* pairs (K, V) from its local storage in parallel. While key K is usually numeric, the value V can contain arbitrary information. In the proposed algorithm, each map function produces a $(ID, Rank)$ pair according to the sorted value of each attribute.

In this phase, TeraSort outputs are sent to map workers. Each map worker computes rank value for each object and creates $(ID, Rank)$ pairs. During this step reducers with higher attributes value are removed. In Fig. 4 shows that reducers of $a_1 > 6$ and $a_2 > 6$ are removed. Next each map worker assign a rank value to an object from the ordered object sequence. In Fig. 4 we can see that objects $O_{1,1}$ and $O_{4,1}$ hold the first position with rank “1”.

4.3 Reduce and Skyline Computation Phase

The coordinator retrieves an object from the ordered objects sequence of each Map worker. The coordinator maintains counter for each retrieved object. According to lemma 1 it will stopped when one of counter value becomes m .

In our running example at first objects $O_{1,1}$ and $O_{4,1}$ are send to the coordinator. Coordinator increments counter value for each retrieved object. Similarly it increments counter value for $O_{1,2}$ and $O_{3,3}$ in the next step. In the third step coordinator increments counter value for $O_{3,3}$. Now the counter value becomes “2” for object $O_{3,3}$ and which is equivalent to m . When the number of occurrence for an object becomes to m , coordinator stops further data pooling. Figure 4 shows the coordinating procedure of proposed MapReduce algorithm.

Lemma 1. *Object with number of occurrence or counter value equal to m is a dominant of all of the descendant objects.*

Proof. Assume a descendant object O_j was not dominated by it ancestor object O_i . O_i has the number of occurrence or counter value equal to m . Then according to the domination definition O_j must have smaller rank than O_i for all attributes or at least for an attribute. Since O_i visited before O_j that means O_i has better rank than O_j . In other words, O_j is dominated by O_i and O_j should be pruned. Hence an object with counter value m can dominate all of its descendant objects. \square

Next the final reduce phase produce skyline only by comparing attribute ranks of each retrieve object. Figure 4 shows that object $O_{1,1}$ has better values than object $O_{1,2}$ in both attributes a_1 and a_2 respectively. In this example, object $O_{1,2}$ is removed because it is dominated by $O_{1,1}$. No other object can become dominant of each other. Finally, reducer retrieves $\{O_{1,1}, O_{4,1}, O_{3,3}\}$ as *Sky (DS)* for our running example.

Algorithm 1. MapReduce Skyline

Input: Distributed Database DS .

Output: Skyline of DS .

Distribute Procedure

Distribute datasets into Machines $\{M_1, M_2, \dots, M_t\}$.

First Map Job

Map sample dataset S_{samp} into one mappers, say M_1 .

First Reduce Job

Sort S_{samp} and pick boundary objects $\{b_1, b_2, \dots, b_{(t-1)}\}$.

Second Map Job (All b_i have been sent to all M_t)

Every M_i mapped objects from its local storage according to boundary.

Second Reduce Job

Every M_i sorts the received objects.

Third Map Job

Map only top reducers output sorted by each attribute.

Assign each retrieve object corresponding rank r

Coordinate Procedure

Collect top $(ID, Rank)$ pairs from each mappers.

For each retrieve object maintain a counter.

If counter value equal to m for any object then stop.

Third Reduce Job

Receive $(ID, Rank)$ pairs from coordinator.

Perform domination check among retrieve objects.

Return skyline result.

5 Performance Evaluation

We set up a cluster of 4 commodity PCs in a high speed Gigabit networks, each of which has an Intel Core i7 3.4GHz CPU, 4 GB memory and Windows 8.0 OS. The machines are connected with a 100 Mbits/S LAN connection. We compile the source codes under JDK 1.6. We conduct a series of experiments with different data distributions, dimensionalities, and data cardinalities to evaluate the effectiveness and efficiency of our proposed methods. We compare proposed method against *SFS* which is the efficient non index based skyline computation algorithm proposed in [2]. Each experiment is repeated five times and the average result is considered for performance evaluation. Two data distributions are considered as follows:

Anti-Correlated: an anti-correlated dataset represents an environment in which, if an object has a small coordinate on some dimension, it tends to have a large coordinate on at least another dimension.

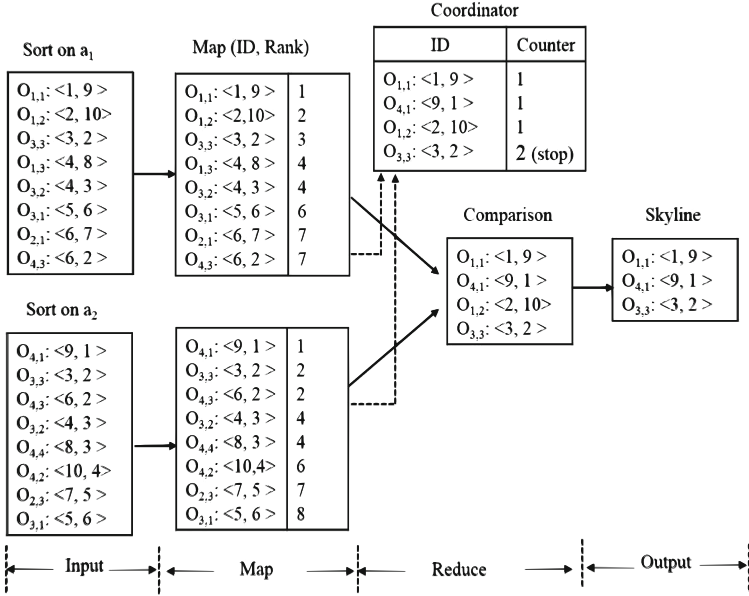


Fig. 4. Skyline query processing

Independent: for this type of dataset, all attribute values are generated independently using uniform distribution. Under this distribution, the total number of non-dominating objects is between that of the correlated and the anti-correlated datasets.

Effect of Data Distribution. We first study the effect of data distribution on our propose method. We fix the data cardinality to 1000 k, dataset dimensionality m to 6. Anti-correlated and independent data distribution are used for this experiments. The runtime results for this experiment are shown in Fig. 5. Both of the methods works well for independent data and slower for anti-correlated data. However, performance proposed method is better than *SFS* for both of the data distribution.

Effect of Dimensionality. We study the effect of dimensionality on our MapReduce techniques. We fix the data cardinality to 1000 K and vary dataset dimensionality n ranges from 2 to 8. The runtime results for this experiment are shown in Fig. 6(a) and (b). The result shows that as the dimension increases the performance of both methods become slower but propose method outperform on *SFS*.

Effect of Cardinality. For this experiment, we fix the data dimensionality to 6 and vary dataset cardinality ranges from 500 K to 1250 k. Figure 7(a) and (b)

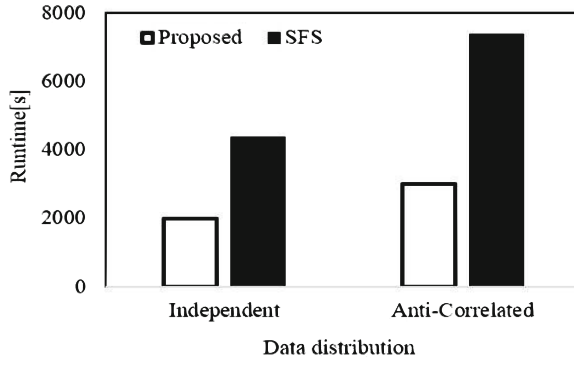


Fig. 5. Performance for different data distribution

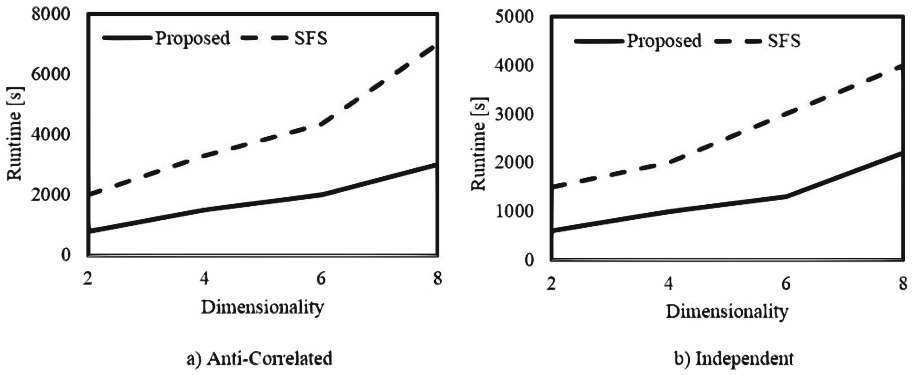


Fig. 6. Performance for different data dimension

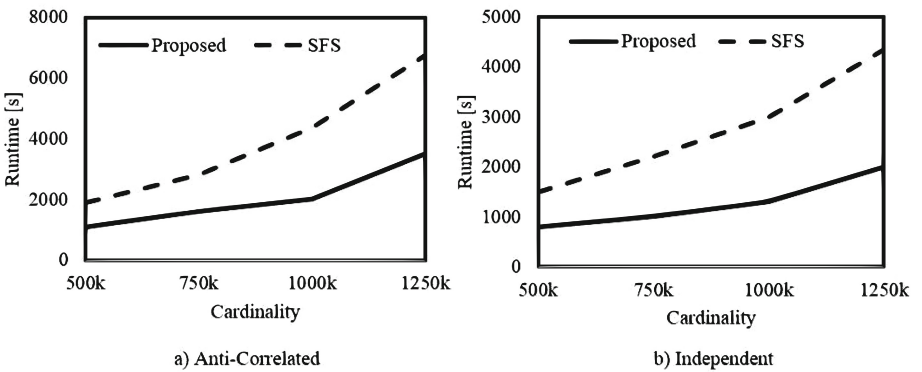


Fig. 7. Performance for different cardinality

shows the performance on anti-correlated and independent datasets. Both of the techniques are highly affected by data cardinality. If the data cardinality increases then the performances decreases. There exist significance difference on the performance of both methods and propose method is 2 times faster than *SFS*.

6 Conclusion

This paper addresses a distributed computation algorithm skyline query computation. In propose method each computer uses only a projected database that is vertically splitted from original database. We applied popular MapReduce architecture for large-scaled parallel computation. Propose method is able to localize sensitive information in a database. Extensive experiments demonstrate the efficiency of propose algorithm for synthetic datasets

It is worthy of being mentioned that this work can be expanded in a number of directions. First, from the perspective of parallel computing, how to compute skyline from streaming dataset. Secondly, to design an efficient index based (R-tree/B-tree) MapReduce algorithm are promising research topics.

Acknowledgments. This work is supported by KAKENHI (23500180, 25.03040) Japan.

References

1. Borzsonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of ICDE, pp. 421–430 (2001)
2. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: Proceedings of ICDE, pp. 717–719 (2003)
3. Tan, K.-L., Eng, P.-K., Ooi, B.C.: Efficient progressive skyline computation. In: Proceedings of VLDB, pp. 301–310 (2001)
4. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. *ACM Trans. Database Syst.* **30**(1), 41–82 (2005)
5. Chan, C.Y., Jagadish, H.V., Tan, K.-L., Tung, A.-K.H., Zhang, Z.: Finding *k*-Dominant skyline in high dimensional space. In: Proceedings of ACM SIGMOD, pp. 503–514 (2006)
6. Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the sky: efficient skyline computation over sliding windows. In: Proceedings of ICDE, pp. 502–513 (2005)
7. Balke, W.-T., Güntzer, U., Zheng, J.X.: Efficient distributed skylining for web information systems. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 256–273. Springer, Heidelberg (2004)
8. Vlachou, A., Doulkeridis, C., Kotidis, Y., Vazirgiannis, M.: SKYPEER: efficient subspace skyline computation over distributed data. In: Proceedings of ICDE, pp. 416–425 (2007)
9. Tao, Y., Xiao, X., Pei, J.: Subsky: efficient computation of skylines in subspaces. In: Proceedings of ICDE, pp. 65–65 (2006)

10. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: Proceedings of SIGMOD, pp. 467–478 (2003)
11. Dellis, E., Seeger, B.: Efficient computation of reverse skyline queries. In: Proceedings of VLDB, pp. 291–302 (2007)
12. Lee, J., Hwang, S., Nie, Z., Wen, J.-R.: Navigation system for product search. In: Proceedings of ICDE, pp. 1113–1116 (2010)
13. Lappas, T., Gunopulos, D.: Efficient confident search in large review corpora. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part II. LNCS, vol. 6322, pp. 195–210. Springer, Heidelberg (2010)
14. Wang, G., Xin, J., Chen, L., Liu, Y.: Energy efficient reverse skyline query processing over wireless sensor networks. *IEEE Trans. Knowl. Data Eng.* **24**(7), 1259–1275 (2012)
15. Zou, L., Chen, L., Özsu, M.T., Zhao, D.: Dynamic skyline queries in large graphs. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5982, pp. 62–78. Springer, Heidelberg (2010)
16. Tao, Y., Lin, W., Xiao, X.: Minimal MapReduce algorithm. In: Proceedings of SIGMOD, pp. 529–540 (2013)
17. Park, Y., Min, J., Shim, K.: Parallel computation of skyline and reverse skyline queries using MapReduce. In: Proceedings of VLDB, pp. 2002–2013 (2013)
18. Jiang, D., Tung, A.K.H., Chen, G.: MAP-JOIN-REDUCE: toward scalable and efficient data analysis on large clusters. *TKDE* **23**(9), 1299–1311 (2011)
19. Blanas, S., Patel, J.M., Ercegovac, V., Rao, J., Shekita, E.J., Tian, Y.: A comparison of join algorithms for log processing in MaPReduce. In: Proceedings of SIGMOD, pp. 975–986 (2010)
20. Vernica, R., Carey, M.J., Li, C.: Efficient parallel set-similarity joins using MapReduce. In: Proceedings of SIGMOD, pp. 495–506 (2010)
21. O'Malley, O.: Terabyte sort on apache hadoop. In Yahoo Technical report (2008)

Short-Term Speed Prediction on Urban Highways by Ensemble Learning with Feature Subset Selection

Mohammad Arif Rasyidi and Kwang Ryel Ryu^(✉)

Department of Electrical and Computer Engineering, Pusan National University,
2 Busandaehak-ro 63beon-gil, Geumjeong-gu, Busan 609-735, South Korea
{arifrasyidi, krryu}@pusan.ac.kr

Abstract. Accurate traffic speed prediction is essential in the development of intelligent transportation systems. Even though a lot of methods have been proposed for traffic prediction, few works pay attention to the application of ensemble learning and feature subset selection. In this paper, we propose an implementation of ensemble learning using combination of M5 model tree and bagging to tackle traffic speed prediction. A method to select optimal neighboring links as features for our prediction model is also introduced, and different feature subset selection methods are compared. Experimental results show that the proposed ensemble with feature subset selection outperforms both single model and nonparametric model (k -NN).

Keywords: Traffic prediction · Ensemble learning · Model tree · M5 · Bagging · Feature subset selection

1 Introduction

Traffic prediction is the task of predicting either the futures value (forecast) of a certain traffic variable [1–6], or some unknown or missing values from the collected traffic data [7]. The object of prediction includes but not limited to traffic speed, flow rate (volume rate), travel time, and road occupancy. In this work, we will focus on predicting future traffic speed on urban highways. Traffic speed is the measurement of average speed of all vehicles passing a link in a predefined time period. Accurate traffic speed prediction is of vital importance in the development of Intelligent Transportation System (ITS). While the drivers cannot be directly benefited from the predicted traffic speed, the prediction can be used to estimate the travel times for the drivers. Other application of traffic speed prediction includes potential traffic congestion warning and fastest route suggestion.

Traffic prediction models can generally be classified into two categories: analytical and statistical models. Analytical models (also called traffic theory based deductive models) use a simulation model to mimic the behavior of a specific traffic system and make prediction based on the simulation result under certain theoretical assumptions [6, 8]. While these models are useful in foreseeing the effect of various factors (e.g., weather, traffic incident, road works, etc.) to the traffic system, they are computationally expensive, require a lot of time to analyze the problem, and thus difficult for

use in real-time estimation or short-term forecasting [8]. Analytical models are also sensitive to the simulation parameters. As a result of the complexity of simulation models, small error in setting up the input parameters can lead to large error in the results [9]. Statistical models (also called inductive or empirical models) on the other hand, make prediction based on the hidden relationship or correlation found in the collected data [6, 8]. The models therefore are data-driven and fewer (or no) theoretical assumptions are needed to build them. As with more and more traffic surveillance systems being developed, real time traffic data can be obtained more easily, resulting in increase of interest in data-driven models. Simple historical average prediction model is an example of statistical models that is widely used in practice. However, this method is prone to error since the traffic pattern often deviates from historical pattern over time. To overcome this problem, researchers have tried different forecasting approaches developed to overcome the limitation of historical average prediction. Common approaches include time series methods [3, 10], Kalman filter [11], linear regression [4], Artificial Neural Network (ANN) [1, 5, 12], Support Vector Machine (SVM) [5, 6, 8], and other nonparametric models such as k -Nearest Neighbor (k -NN) [13, 14] and local linear regression [2].

In our previous work, we have implemented a k -NN algorithm for short term traffic speed prediction [14]. While k -NN shows very good prediction accuracy compared to linear regression and model tree, it is not scalable. With k -NN, at the time of training we only need to select k , the best number of neighbors to be used for prediction, and simply store all the training data. All the work is then done at the time of prediction: finding the k nearest neighbors, combining the prediction results from each of them, and so on. This is not a problem when we work with a small set of links. However, when we work with a large number of links in the urban city roads, a significant amount of computation time is needed to find k nearest neighbors from a large volume of training data, making real time prediction not feasible.

In this study, we propose an application of ensemble learning of model trees based on a bagging method. Ensemble learning is a type of learning in which we select a set (ensemble) of hypotheses and combine their predictions to produce the final result for a given query. The main advantage of ensemble learning is the improvement of accuracy and robustness compared to the use of a single model [15]. Ensemble learning has gained an intense interest over the years and good results have been reported. Assaad et al. [16] use boosting-based recurrent neural networks ensemble for improving time series forecasting. Shigei et al. [17] use bagging and AdaBoost for vector quantization. Yu et al. [18] use multistage radial basis function (RBF) neural network ensemble for exchange rates forecasting. However, other than the ensemble of RBF neural network for traffic flow prediction [19], there are few results on ensemble learning implementation for traffic prediction.

Another issue in traffic prediction is the diverse choices of features for the learning algorithm. Features determine the performance of a model, since a model is only as good as its features [20]. Various kinds of features have been proposed and used in the previous works. Univariate models such as historical average, moving average, and autoregressive integrated moving average (ARIMA) only use current and historical data of the target link as features. Multivariate models on the other hand, often combine historical data of the target link with some time-related features such as time

(hour, minute), day of week (Monday–Sunday), day type (weekday, weekend, holiday) [7], and other related features such as weather condition [12] and traffic incident [13]. Other types of features that are often used are the current and historical data of neighboring links. Generally, there are two approaches to using these types of features: aggregated values and individual values. The aggregated approach exploits the fact that things in contiguous regions could be related in a systematic way [21]. In this approach, the neighboring links are typically divided into several regions or clusters. The values of all the links in each region are then combined in a predetermined way, e.g., by averaging or by weighted averaging. Since the values are aggregated, some potentially important information might be lost, and thus careful consideration must be given in categorizing the neighboring links and combining their values. Kamarianakis and Prastacos [3] compare the performances of univariate and multivariate time-series predictions and propose to categorize neighboring links based on distance and use their weighted average speed as features. Min and Wynter [10] try to improve Kamarianakis’ method by proposing the use of travel time, instead of distance to categorize the neighbors.

In contrast, in individual approaches, the individual values of all neighboring links are directly fed to the learning algorithm as features. In previous works, Park and Rilett [1] try the combination of two uplinks and two downlinks as inputs to a multilayer feedforward neural network. Other than time-related binary features, Lee et al. [7] also use the speed of links directly connected to the target link as the input to a neural network model. Our previous work [14] also uses neighboring link data and compares the performance with those of models built using only historical data of the target link. Individual approaches typically employ more features and may preserve link correlation better than aggregated approaches. However, if there are too many neighboring links employed, the performance of the learned model might suffer especially if irrelevant or redundant features are included. A careful selection of the neighboring links as features is therefore essential in achieving a good performance of the learned model. However, no previous works have addressed the issue of this feature subset selection problem, i.e., the selection of the individual neighboring links as features. The objectives of this research are therefore to apply the ensemble learning for traffic speed prediction, compare the result to the performance of single models and nonparametric models, and introduce a method for selecting neighboring links as candidate features for the learning algorithm.

The rest of the paper is organized as follows. The following section provides the formalization of traffic speed prediction problem. In Sect. 3, we discuss the proposed method in applying ensemble learning for traffic speed prediction and selecting the neighboring links as candidate features. Section 4 gives the experimental procedure. Results are discussed in Sect. 5 and conclusion is given in the final section.

2 Traffic Speed Prediction Problem

The traffic condition of a target link in a near future directly depends on not only the current and recent traffic conditions of the target link itself but also those of its neighboring links both in the upstream and downstream. Consider the scenario where

an accident occurs on an upstream link. There might be fewer vehicles to pass the target link than before, which in turn will increase the average traffic speed on the target link. Things are similar for the downstream links. When link 7 in Fig. 1 for example is congested, the outflow of the target link will be clogged, thus lowering the average speed on the target link. Therefore, to make a short-term traffic prediction of a target link, we need to inspect the recent conditions of the target and those of the neighboring links both in the up and downstream. As the prediction is targeted further into the future, links farther in the up and downstream may have to be investigated. Other factors that may affect the prediction include the historical pattern of the target link obtained from a long-term historical traffic data, time zone of a day, day type (weekdays or weekends), weather conditions, and so on.

Let t denote the current time and $(x_{i,1}, x_{i,2}, \dots, x_{i,t})$ be the sequence of recently observed traffic speeds on link i measured at equal time intervals up to t . Given the current and recent speed data $\mathbf{x}_{i,t} = (x_{i,(t-n)}, \dots, x_{i,(t-2)}, x_{i,(t-1)}, x_{i,t})$ for $i \in \{\text{target and neighboring links in the upstream and downstream}\}$ and optionally a set of other related features $\mathbf{q}_t = (q_{1,t}, q_{2,t}, \dots, q_{l,t})$ measured at the current time, we want to estimate the future speed of target link k at time $(t+m)$ for some positive value m : $x_{k,(t+m)}$ as shown in Fig. 2. Here, m is the prediction horizon specified by the user and n is the look-back period, determining how many time steps behind we are willing to look back to make our prediction. If we let \mathbf{x}_t denote $(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}, \dots, \mathbf{x}_{h,t})$ where h is the number of inspected links and let $y_{(t+m)}$ be the future speed of the target link at time $(t+m)$, our problem is then to find a function f that maps $\mathbf{X}_t = (\mathbf{x}_t, \mathbf{q}_t)$ to the future speed $y_{(t+m)}$ as accurate as possible:

$$f^* = \arg \min_f \sum_{(\mathbf{X}_t, y_{(t+m)}) \in E} L(y_{(t+m)}, f(\mathbf{X}_t)). \quad (1)$$

where L is a loss function and E is the set of training examples.

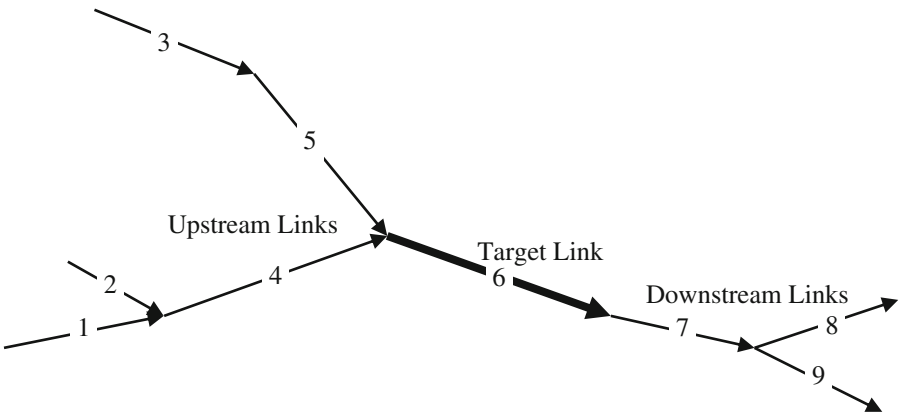


Fig. 1. A sample road network showing a target link (thick line) and its upstream and downstream neighbors. The direction of the arrows indicates the direction of the traffic

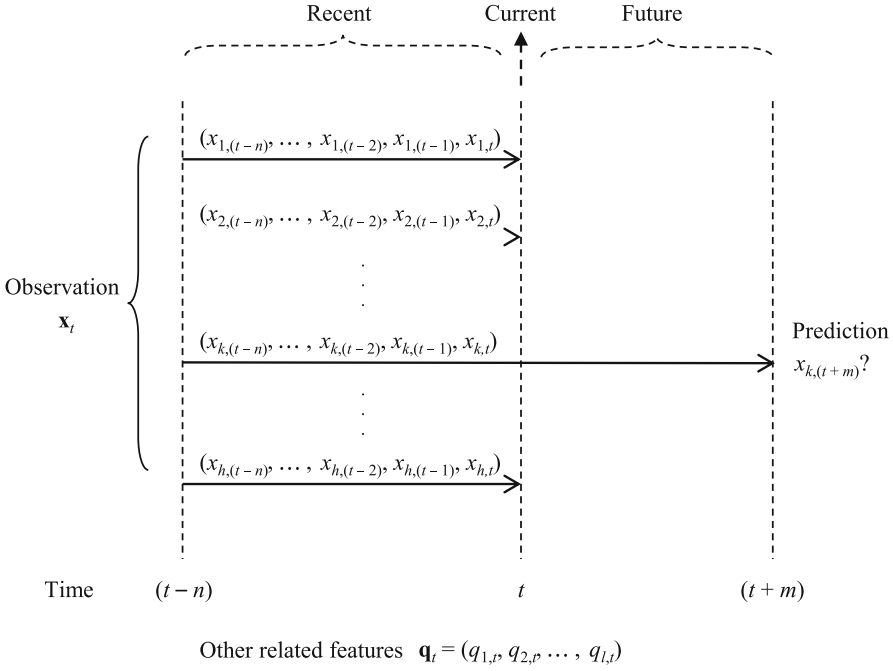


Fig. 2. The relevant features for traffic speed prediction

3 Proposed Method

3.1 Model Tree

Decision tree is widely used both for classification and numerical prediction (regression). Decision trees for regression generally can be differentiated into two categories: those that have constants at their leaves, also called *regression trees*; and those that build a nontrivial model at each of their leaves, called *model trees*. A popular algorithm for building model tree is M5 that is developed by Quinlan [22] and improved by Wang and Witten [23]. M5 outputs model tree with multivariate linear models at its leaves. To construct a model tree, M5 uses top-down approach, using variance reduction as heuristics for selecting the best split (test) at each node. Splitting or branching in M5 is terminated when the target value of all the instances that reach a certain node only vary very slightly (e.g., when the standard deviation is less than 5 % of the standard deviation of the original data set), or only a few instances remaining (e.g., fewer than 4). A linear model is then built for each leaf and interior node using training data that reach that node. The resulting linear models are then simplified to avoid overfitting by dropping terms greedily to minimize the expected error. Once all the linear models have been simplified, the tree is pruned back from the leaves as long as the expected error decreases. Finally, to compensate for the discontinuity of the adjacent linear models, at the time of prediction, the predicted value at the leaf is

filtered along the path back to the root and smoothed at each interior node. An example of model tree is shown in Fig. 3.

The main advantages of model trees over nonparametric models are their explanatory power and the less amount of computation time demanded for prediction. However, their performance is often inferior to that of nonparametric models as shown in our previous work [14]. To improve the performance of model tree, we adopt an ensemble learning approach that combines several models to achieve better accuracy. Our ensemble learning implementation for traffic speed prediction is explained in the next section.

3.2 Ensemble of Model Trees for Traffic Speed Prediction

Ensemble learning generates several models and combines them together to make a prediction. There are two popular methods in ensemble learning: bagging [24] and boosting [25]. Boosting explicitly seeks models that complement one another by iteratively building models, with each one encouraged to become experts for instances handled incorrectly by earlier ones. While this approach is very powerful, it often requires modification to the learning algorithm used. Bagging on the other hand, exploits the instability inherent in learning algorithms to reduce the variance and help to avoid overfitting. Bagging works by creating a set of replicate datasets, each of which is called a *bootstrap*, by sampling with replacement from the original training set. It then builds several models separately, one for each bootstrap, and weights them equally. Bagging is simple to apply and does not require any modification to the learning algorithm. However, it works well only when the learning scheme is unstable [24], i.e., when small changes in the training data will result in very different predictors.

In this study, we use bagging in combination with M5 model tree. Similar to conventional decision trees, model tree learning is unstable. A small difference in the training data can result in quite different model trees, making it works well when combined with bagging. To increase the diversity of our ensemble, we try to make the M5 algorithm as unstable as possible by turning off pruning. The learning algorithm to

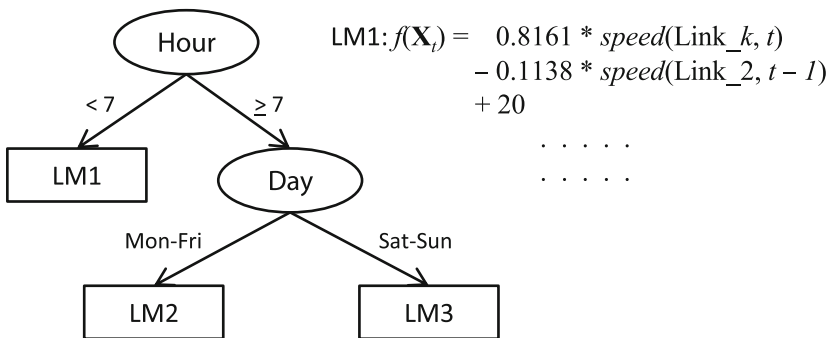


Fig. 3. An example of model tree with linear models at its leaves

build an ensemble of model trees using bagging for our traffic speed prediction can be described as following:

1. Given a training set E , create s bootstraps (B_1, B_2, \dots, B_s) by sampling the training set E with replacement. *Sample ratio* is used to determine the size of bootstraps (as a fraction of the training data). Its value must be greater than 0 and should be lower than or equal to 1.
2. Train an unpruned model tree using each bootstrap B_i ($i = 1, \dots, s$) obtained from the previous step: f_1, f_2, \dots, f_s .
3. Store all the resulting models to form an ensemble.

At the time of testing, given an input query $\mathbf{X}_t = (\mathbf{x}_t, \mathbf{q}_t)$, the prediction $y_{(t+m)}^*$ is made by averaging the outputs of all the model trees in the ensemble:

$$y_{(t+m)}^* = \frac{1}{s} \sum_{i=1}^s f_i(\mathbf{X}_t). \quad (2)$$

3.3 Selecting Neighboring Links

The structure of the links of urban highways can be represented as a directed graph where each link becomes an edge connecting a source point (head vertex) to a destination (tail vertex). To select the neighboring links of a target link, we divide the selection process into two parts: upstream selection and downstream selection. In the upstream selection we select only the links whose tails coincide with or have directed paths to the head of the target link. We start from the target link and continue adding incoming links recursively until no remaining link is within a specified distance threshold. The downstream selection is just the opposite of the upstream selection. We select only the links whose heads coincide with or have directed paths from the tail of the target link.

There are various distance metrics that can be used as a link selection parameter. Typical distance metrics include *the actual length*, *travel time*, and *spread/depth*. With the *travel time* metric, the distance of each link is estimated as its actual length divided by the average speed on that link. When using the *spread/depth* metric, every link is considered to have a unit length. The number of links selected using these metrics will vary depending on the geometric structure of the roads. If the area around the target link is densely connected, using the actual distance or travel time metric as a selection parameter might result in a large set of neighboring links. In a sparse area, however, the same metrics will result in a small set of links. The distance or travel time metric might capture the inflow-outflow relationship better than the simpler spread/depth metric.

Note that the number of selected neighboring links gradually (instead of drastically) increases if we increase the selection parameter carefully. To select the best set of neighboring links, we begin with a small selection parameter (e.g., depth = 1), evaluate the selected set of links using the targeted learning algorithm, increase the parameter, and reevaluate it until we encounter one or more stopping criteria. A simple way to stop this process would be to check if the selection parameter has

reached the allowed maximum value or if a specified time limit has passed. A more sophisticated method would be to see if the validation error rate ceases to improve for one or more consecutive iterations. What we want is the set of neighboring links with the lowest validation error when tested with the targeted learning algorithm.

3.4 Feature Subset Selection

Feature subset selection refers to the task of finding a subset of features that allows the learning algorithm to learn the best model. While model trees already have internal feature selection schemes, it is often still advantageous to apply additional feature subset selection scheme before building the model. The advantages of using feature subset selection include reduced computational cost (by removing useless or redundant features) and avoidance of overfitting to training data.

Feature subset selection methods can be categorized into two main approaches: *wrapper* and *filter* methods. In wrapper method, we evaluate the subset using the machine learning algorithm that will be employed for learning. Because we build a model for each feature set to be evaluated, this method is computationally expensive. However, wrapper usually gives the best performing feature set for that particular learning algorithm. Filter method on the other hand, makes an independent assessment of features based on general characteristics of the data. It is much faster than the wrapper method and usually uses variable (feature) ranking as selection mechanism. There are various filter methods that can be used for regression problems: correlation based feature selection (CFS), Principle Component Analysis (PCA), selection with a Support Vector Machine (SVM), and Recursive Feature Elimination (RFE) with SVM. CFS iteratively adds the feature with the most information (correlation) regarding the label and with the least redundancy to the already selected features. PCA can be used as a form of feature selection/dimensionality reduction by keeping only the best number of principal components obtained from certain data transformation and projections. Feature selection with SVM trains a linear SVM model using all the available training data, sorts the features based on the absolute weights, and keeps only the top k features. When using RFE, we repeatedly train an SVM, discard features with the lowest weights (usually specified by a ratio or fraction of the evaluated features), and retrain until only a specified number of features remain.

In this study, we use filter method as feature subset selection method to reduce the training time. The performance will be compared to a wrapper method to find out if there is any significant drop in performance.

4 Experimental Procedure

4.1 Data Preparation

The historical traffic speed data are provided by the Transportation Information Service Center of Busan Metropolitan City. The data consists of 30 day traffic speed data from September 1 to September 30, 2013 for all the links in Busan. The speeds are measured every 5 minutes from 00:00 to 23:59, giving 288 observations per day and

8640 observations for each link in the whole month. Our target link for speed prediction is the one in a highway called Beonyeong-ro where the speed changes dynamically during the day. Applying the neighboring link selection method described in Sect. 3.3, we choose spread/depth as the selection parameter, because the number of selected neighbors increases more slowly than when we use distance or travel time as shown on Fig. 4. The selection limit is set to the maximum depth of 5. We evaluate the set of neighboring links selected with depth from 1 to 5 and choose the one with the lowest validation error to build the final model.

Other than the recent speed data of the target and neighboring links, we also include six time-related features: day of month (1–30), day of week (1–7), hour (0–23), minute (0–55), minute of day (0–1435), and day type (weekday, weekend, holiday). Each dataset is split into two partitions: 70 % of the data are used for training and the rest 30 % (roughly one week) are used for testing. Six short-term prediction horizons are tested (1–6 steps/5–30 min ahead prediction) and look-back period is set to 5 (i.e., we use data from the current time to 5 steps back for each link). We also investigate the effect of increasing the range of neighboring links when making 12-step-ahead (one hour) predictions.

4.2 Prediction Models

We compare the performance of our proposed ensemble of model trees with a single M5 model tree and k -NN. From the results of preliminary experiments, we set the size of ensemble to be 50 and the sample ratio 0.7. The variant of k -NN that we use is the weighted k -NN where each neighbor is weighted by the inverse of its distance. We use Euclidean distance for the distance measure and ten-fold cross validation to select the optimal number of neighbors (k) for our model.

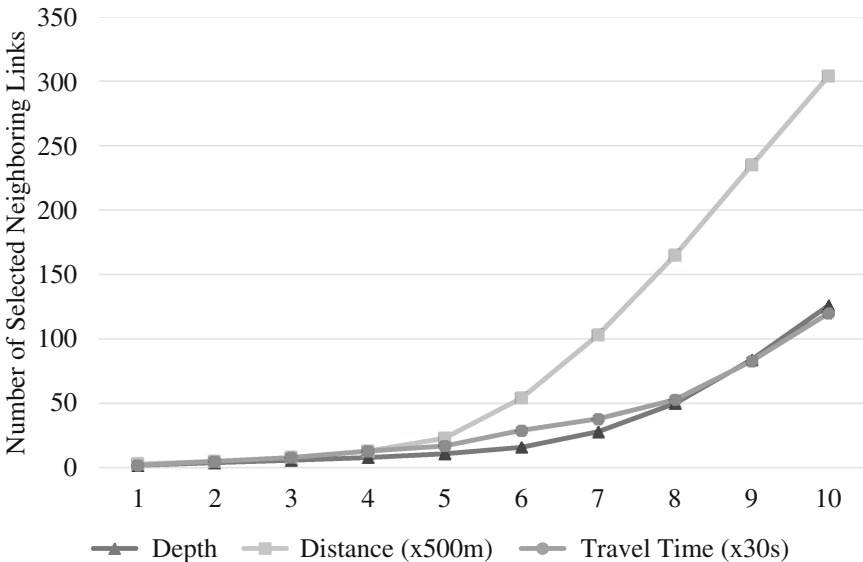


Fig. 4. The number of neighboring links selected across different selection parameters

4.3 Feature Selection Method

There are five feature selection methods that we try: CFS, PCA, SVM, RFE, and a wrapper. We also investigate the case of not using any feature selection method. We select ten features for our learning algorithm using CFS, SVM and RFE. For PCA, we keep 95 % of variance to reduce the dimensionality. In using a wrapper, we evaluate the features with the forward selection approach: we start with an empty set of features and, in each iteration, choose a feature that gives the best performance when added to the current set of features, and repeat until we meet a stopping criterion. We stop the execution of forward selection when the validation error does not improve for two consecutive iterations. The validation error is measured using ten-fold cross validation.

4.4 Error Measurement

To evaluate the performance of the proposed method, we measure the mean absolute relative error (MARE) as shown in Eq. (3). Here, y_i and y_i^* represent the actual and predicted speed values of the target link at time i , respectively.

$$\text{MARE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - y_i^*}{y_i} \right| \quad (3)$$

5 Results

5.1 Dataset Selection

We begin the experiment by finding the best dataset across different prediction horizons. To evaluate each dataset, we use ten-fold cross validation and MARE as error measure. Dataset with the lowest validation error will be chosen and used to build the final model. In case of a tie, we will choose the smaller dataset. Below, we use the term Dataset i to refer to the dataset obtained using the selection parameter of *depth* equal to i (see Sect. 4.1). Table 1 shows the best dataset (Dataset i is indicated simply by i) for each combination of prediction and feature selection methods. While it clearly shows that Dataset 4 gives the best validation error in many cases, it does not show any clear pattern describing the relationship between different prediction horizons and the optimal range of neighboring links. Simple neighboring link selection as used by Park and Rilett [1] that selects the combination of two uplinks/downlinks (similar to Dataset 1 and 2) or by Lee et al. [7] that selects only links directly connected to the target link (similar to Dataset 1) does not look optimal as revealed in our experiments summarized in Table 1. Our proposed method of selecting neighboring links by starting with a small selection parameter, gradually increasing it, and reevaluating the selected dataset has proven to be a viable option to determine the best dataset with the best range of neighboring links for our learning algorithm.

Table 1. Best dataset for each combination of prediction and feature selection methods

Method	Prediction horizon	Feature selection					
		None	CFS	PCA	SVM	RFE	Wrapper
Ensemble of M5	1	4	4	4	4	4	5
	2	4	4	4	4	4	5
	3	4	4	4	4	4	5
	4	4	4	4	4	4	4
	5	4	4	4	1	4	5
	6	3	4	5	4	4	4
M5	1	4	4	5	4	4	4
	2	4	4	4	4	4	4
	3	4	3	4	4	4	4
	4	4	3	4	4	4	5
	5	5	4	5	5	4	4
	6	3	4	4	4	4	3
<i>k</i> -NN	1	4	1	5	3	3	1
	2	5	4	5	4	4	4
	3	4	4	5	3	4	3
	4	4	5	5	3	4	5
	5	4	4	5	2	4	4
	6	5	4	5	4	4	3

5.2 Comparison of Different Feature Subset Selection Methods

Using the best dataset obtained from the previous experiment, we build the final models using different feature subset selection schemes. We then apply the models to the test set and measure the performance. Table 2 shows the performance of our predictors using different feature subset selection methods. Wrapper and no feature selection are provided for comparison purposes. Among all the filter methods tried, CFS is the clear winner for the ensemble of model trees as well as the single model tree. CFS even outperforms the performance of wrapper in some prediction horizons using our proposed models. *k*-NN, on the other hand, performs the best with SVM and RFE as the feature subset selection methods.

A closer look to the comparison shows that our ensemble performs well enough, even without feature selection. This is because model tree already has an internal feature selection scheme that selects the best feature for splitting at each node. However, the result shows that the performance can further be improved using CFS as an extra feature subset selection method. The experiment also shows that the average time required for training (feature subset selection and building the model) drops from 127.71 s to 27.18 s and that for testing from 2.55 s to 0.51 s when we use CFS instead of no feature subset selection (the experiment is conducted on a personal computer with 3.40 GHz CPU and 3 GB maximum Java heap memory).

Table 2. Performance comparison of ensemble of model trees, single model tree, and k -NN using different feature subset selection methods. Best two values are highlighted

Algorithm	Prediction horizon	Feature selection					
		None	CFS	PCA	SVM	RFE	Wrapper
Ensemble of M5	1	0.0286	0.0284	0.0415	0.0286	0.0287	0.0284
	2	0.0417	0.0404	0.0507	0.0416	0.0413	0.0404
	3	0.0517	0.0494	0.0585	0.0520	0.0516	0.0499
	4	0.0517	0.0576	0.0635	0.0611	0.0597	0.0571
	5	0.0672	0.0651	0.0681	0.0702	0.0680	0.0652
	6	0.0720	0.0704	0.0746	0.0729	0.0765	0.0689
M5	1	0.0313	0.0294	0.0497	0.0298	0.0296	0.0294
	2	0.0480	0.0418	0.0566	0.0427	0.0426	0.0423
	3	0.0602	0.0554	0.0651	0.0548	0.0536	0.0524
	4	0.0656	0.0616	0.0703	0.0623	0.0625	0.0587
	5	0.0759	0.0690	0.0795	0.0699	0.0706	0.0688
	6	0.0798	0.0758	0.0781	0.0758	0.0807	0.0701
k -NN	1	0.0457	0.0479	0.0638	0.0320	0.0326	0.0325
	2	0.0524	0.0488	0.0697	0.0433	0.0438	0.0426
	3	0.0584	0.0557	0.0751	0.0538	0.0527	0.0519
	4	0.0638	0.0619	0.0799	0.0617	0.0610	0.0579
	5	0.0691	0.0680	0.0843	0.0695	0.0679	0.0640
	6	0.0736	0.0718	0.0893	0.0719	0.0759	0.0694

5.3 Performance Comparison

We compare the performance of our proposed model using CFS feature subset selection with the performance of single M5 model tree and k -NN. Table 3 shows the performance comparison of the proposed method with the best results of single M5 and k -NN tested with various feature subset selection filters. Our proposed method clearly wins on all prediction horizons against both of the competitors. Even when we include the result of the competitors obtained using wrapper feature subset selection as shown in Table 4, our proposed method still outperforms both single M5 and k -NN on the first four prediction horizons and only loses by a small margin to k -NN with wrapper on prediction horizons of 5 and 6.

5.4 Longer Prediction Horizon

In the last experiment, we compare the effect of increasing the range of neighboring links on longer prediction horizon. We test our proposed method with CFS on 1 hour (12 steps ahead) prediction. Table 5 shows the comparison of validation and test errors for 30 minute and 1 hour predictions. The result shows that for longer prediction horizon, we need to examine a wider area to find the best set of neighboring links for our prediction. While this does not seem to agree with our previous result that shows that there is no clear pattern describing the relationship between the prediction horizon

Table 3. Ensemble of model trees with CFS vs. the best of single model tree and k -NN tested with various feature subset selection filters

Prediction horizon	Ensemble of M5	M5	k -NN
1	0.0284	0.0294	0.0320
2	0.0404	0.0418	0.0433
3	0.0494	0.0536	0.0527
4	0.0576	0.0616	0.0610
5	0.0651	0.0690	0.0679
6	0.0704	0.0758	0.0718

Table 4. Ensemble of model trees with CFS vs. the best of single model tree and k -NN tested with various feature subset selection filters and a wrapper

Prediction horizon	Ensemble of M5	M5	k -NN
1	0.0284	0.0294	0.0320
2	0.0404	0.0418	0.0426
3	0.0494	0.0524	0.0519
4	0.0576	0.0587	0.0579
5	0.0651	0.0688	0.0640
6	0.0704	0.0701	0.0694

Table 5. Performance comparisons of 30 minute and 1 hour prediction across different datasets

Dataset	Validation		Test	
	30 min	1 h	30 min	1 h
1	0.0949	0.1531	0.0880	0.1254
2	0.0864	0.1418	0.0774	0.1179
3	0.0860	0.1487	0.0774	0.1224
4	0.0704	0.1438	0.0704	0.1190
5	0.0704	0.1443	0.0706	0.1191
6	0.0848	0.1450	0.0768	0.1200
7	0.0847	0.1317	0.0765	0.1140
8	0.0855	0.1316	0.0751	0.1124
9	0.0853	0.1385	0.0773	0.1179
10	0.0857	0.1383	0.0774	0.1169

and the optimal range of neighboring links, more investigation on different target links and prediction horizons will be needed before we reach a conclusion. Nonetheless, our proposed approach still stands as a viable option for selecting the optimal neighbors with appropriate stopping criterion.

6 Conclusion

We have proposed the application of ensemble learning using combination of bagging and M5 model tree for a short-term speed prediction problem and a method for selecting optimal neighboring links as the features for our prediction model. The experiments show that our proposed ensemble method with feature subset selection is superior to both single model (M5) and nonparametric model (k -NN) in all the prediction horizon. Even when we compare the performance of our ensemble with CFS to M5 model tree and k -NN with wrapper, our proposed method still dominates, giving best results on the first four prediction horizons and only loses by a small margin to k -NN with wrapper on 25 minute and 30 minute predictions. Experiments for short-term predictions show that there is no clear relationship between the prediction horizon and optimal range of neighbors. However, different result is shown on 1 hour (12 steps ahead) prediction where wider range of neighboring links results in better performances. In future works, we will investigate this issue further on different target links and various prediction horizons. We will also try to improve the accuracy of the prediction by combining the output of the model with the long-term historical pattern of the target link.

Acknowledgments. This research was supported by MSIP (Ministry of Science, ICT & Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2013-(H0301-13-1012)) supervised by the NIPA (National IT Industry Promotion Agency).

References

1. Park, D., Rilett, L.R.: Forecasting freeway link travel times with a multilayer feedforward neural network. *Comput. Civ. Infrastruct. Eng.* **14**, 357–367 (1999)
2. Sun, H., Liu, H.X., Xiao, H., Ran, B.: Short term traffic forecasting using the local linear regression model. *UC Irvine Cent. Traffic Simul. Stud.* (2002)
3. Kamarianakis, Y., Prastacos, P.: Forecasting traffic flow conditions in an urban network: comparison of multivariate and univariate approaches. *Transp. Res. Rec. J. Transp. Res. Board.* **1857**, 74–84 (2003)
4. Zhang, X., Rice, J.A.: Short-term travel time prediction using a time-varying coefficient linear model. *Transp. Res. C.* **11**, 187–210 (2003)
5. Vanajakshi, L., Rilett, L.R.: A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed. In: 2004 IEEE Intelligent Vehicles Symposium. pp. 194–199 (2004)
6. Wu, C.-H., Ho, J.-M., Lee, D.T.: Travel-time prediction with support vector regression. *IEEE Trans. Intell. Transp. Syst.* **5**, 276–281 (2004)
7. Lee, E.-M., Kim, J.-H., Yoon, W.-S.: Traffic speed prediction under weekday, time, and neighboring links' speed: back propagation neural network approach. In: Huang, D.-S., Heutte, L., and Loog, M. (eds.) *Advanced Intelligent Computing Theories and Applications. with Aspects of Theoretical and Methodological Issues SE – 62*, pp. 626–635. Springer, Heidelberg (2007)

8. Wang, J., Shi, Q.: Short-term traffic speed forecasting hybrid model based on Chaos-wavelet analysis-support vector machine theory. *Transp. Res. Part C Emerg. Technol.* **27**, 219–232 (2013)
9. Institute of Transportation Engineers California Border Section Highway Capacity Task Force: A report on the use of traffic simulation models in the San Diego region (2004)
10. Min, W., Wynter, L.: Real-time road traffic prediction with spatio-temporal correlations. *Transp. Res. Part C Emerg. Technol.* **19**, 606–616 (2011)
11. Vanajakshi, L., Subramanian, S.C., Sivanandan, R.: Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses. *IET. Intell. Transp. Syst.* **3**, 1–9 (2009)
12. Dunne, S., Ghosh, B.: Weather adaptive traffic prediction using neurowavelet models. *IEEE Trans. Intell. Transp. Syst.* **14**, 370–379 (2013)
13. Guo, F., Krishnan, R., Polak, J.W.: Short-term traffic prediction under normal and incident conditions using singular spectrum analysis and the k-nearest neighbour method. In: *IET and ITS Conference on Road Transport Information and Control (RTIC 2012)*, pp. 1–6 (2012)
14. Rasyidi, M.A., Kim, J., Ryu, K.R.: Short-Term Prediction of Vehicle Speed in Main City Roads using k-Nearest Neighbor Algorithm. In: *Proceedings of 2013 Korea Intelligent Information System Society Conference on Intelligent Technology and Data Science.*, pp 190–195. Korea Intelligent Information System Society, Seoul (2013)
15. Garcia-Pedrajas, N., Hervas-Martinez, C., Ortiz-Boyer, D.: Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Trans. Evol. Comput.* **9**, 271–302 (2005)
16. Assaad, M., Boné, R., Cardot, H.: A new boosting algorithm for improved time-series forecasting with recurrent neural networks. *Inf. Fusion* **9**, 41–55 (2008)
17. Shigei, N., Miyajima, H., Maeda, M., Ma, L.: Bagging and AdaBoost algorithms for vector quantization. *Neurocomputing* **73**, 106–114 (2009)
18. Yu, L., Lai, K.K., Wang, S.: Multistage RBF neural network ensemble learning for exchange rates forecasting. *Neurocomputing* **71**, 3295–3302 (2008)
19. Chen, L., Chen, C.L.P.: Ensemble learning approach for freeway short-term traffic flow prediction. In: *IEEE International Conference on System of Systems Engineering, 2007. SoSE '07*, pp. 1–6 (2007)
20. Flach, P.: *Machine Learning: the Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press, New York (2012)
21. Giacomini, R., Granger, C.W.J.: Aggregation of space-time processes. *J. Econom.* **118**, 7–26 (2004)
22. Quinlan, J.R.: Learning with continuous classes. In: *Proceedings of the Australian Joint Conference on Artificial Intelligence*, pp. 343–348. World Scientific, Singapore (1992)
23. Wang, Y., Witten, I.H.: Induction of model trees for predicting continuous classes. Poster papers of the 9th European Conference on Machine Learning. Springer (1997)
24. Breiman, L.: Bagging predictors. *Mach. Learn.* **24**, 123–140 (1996)
25. Schapire, R.E.: A brief introduction to boosting. In: *Proceedings of the 16th International Joint Conference on Artificial Intelligence – vol. 2*, pp. 1401–1406. Morgan Kaufmann Publishers Inc., San Francisco, (1999)

Graph Summarization Using Word Correlation Analysis on Large Set of Documents

Putu Y. Kusmawan¹(✉) and Joonho Kwon²

¹ Department of Electrical and Computer Engineering,
Pusan National University, Busan, South Korea

² Institute of Logistic Information and Technology,
Pusan National University, Busan, South Korea
{putuyuwono, jhkwon}@pusan.ac.kr

Abstract. As there are a lot of available documents in the Internet, it is impossible to manually extract their important information. In this paper, we propose a system for extracting important information automatically from huge volume of documents using word correlation analysis. Our system analyzes words' occurrence and co-occurrence frequencies on several levels: sentence, paragraph, and document. And then, it performs three different analysis steps: occurrence frequency, adjacent correlation, and importance score analysis, to calculate the importance score of each word. Finally, it can extract keywords and store them in a graph structure. The benefits of using a graph structure were twofold. We could effectively manage the keywords and their connections; and it assisted us with the retrieval of relevant documents. Our preliminary experiment shows that our technique can be used for analyzing large set of documents well.

Keywords: Word correlation analysis · Large set of document · Graph summary

1 Introduction

In this digital era, people might want to grab important information from documents quickly, but it is difficult to be done since there are too many documents to read. This problem is often called information overload. For an instance, in digital forensic field, investigators are facing difficulties in finding suspicious relationships among individuals during a crime scene investigation [1] as there are so many documents or files to be analyzed. The similar problem occurs in a domain of online news broadcasting. Because there exist enormous amount of rapidly updated information, consumers do not have much time to read all of them. Thus, several online news aggregators [2–4] help consumers staying in touch with the latest news easily. However, they ignore connections among information.

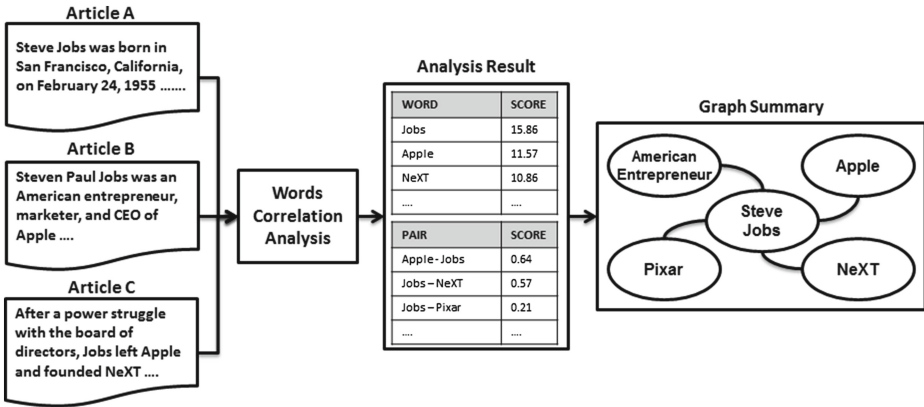


Fig. 1. Example of correlation analysis

Example 1. Let us describe the necessity of extracting information by an example. Assume that there are several articles about Steve Jobs. An article A tells much more about his childhood life, and less about his career. Another article B tells about the time when he founded a company called Apple. The others mention his past activities in several companies such as NeXT and Pixar which he had founded and had worked for. After reading all of those articles one by one, readers might discover that Steve Jobs is very related to the most successful American Entrepreneur.

It will be great, if we can build a system that can analyze the relationships among information scattered across different sources. Figure 1 illustrates the system of graph summarization using word correlation analysis on several articles about Steve Jobs. As we can see in the graph summary, a vertex labeled Steve Jobs is connected to several vertices labeled as Apple, NeXT, and Pixar. This helps us to understand why he is one of the most famous “American Entrepreneur”.

An automatic document analysis has been an interesting topic as several researchers conduct their work [5–7] in this field. However, to our knowledge, none of them try to analyze a large collection of documents. Most of them only consider how their approach can extract keywords or summarize documents well by evaluating their approach on a small number of documents. Wartena and Matsuo [5,6] try to extract keywords from short texts. During an extraction process, they consider connections between words by checking a single level of co-occurrence. However, when the size of document grows very large, analyzing a single level of co-occurrence is not enough since there might be long distance correlation between words. Hu [7] breaks down an input text into different types of levels, and assigns a fixed weighting value for each of them. However, if we analyze large set of documents, the number of sentences and paragraph within documents may vary. This will impact the distribution of word co-occurrence

on each level, thus we should assign a proper weighting value for each level of documents regarding to its portion.

In this paper, we introduce a new technique for extracting meaningful relations among words to produce keywords as well as creating graph summary from large set of documents. Our contributions are summarized as follows:

- We propose a technique to create a graph summary by analyzing word correlation on three different levels: sentence, paragraph, and document.
- We assign a dynamic weighting value to every level by considering its portion. For example, we divide the number of sentences by the total number of levels to obtain the weighting value of sentence level. These weighting values might also be configured based on experimental evaluation.
- We describe the usage of the resulting graph summary for storing important information and retrieving relevant documents.
- We implement our technique in Map-Reduce framework and conduct a preliminary evaluation that shows the performance of our technique in analyzing large set of documents.

The rest of this paper is organized as follows: Sect. 2 presents the overview of our system. Section 3 explains about how we can compute the importance score of words using our technique. Section 4 describes about the process of storing the analysis result into a graph structure and how we can take advantage of the resulting graph structure in retrieving relevant documents. To evaluate the performance of our method, we have conducted a preliminary experiment and describe it in Sect. 5. Finally, we conclude our work in Sect. 6.

2 System Overview

In this section, we shall explain the architectural overview of our system as shown in Fig. 2. Our system consists of five modules: a preprocessing module, three correlation analysis modules implemented by using Map-Reduce framework, and a graph construction module. We briefly explain the functionalities of five modules here and describe the details in the next section.

- **Document Preprocessor (PRE)**: This module takes input documents and creates level maps by parsing them into sentences, paragraphs and documents. It also removes stop words such as the, is, are, at, which, in, and on.
- **Occurrence Frequency Analyzer (OFA)**: This module reads the preprocessed documents and checks the occurrence frequency of each word on each level. The output will be a list of words containing their set of frequencies.
- **Adjacent Correlation Analyzer (ACA)**: This module analyzes the strength of connections among words by using the preprocessed content and the level statistics. The result of this analysis step is a list of pairs with their correlation scores.
- **Importance Score Analyzer (ISA)**: This module combines the result of two previous steps to calculate the importance scores of all words. The result of step is a list of words with their importance scores.

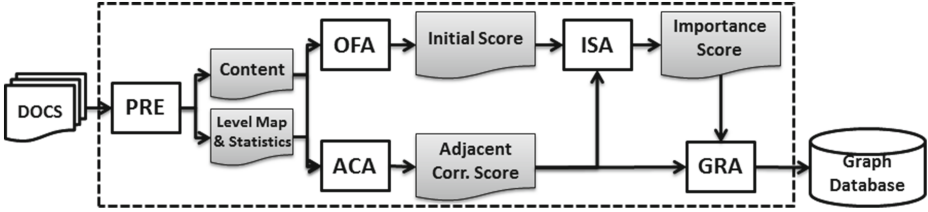


Fig. 2. Architectural overview

- **Graph Constructor (GRA)**: This module takes the analysis results and stores them in a graph structure. The output of ISA will be stored as vertices and the output of ACA will be stored as edges. In addition, we also perform an entity resolution process to identify people, companies, organizations, cities, geographic features, and other types of entities.

3 Importance Score Analysis

In this section, we shall describe the detailed steps of calculating the importance score of all words. We divide and explain all of them in several subsections.

3.1 Document Preprocessing

As there might be a lot of meaningless words and irrelevant characters, we cannot directly perform our analysis steps on the input documents. We should perform several preprocessing steps to prepare the documents before being analyzed. There are several tasks that must be done in this step:

- Extracting sentence from input documents.
- Eliminating stop words.
- Assigning document ID, paragraph ID, and sentence ID (Level Map) to each sentence
- Checking the number of each level (Level Statistics).
- Partitioning documents into several blocks which are equal before loading into HDFS.

The outputs of this step are level maps and statistics as well as preprocessed documents' content. Due to the tiny size of a single document, we concatenate and partition the preprocessed documents into several equal-sized files to fit HDFS block size.

Example 2. For providing a running example, we use two documents (D1 and D2) about Steve Jobs' biography [8, 9] in which each document has four sentences and two paragraphs. Figure 3 illustrates an example of document preprocessing. We use an existing sentence parser [10] to extract sentences and paragraphs. For assigning level map, we use an incremental integer number on each level. By doing so, we can also obtain the level statistics of the input documents.

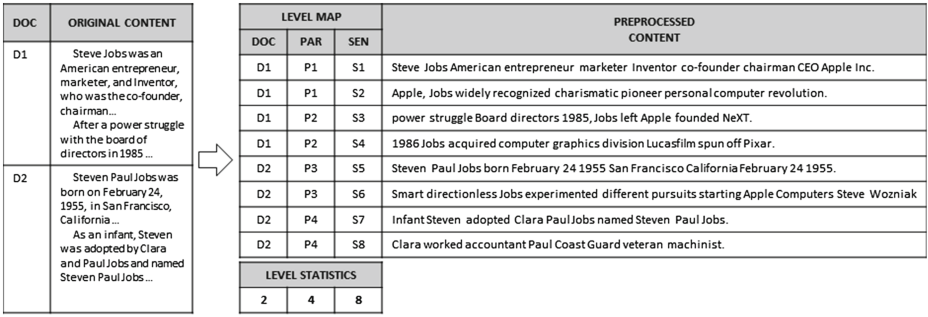


Fig. 3. Example of document preprocessing step

3.2 Occurrence Frequency Analysis

As an initial score, we analyze words occurrence frequency on each level: sentence, paragraph, and document and implement it in Map-Reduce framework. First, mappers read the preprocessed input documents to emit word as key, and level map as value. And then, reducers summarize the frequency of each word regarding its occurrence location. The final output is a list of word including its occurrence frequency on each level and the overall frequency summation. Figure 4 shows an example output of analyzing word occurrence frequency.

3.3 Adjacent Correlation Analysis

Next, we analyze the strength of connection between two words by calculating adjacent correlation scores. To calculate these scores, we need to check the co-occurrence frequency and location of every pair of words. Then, we compute the correlation scores level by level, formalized as Level-specific correlation (*LC*) analysis. For performing this step, we have observed the following characteristics:

1. Important pair of words must appear at least once in the same sentence.
2. Words may not stand alone to deliver information, they must work together to form at least a sentence.
3. After forming a sentence, words can form larger structure called paragraph and document to deliver more complete information.

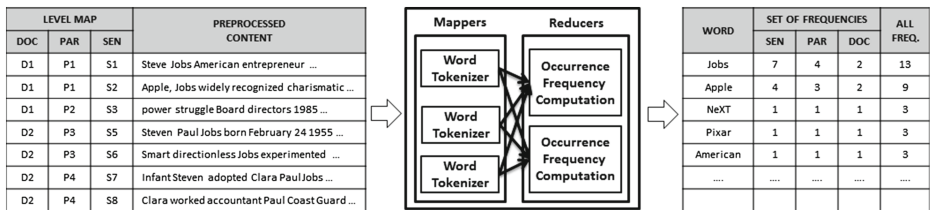


Fig. 4. Example of occurrence frequency analysis

Table 1. Example of calculating adjacent correlation score

Pair	Level-specific Correlation (LC)			Adjacent correlation
	Sentence (N=8)	Paragraph N=4	Document N=2	
Jobs-Apple	Co-occur: 4 LC: 4/8 = 0.50	Co-occur: 3 LC: 3/4 = 0.75	Co-occur: 2 LC: 2/2 = 1	0.64
Jobs-NeXT	Co-occur: 1 LC: 1/8 = 0.125	Co-occur: 1 LC: 1/4 = 0.25	Co-occur: 1 LC: 1/2 = 0.50	0.21
Jobs-Pixar	Co-occur: 1 LC: 1/8 = 0.125	Co-occur: 1 LC: 1/4 = 0.25	Co-occur: 1 LC: 1/2 = 0.50	0.21
Jobs-American	Co-occur: 1 LC: 1/8 = 0.125	Co-occur: 1 LC: 1/4 = 0.25	Co-occur: 1 LC: 1/2 = 0.50	0.21

Example 3. Let us consider again the preprocessed content in Fig. 3. The words, “Jobs” and “Apple”, co-occur in four sentences, and there are eight sentences in the whole input documents. We can calculate sentence-level correlation score as follows: 4/8 = 0.5. We perform the same operation for the upper level of co-occurrence, until we can get three different level-specific correlation scores (sentence, paragraph, and document).

We then summarize all level-specific correlation scores by considering the weighting values on each level as an adjacent correlation (AC) score. The AC score between two words (*a, b*) is formalized as follow:

$$AC_{a,b} = \sum_{x=s}^d \alpha_x \cdot LC_{a,b}^x \tag{1}$$

where $LC_{a,b}^x$ is the level-specific correlation of word a and b, α_x is the weighting value for each level, and x is a set of levels containing sentence(s), paragraph(p), and document(d). To give a proper weighting value, we use the level statistics obtained from the document preprocessing step.

Example 4. Consider again the level statistics in Fig. 3. There are 8 sentences, 4 paragraphs, and 2 documents which produces 14 total levels of our dataset.

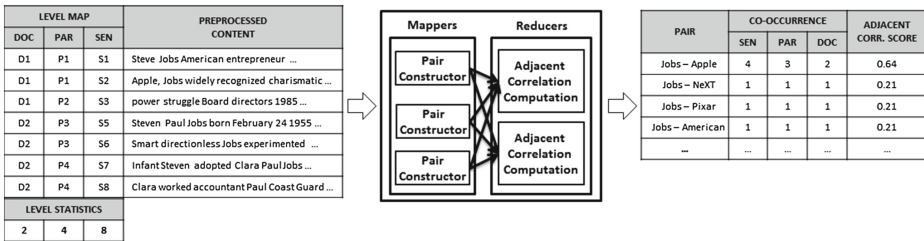


Fig. 5. Overview of adjacent correlation analysis in map-reduce

We will divide the number of corresponding level by the total number of levels in the whole documents. Therefore, we can calculate sentence level weighting value as follows: $8/14 = 0.57$. And then, we can calculate the adjacent correlation score for “Jobs-Apple”: $(0.57 \times 0.50) + (0.29 \times 0.75) + (0.14 \times 1) = 0.64$. Table 1 shows the example of calculating adjacent correlation scores.

The higher adjacent correlation score means the stronger connections between two words. As we can see in Table 1, the pair “Jobs-Apple” has higher score than the pairs of “Jobs-NeXT” and “Jobs-Pixar”. Thus, we can conclude that based on our dataset “Jobs” is more related to a company named “Apple” than “NeXT” or “Pixar”. In addition, we can also use this value to prevent storing irrelevant pairs.

We also have implemented this analysis step in Map-Reduce to make it scalable. Mappers will read the preprocessed content as well as the level map to construct all pairs. After map task has finished, it will emit the pair of two words (separated by comma) as key and level map as value. Reducers will retrieve the intermediate result and check the level map to calculate word co-occurrence on each level. To compute the adjacent correlation score, they utilize the level statistics that has been loaded into Hadoop distributed cache before the job starts. As the result, reducers will emit the pair as key and its adjacent correlation score as value. Figure 5 shows the processing flow to calculate the adjacent correlation scores in Map-Reduce framework.

3.4 Importance Score Analysis

After performing occurrence frequency and adjacent correlation analysis, we can calculate the importance score of each word. There will be two mapper classes for reading two different data sources which are the result of the previous analysis steps, OFA and ACA. The first mapper class will read the result of OFA, then emit each word as key and its occurrence frequency as value with a character ‘F’ as prefix. The second mapper class will read each pair of words from ACA’s output and split them into words. Then, it emits each word as key and emit its adjacent correlation score as well as its co-occurring word as value.

After map tasks have finished, reducers will begin their work by collecting all words occurrence frequencies and adjacent correlation scores. Prior to calculating the importance score, they will perform one additional step called transitive correlation analysis. This step is needed to measure the impact score that is shared between two co-occurring words. The impact must be proportional to the adjacent correlation score and the occurrence frequency difference between them. The bigger transitive correlation score a pair has, the bigger supporting score they share.

Example 5. Recall to the result of the two previous steps in Fig. 4 and Table 1, we can obtain the overall frequency difference between “Jobs” and “Apple” as follow $13 - 9 = 4$, and we have also computed the adjacent correlation score of “Jobs-Apple” which is 0.64. Thus, we can calculate the transitive correlation score of “Jobs-Apple” as follow $4 \times 0.64 = 2.57$.

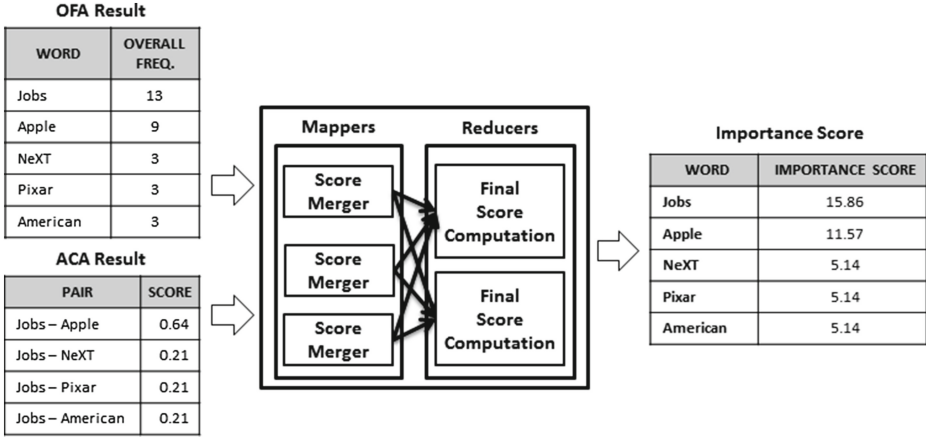


Fig. 6. Overview of importance score analysis in map-reduce

After calculating the transitive correlation of all pairs, reducers will calculate the importance score by adding the overall occurrence frequency of each word with the maximum transitive score (MAX(tra)) that it has. Figure 6 shows the Map-Reduce processing flow to calculate the importance scores.

Example 6. Consider again the result of OFA in Fig. 4, the overall occurrence frequency of “Jobs” is 13, and it is connected to several vertices via four different edges representing their transitive correlation scores. Thus, we can calculate the importance score of “Jobs” by adding its frequency to its maximum Transitive Correlation score: $13 + \text{MAX}(2.57, 2.14, 2.14, 2.14) = 15.57$.

As the result of this step, reducers will emit word as key and its importance score as value. The higher importance score, the more likely we can regard a word as a good candidate for keyword. To make the process easier to understand, we describe this mechanism using a graph structure in Fig. 7.

4 Graph Structure

4.1 Graph Structure Construction

After performing all of the analysis steps, we construct a graph structure using the analysis results. As we know, a graph mainly consists of two main components: vertex and edge. Both vertex and edge may have several properties to store more detailed information. Figure 8 depicts the resulting graph structure that can be built by our system. There are 5 vertices in the graph, each of them has several properties. For an instance, a vertex labeled “Steve Jobs” stores the importance score, occurrence frequency and location. The edges also have several properties to store detailed information of all pairs of words.

We use the result of ISA and ACA to construct the graph structure. First we read all of the ISA’s output and store them as vertices. And then, we create

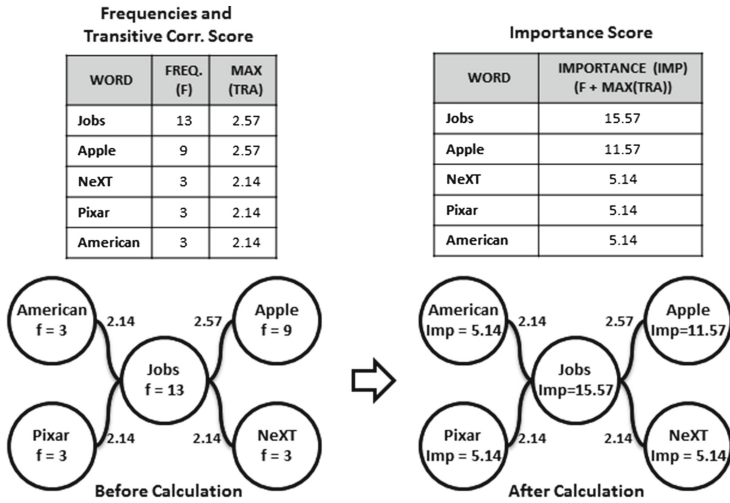


Fig. 7. Example of importance score calculation

edges using the output of ACA. After all vertices and edges have been stored in the graph, we perform an entity resolution using an existing NER library [11] to obtain the complete term form each vertex label.

4.2 The Usability of Graph Structure

In this subsection, we will discuss about the usage of the resulting graph structure produced by our system. By using the graph structure, we can manage all keywords and their connections well. In addition, it might also be useful for relevant document retrieval system.

Example 7. For this occasion, we will use the example documents that we have mentioned in Fig. 3. Suppose user wants to retrieve all documents which contain any information about “Jobs” and “Apple”. If we represent keywords as a bag-of-words, we might be able to return all documents which contain both “Jobs” and

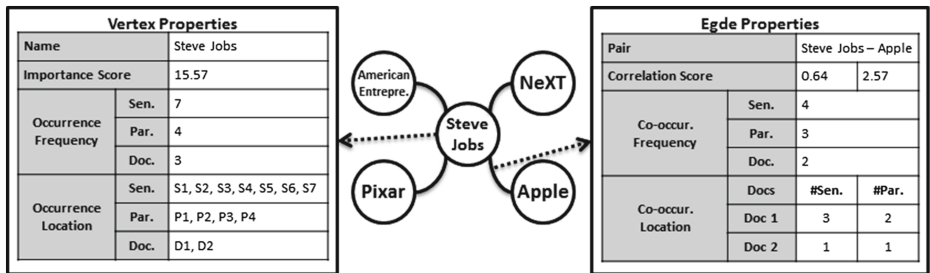


Fig. 8. Example graph structure

“Apple”. However, such kind of representation does not store the connectivity between keywords. It will be difficult for us to rank the query answer based on the relevance of document’s content. Finally, user must decide manually which document to read first. Using our approach, we can tackle this issue by analyzing the connection between “Jobs” and “Apple” in more detail. Even though they co-occur in both documents, the co-occurrence distribution in lower level is different. Doc.1 has more sentences mentioning about “Jobs” and “Apple” than Doc.2 does, thus we should recommend user to read Doc.1 prior to reading Doc.2. Thus, our system can quickly answer user’s query using the following steps:

1. Locate “Jobs” vertex.
2. Find an edge that connects “Jobs” vertex with “Apple” vertex.
3. Check in which document they co-occur by analyzing the corresponding edge’s properties.
4. Rank the result according to the number of sentence and paragraph co-occurrence on each document.

Finally, our system will returns Doc.1 at the top of the query answer.

We may also employ indexing system on the edge structure to improve the query performance. However such improvement approach will have a trade-off since it will also increase the complexity of graph construction.

5 Evaluation

In this section, we present experimental result to evaluate the accuracy and the performance of our system.

5.1 Experimental Environments

Hardware. We used two different hardware setups for evaluating our system. First, we used a single machine (Intel Core2Quad Q6600 @2.4 GHz, 64-bit, 4 GB RAM) running on Windows Server 2008 R2 Standard 64-bit for evaluating the accuracy of our system. Second, we used 15 commodity machines (Intel Core2Quad @2.66 GHz, 64-bit, 2 GB RAM, 500 GB HDD) running on Ubuntu 12.10 and configured them to work together on top of Hadoop version 1.2.1 for evaluating the performance of our system.

Dataset. We used two data sets: (1) IEEE dataset for an accuracy test and (2) digital books [12] for a performance test. The IEEE dataset consists of several papers which are randomly selected from IEEE Explore website. The overall statistics of our first dataset are as follows: 90,990 words, 20,511 sentences, 15,495 paragraphs, and 20 documents. The digital book dataset consists of 9,487,087 sentences, 2,873,700 paragraphs, and 2,620 documents. In total, the size of the digital book dataset is 1.1 GB with the average size of 2.7 MB.

5.2 Accuracy Test

In this evaluation, we tried to extract several important information in the form of keywords from several scientific paper documents. We combined the author’s keyword and manually-selected keyword as the basis of forming relevant keyword list for each paper. Prior to this experiment, we broke down relevant keywords into words and eliminated duplicate words. As the result, each paper has at least 5 distinct words and at most 18 distinct words, with an average of 10 distinct words. Then, we extract keywords using our method and compute the precision-recall average score. As we know, precision and recall formula are defined as follows:

$$Precision(P) = \frac{CorrectWord}{ExtractedWord}; Recall(R) = \frac{CorrectWord}{RelevantKeyword} \quad (2)$$

For the first attempt, we want to extract as many correct keyword as possible; thus, we select Top-25 most important words from all documents. In this case, we manage to get a good recall score ($R = 0.73$), however the precision score is very low ($P = 0.29$). It makes sense since there are only 10 keywords on each paper in average. Therefore, we gradually decrease the number of extracted keyword for each document (Top-20,15,10,5) to achieve better precision score. Finally, we extract Top-5 most important words from all documents and we get $R = 0.32$ and $P = 0.63$. Figure 9 shows the result of our accuracy evaluation.

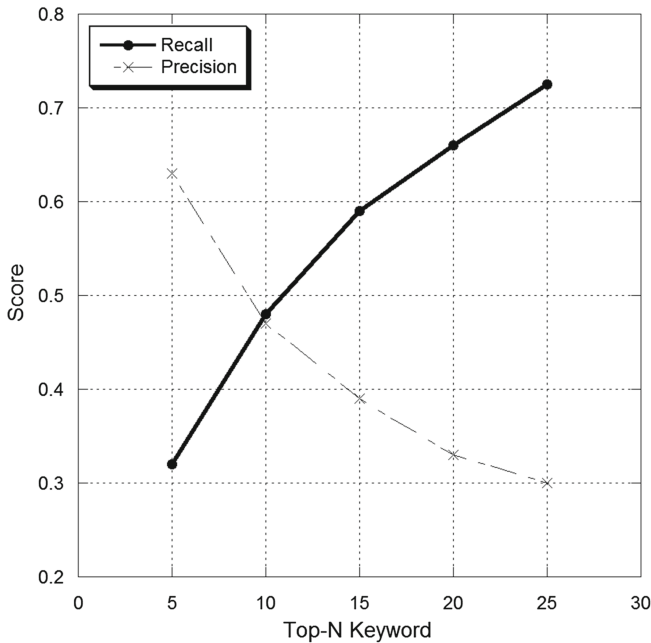


Fig. 9. Precision and recall result

5.3 Performance Test

To measure the performance of our technique, we have also conducted another experiment using the digital book dataset. We used 15 machines to process varying size of the datasets. Since the average size of a single book is only 2.7 MB, we concatenated the books to construct 50 MB input split size before running the Map-Reduce job. We used a single machine to preprocess the dataset, thus it takes several minutes to completely preprocess the dataset. Figure 10(a) shows the time needed for preprocessing the dataset(PRE) and for loading it into HDFS(LOAD).

During the first execution, our system requires around 10 min to completely analyze 100 MB of documents. When we gradually increase the dataset size up to 400 MB, our system requires considerably few more seconds to analyze them all. We get a significant performance degradation, when it comes to analyze 800 MB of documents. However, we can say that our system only requires less than a second to analyze a single book since it can finish processing 1 GB dataset in 37 min.

As shown in Fig. 10(b), the most time consuming step is adjacent correlation analysis (ACA). It consumes more than a half of the whole processing time. This is due to the very large amount of pairs that needs to be processed. For an instance, it calculates ACA score of 27 millions of pairs during the analysis of 1GB dataset. Our future works will focus on improving our technique especially in this step. It will be much faster, if we can eliminate some irrelevant pairs prior to this step.

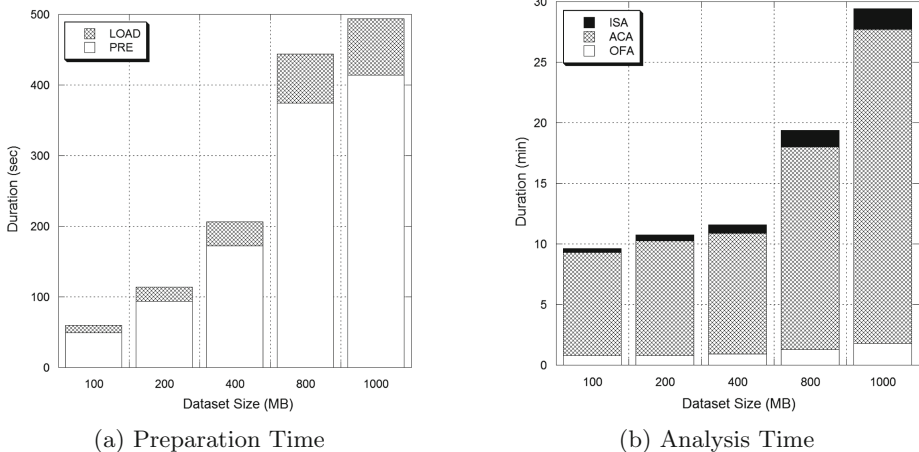


Fig. 10. Execution time

6 Conclusion

In this paper, we have proposed a system for summarizing a large set of document as a graph structure using word correlation analysis. Our system relies on the analysis of words occurrence, and co-occurrence statistics on each level of documents. First, it performs the occurrence frequency analysis to calculate word's initial importance score. Second, it analyzes the strength of connection among words by performing the adjacent correlation analysis. The adjacent correlation score is useful for calculating the impact score that a word gives or receives during the transitive correlation analysis. Then, our system performs importance score analysis to calculate the final score combining the initial importance score and the maximum transitive correlation score. Finally, a graph structure can be constructed from the importance scores and correlation scores. We have also described the usability of our graph structure for storing important information and retrieving relevant documents. Experimental results have shown that our system can give a considerably accurate result and perform well in analyzing large amount of documents.

Acknowledgement. This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program (NIPA-2013-(H0301-13-1012)) supervised by the NIPA(National IT Industry Promotion Agency).

References

1. Carrier, B.D.: Defining digital forensic examination and analysis tool Using abstraction layers. *Int. J. Digital Evidence* **1**(4) (2003). <http://www.utica.edu/academic/institutes/ecii/publications/articles/A04C3F91-AFBB-FC13-4A2E0F13203BA980.pdf>, DBLP, <http://dblp.uni-trier.de>
2. Chowdhury, S., Landoni, M.: News aggregator services: user expectations and experience. *Online Inf. Rev.* **30**(2), 100–115 (2006)
3. Summly: Summly news aggregator (2014). <http://summly.com/>
4. Inc., G.: Google news (2014). <https://news.google.com/>
5. Wartena, C., Brussee, R., Slakhorst, W.: Keyword extraction using word co-occurrence. In: *Proceedings of Seventh International Workshop on Text-based Information Retrieval*, Bilbao, Spain, pp. 54–58 (2010)
6. Matsuo, Y., Ishizuka, M.: Keyword extraction from a single document using word co-occurrence statistical information. *Int. J. Artif. Intell. Tools* **13**(1), 157–169 (2004)
7. Hu, X., Wu, B.: Automatic keyword extraction using linguistic features. In: *Proceedings of 6th ICDM Workshops*, pp. 19–23 (2006)
8. Wikipedia: Steve jobs (2013). <http://en.wikipedia.org/wiki/SteveJobs>
9. Website, T.B.C.: Steve jobs (2014). <http://www.biography.com/people/steve-jobs-9354805>
10. Group, T.S.N.: Stanford corenlp (2013). <http://nlp.stanford.edu/software/corenlp.shtml>

11. AlchemyAPI: Entity extraction api (2013). <http://www.alchemyapi.com/api/entity-extraction/>
12. Project Gutenberg Organization: Free ebooks - Project Gutenberg (2013). <http://www.gutenberg.org/dirs/>

Distributed K-Distance Indexing Approach for Efficient Shortest Path Discovery on Large Graphs

Jihye Hong, Hyunwook Kim, Waqas Nawaz, Kisung Park,
Byeong-Soo Jeong, and Young-Koo Lee^(✉)

Department of Computer Engineering, Kyung Hee University,
Seocheon-Dong, Giheung-Gu, Yongin-Si 449-701, Korea
{hjhh, hwook956, wicky786, kspark, jeong, yklee}@khu.ac.kr

Abstract. The emergence of large real life networks such as social networks, web page links, and traffic networks exhibits complex graph structures with millions of vertices and edges. Among many operations for exploiting these graphs, the shortest path discovery is a major and expensive one. Besides the in-memory approaches, many efficient shortest path computation methods have been developed on top of distributed and parallel platforms. Pregel, a bulk synchronous parallel framework, is one of them for processing large graphs. The known shortest path computation approach with Pregel is computation intensive and unable to target real-time services. In this paper, we propose a Pregel based efficient k-distance index technique that allows efficient single pair shortest path discovery. We reduce the network cost and unnecessary operations by transmitting more information in a single superstep. The extensive experiments on both real and synthetic datasets reveal the superiority of the proposed approach.

Keywords: Shortest path discovery · Large graph · Graph indexing · Pregel

1 Introduction

Having millions of vertices and edges, real life networks such as social networks, web page links, and traffic networks are very large. Many applications have been developed on top of these huge and complex networks. They include social network services and navigation systems. The shortest path discovery is one of the most important search queries to support graph services. For instance, a distance between two users represents the closeness in social networks, and is used in a social search to find related contents [1] or to analyze features of influential people and communities [2]. Moreover, the relevance among web pages is determined through a shortest path distance approach to facilitate and suggest the related contents for search queries [3, 4].

The most emerging real-time services using a graph structure, e.g. context aware search, based on large graphs, require an efficient shortest path discovery approach. To improve the performance of the shortest path discovery, various in memory-based efficient algorithms are proposed such as Dijkstra's algorithm [5] and Bi-directional

search strategy [6] which can reduce the search space by running two directions simultaneously. Moreover, Indexing techniques [7–11] have also been introduced to improve the performance in running time by pre-computing the shortest paths. These approaches are not fast enough to support real-time services on very large graphs due to huge cost on preprocessing.

Recently, many scalable efficient techniques have been proposed for large graphs which can be categorized into single PC based partitioning, map-reduce and vertex centric parallel approach. Yuan et. al. [12] has introduced a disk based large graph partitioning approach on a single PC. It partitions the graph into segments of memory size which are sequentially processed to discover the shortest path. Map-Reduce based distributed frameworks [13, 14] have been studied in literature to analyze big data. However, the Map-Reduce paradigm causes expensive I/O costs for graph algorithms due to its complex structure which requires more iterations. There are many algorithms which require an iterative vertex-centric logic such as a single source shortest path and a subgraph isomorphism. To overcome disadvantages for processing large complex data, BSP (bulk synchronous parallel) framework [15] is proposed. Google’s Pregel framework [16] is a representative BSP framework. Pregel’s computational model is based on an iterative program representation with two distinct entities that abstracts distribution related details behind a user API. Existing graph algorithms with Pregel are still not fast enough.

A pre-computed indexing approach [17] which stores partial shortest path is well-known approach for supporting shortest path discovery efficiently over large graphs. However, though all-pairs shortest path (APSP) algorithm with Pregel has been proposed, this approach is also too slow since this performs Dijkstra’s algorithm N times, where N is the number of vertices. The complexity is close to $O(N^2 \log N)$. Therefore, there are no indexing techniques can process in reasonable time based on Pregel framework for a single pair shortest path discovery until now.

In this paper, we propose a Pregel based efficient k -distance index technique for supporting s - t shortest path discovery efficiently. Our contributions are as follows.

- **Efficient k -distance index approach based on Pregel framework** we reduce the number of supersteps by passing messages including more information that represent partial paths and distances from other vertices in each superstep. Workers should perform additional operations for synchronizing all the workers when the superstep starts or ends. Therefore, the number of supersteps affects to the performance for achieving the shortest path results. The proposed approach only requires the maximal number of supersteps among the number of supersteps from vertices.
- **Efficient s - t shortest path discovery based on Pregel framework** we propose an indexing approach for supporting efficient s - t shortest path discovery without modifying the input data. After merging the index with original data, we can reach the destination vertex by expanding longer distance than the original edges in each superstep.

2 Related Work

In this section, existing methods for the shortest path computation are briefly explained with a inherent limitation for large graphs. Shortest path discovery is fundamental and important in graph applications. Conventional algorithms-Dijkstra's algorithm [5] and a bi-directional search strategy [6], operate only on memory resident graphs. The bi-directional approach reduces the search space by running forward and backward searches simultaneously. The indexing techniques [7-9] also improve the performance in running time by pre-computing the shortest path. However these methods are not scalable for large graphs due to limited memory.

The subsequent discussion explain the recent scalable approaches for shortest path computation on very large graphs using single PC, distributed, and parallel frameworks.

Graph frameworks based on a single computing have been developed such as LEDA [18] and iGraph [19]. These frameworks load the entire graphs on a memory, or partial graph can fit into the memory through disk I/Os. In addition, a relational database based graph processing approach is one of the well-known approaches. HDB-SUBDUE [20] and DB-FSG [21] proposed a RDB based frequent subgraph mining method. Gao et al. [17] proposed a generic Frontier-Expansion-Merge (FEM) framework for graph search operations in RDB context, and implemented the shortest path discovery on the framework. In order to improve the performance, the FEM framework uses an index table that stores pre-computed partial paths. However, single computing approaches are still not fast enough for supporting real-time services.

In contrast to single PC, parallel computing frameworks for processing graphs use multi-threads or a number of processors, and improve the performance such as Parallel BGL [22] and CGMgraph [23]. Moreover, distributed computing frameworks have also been actively used to achieve performance gain. In particular, Map-Reduce framework [13] that stores large graphs in the distributed file system over a cluster of computers and processes them in parallel. However, accessing graphs is difficult because Map-Reduce framework does not fully support schema and index mechanism [13, 14], and incurs unnecessary disk I/Os and network costs.

Recently, vertex-centric frameworks which distribute the graph data, and perform the jobs parallel have been developed. Google's Pregel framework [24] is one of the representative vertex-centric frameworks. The computations are expressed as a sequence of iterations, where a vertex can receive messages that are sent in the previous iterations. The vertex can send messages to other vertices, and modify its own state on its outgoing edges or mutate graph topology. Some algorithms with Pregel have been proposed such as a page rank, a single source shortest path algorithm and an all-pairs shortest path (APSP) discovery algorithm [25]. However, these naïve methods still require optimization for efficiency. For example, the all-pairs shortest path approach performs Dijkstra's algorithm N times, where N is the number of vertices. Therefore, indexing mechanism can improve the performance of APSP algorithm using pre-computed information by reducing the computation overhead.

3 Preliminary

3.1 Pregel Framework

Pregel is a graph processing system which supports a vertex-centric parallel function and partitions a graph into many machines. This framework is made up of a cluster of machines. One of these machines acts like a server called master. Master is not assigned any portion of the graph, however coordinates activities of remaining machines called workers.

The master maintains a list of all workers currently known to be alive, including the worker's unique identifier, address information and portion of the graph assigned to each worker. Each worker maintains the state of its section of the graph, and executes the user function on all contained vertices, and managing messages to and from other workers.

Each worker performs a superstep that loops through all vertices and calls Compute() function, passing it the current value, an iterator to the incoming messages, and an iterator to the outgoing edges. Workers perform following 3-steps iteratively.

- Each vertex v processes all messages received by other vertices from previous superstep.
- Each vertex v sends new messages to all connected vertices of the graph, or decides to halt.
- There is a bulk synchronization process which makes sure all the messages get to their final destination.

3.2 Single Source Shortest Path Algorithm with Pregel

The single source shortest paths problem requires finding a shortest path between a single source vertex and every other vertex in the graph.

In the single source shortest path algorithm with Pregel, the value of all vertices is initialized to infinite value. In each superstep, each vertex first receives messages from its neighbor including updated potential minimum distances from the source vertex. If the minimum of these messages is less than the value currently associated with the vertex, then this vertex updates as minimum value and sends out messages including potential updates to its neighbors, consisting of the weight of each outgoing edge added to the updated minimum distance. In the first superstep, only the source vertex will update its value from infinite value to zero, and send updates to its immediate neighbors. These neighbors in turn will update their values and send messages. After the value associated with each vertex denotes the minimum distance from the source vertex to own vertex, the algorithm terminates when no more updates occur. Termination is guaranteed if all edge weights are non-negative.

3.3 Distance Based Index Table for Shortest Path

Graph searching is widely used for graph algorithms finding specific subgraphs such as the shortest path and the graph reachability. Most of graph searching algorithms

share a generic search process that iteratively extends nodes having results of query with high possibilities.

Shortest path searching generally adopts a breadth first search (BFS) to traverse a graph. A BFS can only reduce the search space in the case that a shortest path has a small number of nodes. A large-scale graph must have a long shortest path. Therefore, a BFS requires a large number of iterative expansions in for huge graphs.

For efficient implementing the generic search process, the FEM framework [17] proposed three operators. The FEM framework also requires BFS to expand all of edges of frontier nodes. For the efficient searching, the FEM framework pre-computes shortest segments with their distances shorter than the given distance threshold and stores the shortest segments into an index table called SegTable. Two kinds of index tables are maintained such as ToutSegs and TInSegs since the FEM framework performs bi-directional expansion. The index tables consist of source nodes (fid), target nodes (tid), parent nodes of target nodes (pid), distances of the shortest segments (cost). By expanding shortest segments in the index table, we can reduce unnecessary re-expanding for the path segments contained in the shortest segments. Therefore, we can meet the termination condition of the searching quickly.

Figure 1 shows an example of the index table. The index table is the relational table containing shortest segments of out-edges from the original graph. The graph with shortest segments is a graph representing all shortest segments as dotted-paths. If the distance threshold l_{thd} is set to 4, all shortest segments having distances shorter than 4 are stored into the index table. The dotted-path $a \rightarrow d$ with cost 3 is a pre-computed shortest segment. If we start path searching from a , d can be found in one expansion instead of two expansions such as $a \rightarrow b \rightarrow d$.

4 K-Distance Index Table Construction with Pregel for Efficient Shortest Path Discovery

4.1 Notations

We define the notations related to k -distance index table construction with Pregel.

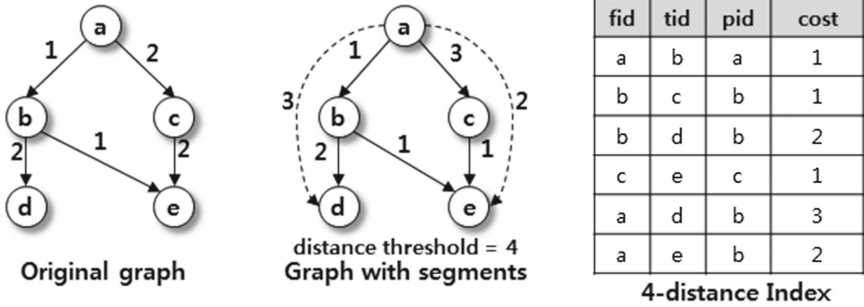


Fig. 1. Example of index table

Definition 1. Shortest Segment. Given the shortest path $p = u \rightarrow \dots \rightarrow v$, we call an edge $e(u, v)$ a shortest segment, and the cost of edge $e(u, v)$ is the distance of shortest path p . We denote the shortest segment from vertex u to vertex v as $e_{seg}(u, v)$.

Definition 2. k -distance index graph. Given the graph $G = |V, E|$, we call a graph G' a k -distance index graph if the graph G' has all the shortest segments which have the distance below k . We denote the k -distance index graph as $G_k = |V, E \cup E_k|$, where E_k is the shortest segment edge set which have the distance below k .

Definition 3. Local shortest distance message $M_{ij}(s, dist)$. Given the path $p = s \rightarrow \dots \rightarrow i \rightarrow j$, we call a message having the distance from vertex s , and transmitted from vertex i to vertex j local shortest distance message. We denote the local shortest distance message as $M_{ij}(s, dist)$, where s is the identifier of start vertex, i is the sender's identifier, j is the receiver's identifier, and $dist$ is the distance from s to j .

4.2 k -Distance Index Table Construction

To search s - t shortest path, each vertex iteratively transmits messages including the distance between start and current vertices. Since this approach move only one vertex in every superstep, so it requires many supersteps. In worst case, the number of superstep is equivalent as the number of vertices.

Each worker maintains the distance value from source vertex to a specific vertex that is assigned to this worker. The distance value is decided through messages that occurred from vertices on the shortest path. In this process, two kinds of operations are carried out: (a) a message processing operation in each worker, (b) a message sending among workers in the network. Each vertex first processes received messages, and generates new messages if there exist any update. After generating new messages, the vertex sends messages to adjacent vertices. The message passing mechanism is very expensive in terms of network communication. Therefore, we need to reduce the number of messages and supersteps to speed up the shortest path discovery process.

The k -distance index approach can help to reduce the search space from the source s to the target t (s - t shortest path) by expanding longer distance at once. Since the all pairs shortest path discovery requires large space and time, it only stores the partial shortest path having the distance below k . Figure 2 shows the shortest path search tree when we search the shortest path from vertex 1 to vertex 6. The depth of this tree without the index table is 3. However, the depth of that with index decreases as 2 since the shortest segment can reach to farther vertex than the original edge at once.

The k -distance index approach has in common features with the all pair shortest path approach that considers a distance constraint. Therefore, we need an efficient all pair shortest path discovery algorithm. We can construct a k -distance index by searching all pair shortest path that have the less distance than k . However, To the best of our knowledge, the existing all pair shortest path algorithm with Pregel is inefficient since this approach repeats the single source shortest path algorithm N times where N is the number of vertices.

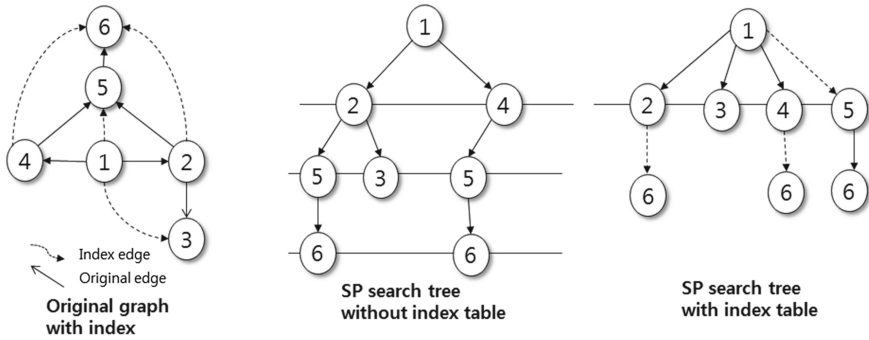


Fig. 2. Shortest path discovery using index table

We propose a 3-step approach for generating k -distance index graph. The overall process used to generate the index graph is shown in Fig. 3. First, we preprocess the input graph dataset for assigning partial graphs to workers. Second, each vertex performs a compute function that updates the distance between this vertex and adjacent vertices. For generating the k -distance index graph, we merge the result of compute function into the original dataset. Finally, we can perform the s - t shortest path discovery efficiently through the k -distance index graph. To determine the shortest distance, the compute function performs following 4-steps iteratively.

- **Compare distances** Compare the distance in local shortest distance message with that in existing shortest segment.
- **Update index table** Update the distance of shortest segment when we find the shorter distance in received messages.
- **Send messages to adjacency vertices** Send messages which include the local shortest distance messages to adjacency vertices when the shortest segments that have the target as own vertex has been updated as shorter distance.
- **Check states of vertices** The vertex state will be changed the active state into the inactive state when any messages do not be applied to index. Compute function will be terminated.

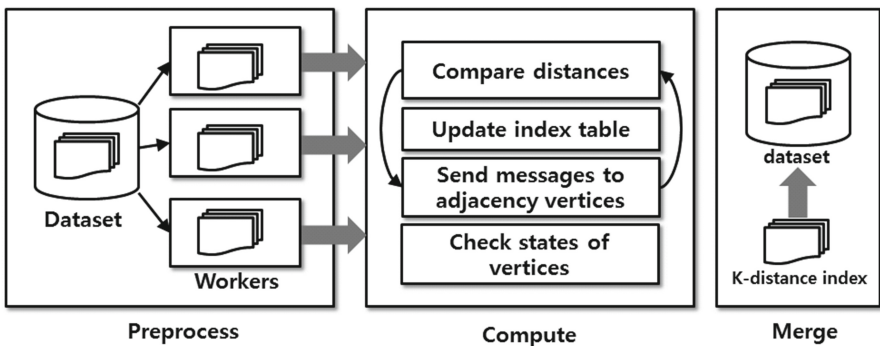


Fig. 3. The overall process of the k -distance index table construction

An example showing how the messages will be transmitted according to supersteps in Fig. 4. In the initializing step, all vertices send the messages that have direct distance to adjacent vertices. For example, vertex 3 sends the message $M_{32}(3, 9)$, a message that has the distance 9 from vertex 3 to vertex 2, to vertex 2 in superstep 1. In the second superstep, we only transmit the updated messages to adjacent vertices. For example, the local shortest distance message $M_{31}(3, 3)$ in vertex 1 will transmit as $M_{12}(3, 4)$ to vertex 2. The distance of message can be calculated as summation of the shortest segment distance and edge weight between vertex 1 and 2. After finishing the superstep 2, we can get the shortest path $p = 3 \rightarrow 1 \rightarrow 2$. The distance of this path is 4. We provide a result of searching for 6-distance index graph in Fig. 5. After terminating the compute function, shortest segments will be generated such as $p_{seg} = 3 \rightarrow 1 \rightarrow 2$ and $p'_{seg} = 1 \rightarrow 3 \rightarrow 4$. All the shortest segments that have the less distance than 6 are generated.

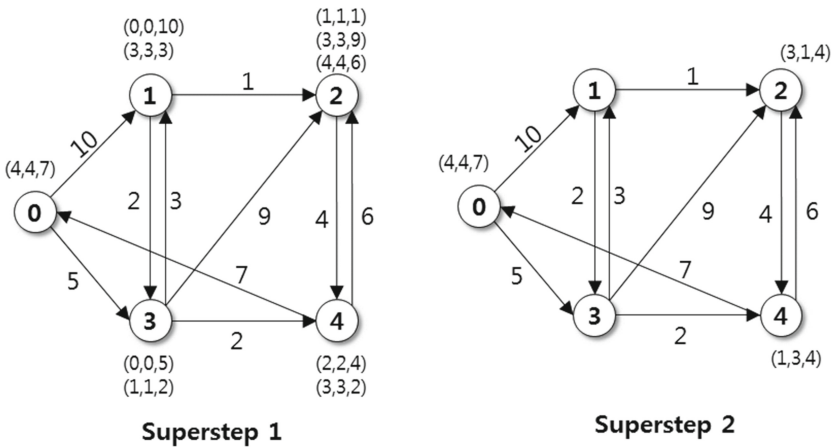


Fig. 4. An example of message passing in superstep 1, 2

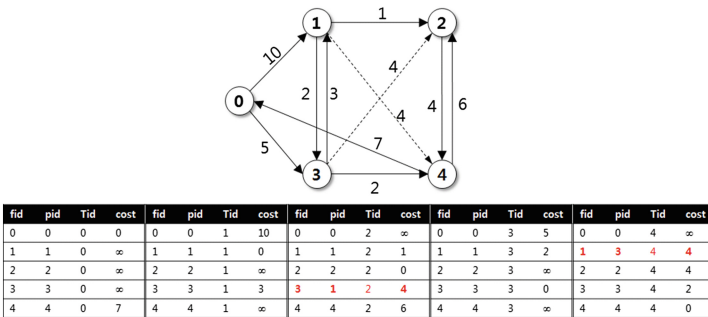


Fig. 5. An example of 6-distance index

We illustrate the pseudo code for searching k -distance index graph in Algorithm 1. In the initializing step, all vertices send the weight of edge to adjacent vertices. After completing to initialize, we check whether the received messages from adjacent vertices should be updated or not, and send newly generated messages to adjacent vertices. Received messages are processed as follows. First, we check whether the distance in each message is larger than k (lines 2–3). If the minimum distance of messages is shorter than existing distance, update the distance of shortest segment (lines 7–10). If shortest segment that have same source identifier does not exists, insert the new shortest segment into the k -index (lines 4–5). Transmitting the messages step is as follows. Each vertex send messages when the summation the edge weight and updated distance of shortest segment is less than k (lines 10–13).

Algorithm 1. Compute function for constructing k -distance index

Algorithm 1. Compute_function_k-dist

- Input: Messages set M
- Output: $S_{out} = \{s | \forall e_{seg} \in E_k\}$

```

1.  FOR each message  $m$  in  $M(v, p, dist)$ 
2.    IF  $dist > l_{thd}$ 
3.      RETURN;
4.    IF there is no shortest segment  $e_{seg}(v_0, v)$ 
5.      Push  $e_{seg}$  into the index
6.    ELSE IF  $e_{seg}(v_0, v)$  exists and  $dist \leq cost(e_{seg})$ 
7.      Update the distance of shortest segment to  $dis$ 
8.      where  $vid=v_0$  and  $pid=v$ 
9.    FOR each edge  $e$  which is connected with  $v_0$ 
10.   FOR each shortest segment  $s$  which is updated in
11.     this superstep
12.   IF  $dist+cost(v_0, e) < l_{thd}$ 
13.     sendMessage( $v_0, v, dist+cost(v_0, e)$ )
14.   RETURN;
15. RETURN;

```

5 Experiments

5.1 Experimental Setup

We implement all methods in Java with JDK 1.7. We use 5 nodes as our distributed system for our experimental environment. Each machine has 7 GB of RAM. The cluster was using Apache Hadoop 1.2.1 [26] on Ubuntu 12.04. All experiments were conducted by Amazon EC2 [27].

We use four kinds of a real-world graph data set including the wikipedia talk network dataset [28], Skitter dataset [29], citation of patent dataset [30] and road network dataset [28], and five synthetic graphs generated by synthetic graph generator

Table 1. Description of data sets

DataSet	# Nodes	# Edges	Weight range
Wikipedia talk network	2,394,385	5,021,410	1
Skitter	1,696,415	11,095,298	1
Citation of patent	3,774,768	16518948	1
Road network	1,965,206	2,766,607	1
Synthetic dataset (R011)	10,000	1,000,000	[1, 9]
Synthetic dataset (R012)	10,000	2,000,000	[1, 9]
Synthetic dataset (R013)	10,000	3,000,000	[1, 9]
Synthetic dataset (R021)	20,000	1,000,000	[1, 9]
Synthetic dataset (R031)	30,000	1,000,000	[1, 9]

[30]. The weights of edges in all graphs are assigned [1, 9] using uniform distribution. Some statistics of these graphs are summarized in Table 1.

In order to show the efficiency of our method, we conduct the following experiments over a commercial database system. For searching shortest paths, we use the single directional set Dijkstra’s approach.

- Comparison of the average time cost between the proposed method (k -DIST) and a naïve approach (k -DijkstraN) that search all the shortest segments having distance below k .
- Analysis of the time cost of the proposed method according to the distance threshold.
- Analysis of the time cost of the proposed method according to the size of graph.
- Comparison of the average time cost for processing SSSP between the original graphs and a k -index graphs.
- Analysis of the computation time according to the number of tasks.

5.2 Performance Evaluation

In the first experiment, we compared the time cost of the proposed method (k -DIST) with that of a naïve approach (k -DijkstraN) that search all the shortest segments having less distance than k . We perform Dijkstra’s algorithm N times, where N is the number of 1 % vertices and calculate the average cost of the single source shortest path algorithm for one vertex due to the number of iterations and time cost increases so much that the execution exceeds the limitations. However, we can estimate the total cost of k -DijkstraN algorithm when we suppose the time cost of each vertices is average. As shown in Fig. 6, the proposed approach required about 60–80 s, while the average time cost of k -Dijkstra is similar to proposed one although the k -Dijkstra performs single source shortest path algorithms for one vertex.

We observed that the proposed method had a high time efficiency because the less number of supersteps are required. In the worst case scenario, the number of supersteps in proposed approach is same as the maximal number of vertices in the graph.

In the second experiment, we execute the proposed algorithm for distance thresholds of 5, 10, and 15 for all the datasets. We show the time cost for constructing

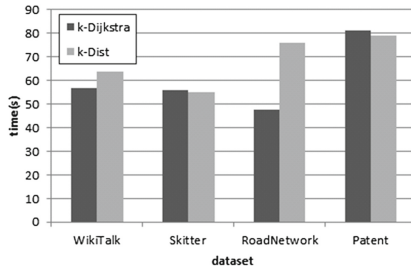


Fig. 6. Comparisons of average time costs

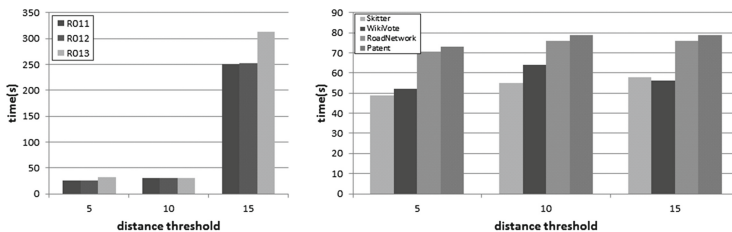


Fig. 7. Comparisons of time costs according to the distance threshold

k -index according to the distance threshold in Fig. 7. The distance threshold affects the size of the index table. The time cost increases according to the distance threshold in most datasets. We analyze that the average degree of vertices and the network structure may affect the performance. For instance, the time cost of the distance 10 and 15 are similar in WikiTalk dataset. It means that the network structure of WikiTalk dataset is clustered that have less diameter than 10. We setup the average degree of synthetic datasets as 100, 200, and 300. We show that the time cost roughly increases due to the number of shortest segments exponentially increases in every superstep.

In the third experiment, we evaluate the time cost of the constructing k -index graph according to the size of graph. We set the distance threshold to 15. When the size of graph increases, the time cost also increases. The bigger dataset require more time cost due to the size of graph affects the number of messages. We observe that the number of vertices and edges increase by N times has a relative impact on 0.05–0.2 times increase in the time cost (Fig. 8).

In the fourth experiment, we evaluate the time cost of a single source shortest path algorithm using the proposed index graph comparing with using the original graph. We randomly select the five vertices as a start vertex in each case. Since the proposed algorithm using the index graph can expands longer distance in each superstep, it shows the better performance than the no index graph. In particular, the graph having high average degree shows better performance to search SSSP. When we search the single source shortest path without proposed index, we should check more paths that can be shortest paths in the graph that has higher average degree. We can reduce the

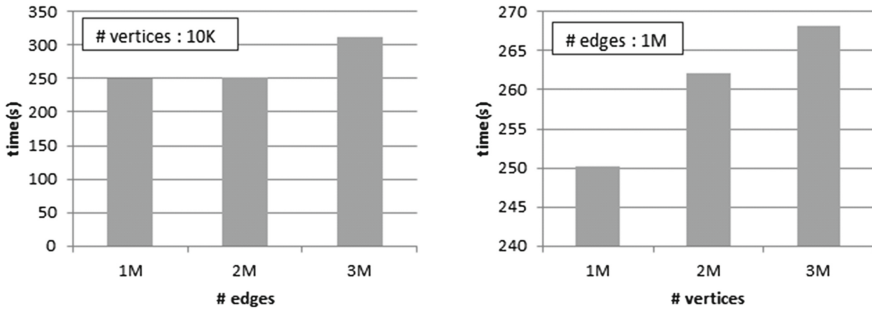


Fig. 8. Comparisons of time costs according to the size of graph

candidate paths by expanding shortest segments. Thus, our approach is more useful in processing the dense graph (Fig. 9).

In the fifth experiment, we examine the running time of the proposed algorithms with different numbers of workers to study the performance and scalability. For this experiment, we conduct the experiment that constructs the k -distance index using 16

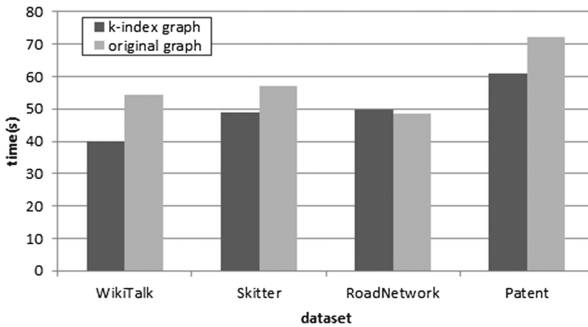


Fig. 9. Comparison of the average time cost for processing SSSP

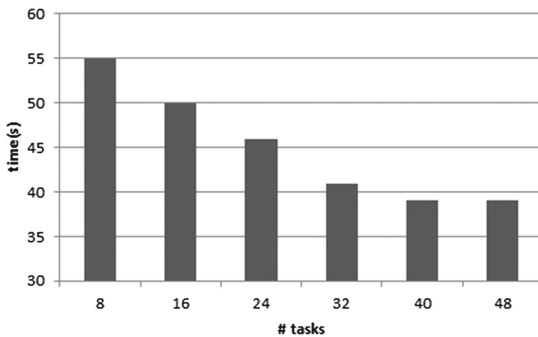


Fig. 10. Comparison of the computation time for construction k -distance index according to the number of tasks

workers. We use the Skitter dataset and set the distance threshold to 10. We observe that with the increase in the number of workers, the total computation time decrease linearly. As depicted in Fig. 10, the algorithm improves in execution time with the increase of the tasks till it reaches the 40 tasks, which is shown to be the optimal amount of tasks.

6 Conclusion

We propose an efficient indexing approach based on Pregel framework for supporting a s - t shortest path discovery. We reduce the number of supersteps by passing messages including more distance information from other vertices in each superstep. We can reach the target vertex by expanding longer distance than original edges in each superstep using the proposed k -index graph. From the experimental results, we show that the proposed method can efficiently maintain partial shortest paths that can help an efficient s - t shortest path search on large graphs. We analyze the experimental results in terms of structural features such as degree and density. Moreover, we observe that with the increase in the number of workers, the total computation time decrease linearly.

Acknowledgments. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2013R1A2A1A05056375).

References

1. Vieira, M.V., Fonseca, B.M., Damazio, R., Golgher, P.B., Reis, D.d.C., Ribeiro-Neto, B.: Efficient search ranking in social networks. In: CIKM, pp. 563–572 (2007)
2. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: KDD, pp. 137–146 (2003)
3. Ukkonen, A., Castillo, C., Donato, D., Gionis, A.: Searching the wikipedia with contextual information. In: CIKM, pp. 1351–1352 (2008)
4. Potamias, M., Bonchi, F., Castillo, C., Gionis, A.: Fast shortest path distance estimation in large networks. In: CIKM, pp. 867–876 (2009)
5. Dijkstra, E.: A note on two problems in connexion with graphs. *Numer. Math.* **1**, 269–271 (1959)
6. Wagner, D., Willhalm, T.: Speed-up techniques for shortest-path computations. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 23–36. Springer, Heidelberg (2007)
7. Cohen, E., Halperin, E., Kaplan, H., Zwick, U.: Reachability and distance queries via 2-hop labels. In: SODA, pp. 937–946 (2002)
8. Wei, F.: Tedi: efficient shortest path query answering on method for efficient shortest path discovery graphs. In: SIGMOD, pp. 99–110 (2010)
9. Potamias, M., Bonchi, F., Castillo, C., Gionis, A.: Fast shortest path distance estimation in large networks. In: CIKM, pp. 453–470 (2009)
10. Goldberg, A., Harrelson, C.: Computing the shortest path: search meets graph theory. In: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms SODA, Vancouver, British Columbia, 23–25 January 2005

11. Wei, F.: Tedi: efficient shortest path query answering on graphs. In: Proceedings of the 29th ACM SIGMOD International Conference on Management of Data, Indianapolis, USA, 6–11 June 2010
12. Yuan, Y., Wang, G., Wang, H., Chen, L.: Efficient subgraph search over large uncertain graphs. *PVLDB* **4**(11), 876–886 (2011)
13. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. In: Proceedings of the 6th Symposium on Operating Systems Design and Implementation, San Francisco, CA, 6–8 December 2004
14. Bahmani, B., Chakrabarti, K., Xin, D.: Fast personalized pagerank on mapreduce. In: Proceedings of the 30th ACM SIGMOD International Conference on Management of Data, Athens, Greece, 12–16 June 2011
15. Valiant, L.G.: A bridging model for parallel computation. *Comm. ACM* **33**(8), 103–111 (1990)
16. Malewicz, G., Austern, M.H., Bik, A.J.C., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: SIGMOD (2010)
17. Gao, J., Jin, R., Zhou, J., Yu, J., Jiang, X., Wang, T.: Relational approach for shortest path discovery over large graphs. *PVLDB* **5**(4), 358–369 (2011)
18. Mehlhorn, K., Naher, S.: The LEDA Platform of Combinatorial and Geometric Computing. Cambridge University Press, Cambridge (1999)
19. The iGraph library. <http://igraph.wikidot.com/>
20. Padmanabhan, S., Chakravarthy, S.: HDB-subdue: a scalable approach to graph mining. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 325–338. Springer, Heidelberg (2009)
21. Chakravarthy, S., Pradhan, S.: DB-FSG: an SQL-based approach for frequent subgraph mining. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 684–692. Springer, Heidelberg (2008)
22. Gregor, D., Lumsdaine, A.: The parallel BGL: a generic library for distributed graph computations. In: Proceedings of Parallel Object-Oriented Scientific Computing POOSC (2005)
23. Chan, A., Dehne, F.: CGMGRAPH/CGMLIB: implementing and testing CGM graph algorithms on PC clusters and shared memory machines. *Int. J. High Perform. Comput. Appl.* **19**(1), 81–97 (2005)
24. Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: SIGMOD, pp. 135–46 (2010)
25. Iosup, A., Lampraki, N.P., Penders, A., Biczak, M., Guo, Y., Varbanescu, A.L.: Parallelization and Distribution for Large Scale Graph Processing. HPD, Delft, The Netherlands (2012)
26. Apache Hadoop. <http://hadoop.apache.org/>
27. Amazon EC2. <http://aws.amazon.com/ec2/>
28. Leskovec, J., Huttenlocher, D., Kleinberg, J.: Predicting positive and negative links in online social networks. In: WWW (2010)
29. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD) (2005)
30. Leskovec, J., Lang, K., Dasgupta, A., Mahoney, M.: Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. *Internet Math.* **6**(1), 29–123 (2009)
31. Hang, T.L.: A Java Library of Graph Algorithms and Optimization. Taylor & Francis, Hoboken (2007)

Customized Information Interface with Web Applications

Wookey Lee¹, Suan Lee², and Jinho Kim²(✉)

¹ Informatics Engineering Lab, Department of IE,
Inha University, Incheon, Korea
trinity@inha.ac.kr

² Department of Computer Science, Kangwon National University,
Chuncheon, Korea
{webdizen, jhkim}@kangwon.ac.kr

Abstract. When information is searched via internet, a browser indicates information about web pages on a single window, but the existing browser shows only fragments of page information to web surfing users who visit several sites at once and in turn causes insufficiency and inconvenience to the users. Rich Internet Application techniques, which are web application techniques for the simple and easy operation and diverse and dynamic screen composition, have received a lot of attention as a next-generation UI technique emphasizing on users' convenience. In this dissertation, a two-dimensional and sequential advanced search is realized with the use of dynamic UI so users can save and employ the customized search information for further web search. Also, the search structure has been designed with the use of user-oriented keyword preference to have more customizes search results than the existing web search. Furthermore, this paper has proven a decrease in the number of searched pages by employing the customized search administrator using RIA techniques. Thus, it could be concluded that the customized search administrator supports users of the more efficient and flexible customize web search.

Keywords: Web browser · UI technique · Web search · Rich internet application technique · Customized search

1 Introduction

To find the desired information from vast web data, users use various search tools and web search engines where search engines allow users to search for information not only by a simple keyword, but also having advanced search so that users may assign certain search conditions including the date, region and file format, etc. The search functionalities, however, cannot support for storing the advanced search conditions or results by which such limitations have become an obstacle for the efficient utilization of personalized search. Moreover, since the advanced web search setting consists usually of text-based options, so that the user should type an additional query into the search engine and re-search if he/she wants to exclude a certain keyword.

Recently, the recommended word automation service is to recommend a word which is extracted by assigning a weight to each entry. The weight evaluation is

processed with which a web can be expressed by a u-shape graph [1] and the weight in turn reflects on a user's query. In this method, however, the criteria for the weight assignment are based on the analysis of the whole web search engine perspective not on the direct evaluation of individual user. Consequently, they cannot fully support of the personalized search.

RIA technique, a representative technique of new web technology, plays a major role to handle multiple processes in a single interface by linking the user interface and data API with the two dimensional expression and sequential factors of the existing web applications. The RIA technique has been developed through some common techniques that are dynamic and expandable [2] which includes common examples like Ajax, Flash-based Flex, desktop-based Opera, Widget of Yahoo and Windows Live Gadget of Microsoft. By incorporating the web application including RIA techniques, we suggest a novel method for the advanced search which includes not only the data of the personal local area but also the documents on the internet and the direction for web search to find a value in line with a user's intention and interest.

This paper is organized as the followings. The basic concepts and techniques of the web application are reviewed in Sect. 2 and a user-oriented search structure based on the web application techniques are suggested in Sect. 3. As for the last, it is concluded and a future study direction is suggested in Sect. 4. Note that this paper is an extension of our previous works [8, 10].

2 Web Application

With the rapid development of web-related techniques, developers have to understand web applications as much as local computer-based applications [3]. The features of the web application techniques are introduced in this chapter along with representative RIA and AJAX core components. Then, the latest trends of the web application techniques are introduced and evaluated.

2.1 Difference Between the Web Application and the Existing Applications

The web application with RIA and AJAX techniques, are the representative techniques of web which enables web applications to have similar features and functions of desktop applications. Starting from the AJAX (Asynchronous JavaScript and XML), Active X Control of MicroSoft, Java of Sun to the latest of Apollo of Macromedia [4], these techniques are collectively termed as RIA, and W3C named a working group as Rich Web Client [5] and started to activate since the middle of year 2005. Among these vast types, AJAX, supporting of openness, expandability and standardization of web, has received the most attention since ActiveX has a limitation on its OS compatibility and flex-based APOLP, in development, is a commercial application not an open source. AJAX techniques do not require of the installation the Active X for providing interactive services. It prevents security issues which may arise by installing the unproven Active X.

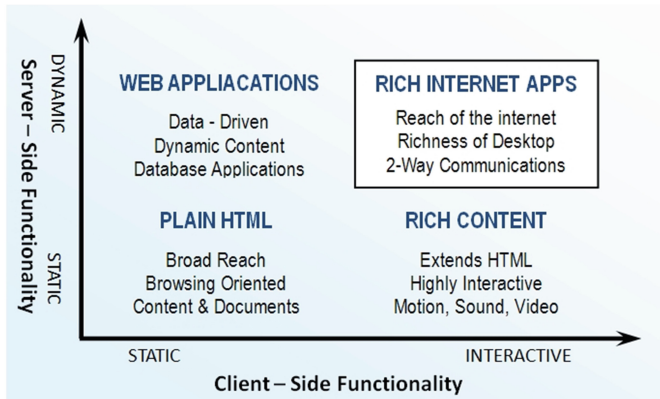


Fig. 1. Dynamic and bilateral RIA technique

RIA and AJAX techniques have one thing in common that they both assure of enhancement of UX (User eXperience) than the existing techniques [6]. As shown in the Fig. 1, RIA and AJAX techniques are dynamic and interactive on both a server side and a client side unlike the existing web techniques. Especially since they interpret XLM and script, not compiled code in real-time, interaction with users can be much closer. The most significant strength of RIA and AJAX technologies is that they can support of web search personalization by user profiling which makes it so simple and easy to have the advanced web search.

2.2 AJAX (Asynchronous JavaScript and XML)

AJAX, an acronym for Asynchronous JavaScript and XLM, was coined by J. Carret [3], where the Engine can be improved the user experience by renewing only a part of a page requested by a user through the use of JavaScript and XLM rather than renewing by a page. The AJAX technology has cleared a boundary between desktop applications and web applications and in turn, has built the foundation for realizing a web application with more enhanced interfaces. Also, AJAX techniques are not bound to specific features of operation systems and/or browsers. Via XMLHttpRequest, AJAX could control the data communication with HTTP. This technique, first introduced and realized by the Mozilla, was widely adopted by browsers of many companies and then became a standardized technique [9].

2.3 FLEX and Laszlo

FLEX has been developed upon the strong integrated development environment of Adobe as well as Flash techniques which are presumed to be standardized techniques. FLEX makes it possible to create dynamic UI, which could not be achieved with HTML. Despite of its strength, however, it does not get that much attention, because it is not an open source platform unlike the library and framework of AJAX and it is a

commercial tool. On the other hand, Laszlo, another flash-based RIA and AJAX technique, aims to be an open source platform. Laszlo deploys a XLM-based script named as LZX and it is an object oriented, tag-based language that uses XLM and JavaScript. Also, it allows compiling a source code into either flash code or DHTML.

2.4 Firefox and Silverlight (WPF/e)

XUL (XML User Interface Language) of Mozilla is a XML-based markup language which can be used in browsers developed by Mozilla and it is used to write the user interface control. Moreover, it uses the document object model (DOM) for a hierarchical document structure. For instance, Firefox of Mozilla is realized as one package containing a XUL file, JavaScript and CSS (Cascading Style Sheets). On the other hand, Microsoft developed a web presentation technique called ‘Silverlight’ under a code name of WPF/e. Silverlight is a powerful development tool which has various functions, rich visual and interactive implementation on multiple platforms [8]. The core of this technique is a XAML (eXtensible Application Markup Language)-based presentation function. XAML is a text-based XML. Therefore, it can be easily used in a firewall environment and it can be used to write an event handler and/or allow the interaction of contents with the use of JavaScript.

2.5 Widget and Gadget

Widgets are client applications expanded with the use of Flash or JavaScript techniques, just like Plugin. The basic format of a widget is to provide a dynamic user interface based on XLM and it generally comes in JavaScript and CSS. In general, it has been developed for either providing or controlling data from Open API [9–12] as GUI. As many sites have decided to open their API recently, it has also been developed in a format of MashUp or Web Application Hybrid. Such widgets are currently developed by Opera and Yahoo and Microsoft has started to offer a Gadget, which has the same functions as Widget, for Vista.

2.6 XMLHttpRequest Object

It is a technique for exchanging data between a client and a server with the use of HTTP. Asynchronous transmission is the core of Ajax techniques. Most browsers support the XMLHttpRequest and from version 5.0, Internet Explorer provides it in a format of ActiveX object.

2.7 Widget-Based Web Search Application

Widgets can operate independent of a web browser but still can have free processing in a client area. Even though there are various types of applications including Opera, Widget of Avedesk and Gadget of MS Vista, the Widget of Yahoo seems to be a proper application for the customized web search and evaluation since it can freely integrate with Yahoo search API.

2.8 Weakness and Limitation of AJAX

The fact that AJAX uses an open source could be a huge advantage, but also its weakness at the same time. Open source means that everyone can easily acquire a source code so it is hard to differentiate a site. Also, it could cause severe loads on the server when data is requested consistently since it exchanges data in an asynchronous fashion. Its biggest weakness lies on its security. In case of the existing methods, business logic exists on all servers, but AJAX has business logic on a client side as well so it cannot be safe from hackers' attacks. For dealing with personal information, it requires extra attention on security.

The existing yahoo widget has a search widget with a simple search function and a selective search widget which can select a search engine with the Open API, but not a widget which provides an advanced search function as GUI or provides and then saves and maintains an advanced search function as GUI.

On the other hand, Google saves and manages users' queries and links for the search results of these queries for a certain period with a beta service called 'search record', but does not reflect them on new web search of the user. This paper has designed the customized web search structure with the widget-based web search application utilizing RIA and AJAX techniques which can overcome limitations of the existing advanced web search, differentiating from the existing search widget and simple storage of searched results.

3 Search System with RIA

3.1 Search System Structure

The search system with RIA and AJAX techniques which has been suggested by this paper is revealed in the Fig. 2.

- GUI Application Based XML

Activities of the existing techniques including Active X, NS Plug-in and Flash are limited to be within web browsers. In turn, they are not proper enough to provide sufficient UE. Therefore, they should be enhanced in this term. First of all, Widget, the most actively being developed technique among RIA and AJAX techniques, Widget was chosen to take out a web out of a browser.

Widgets can operate independent of web browsers but still maintain to be interactive with web browsers and also, it can renew only a page block rather than a whole page by using an asynchronous language. Moreover, dynamic GUI can manage values of strings of text types, URL or even file information on a different client area which allows users to have easier access to the advanced search. It decreases a unit of page renewal and in turn increases the application processing speed and enhances fusibility and visual effects. Also, developers can manage errors and/or bugs in a more efficient manner [12–16].

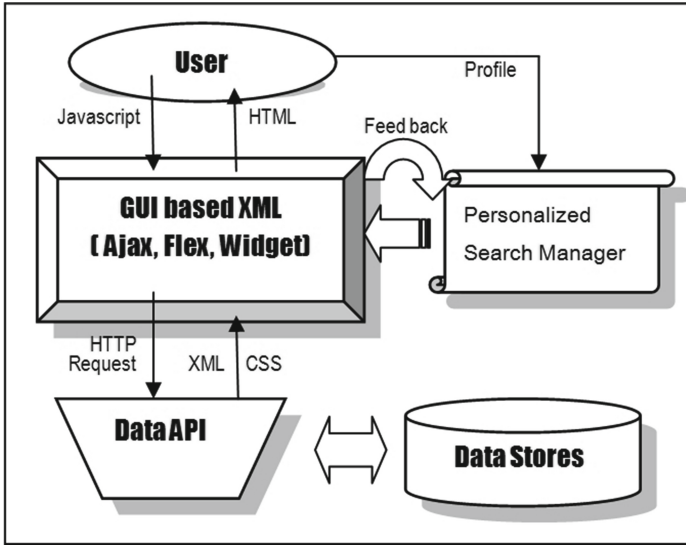


Fig. 2. Search system with RIA techniques

- Personalized Search Manager

For the existing advanced search services, entries of users are automatically initialized back to their default values upon updating of a page rather than reapplied. However, a personalized search administrator saves and manages users’ entries and reflects them onto another search. Furthermore, the user profiling managed by a personalized search administrator increases a possibility for listing best-matching web pages satisfying users’ search intention.

Thus, the administrators remember users’ preferred keywords and non-preferred keywords through users’ search patterns and/or setting and reflect them onto filtering of searched results.

- Data API

API, an abbreviation of Application Programming Interface, is an interface for accessing web tools provides by operation systems and web sites and controlling data. Since many companies including Google, Amazon and Ebay have recently disclose their AIP sources, now anyone can create his/her own application with the RIA and AJAX techniques and open APIs. In other words, the RIA and AJAX techniques allow people to select APIs that they favored and create more light and expandable applications with data obtained by using open APIs.

3.2 Site Access Process

Figure 3 depicts the site access in algorithm. At first, click a button on LinkForm. Then, test the entered URL and window number and access to a site. on_loadedLink(url) on the first line is a function generating upon clicking the button after entering Url onto

```
1 | function on_loadedLink(url) {
2 |   if(!draggableFloatId[selectBox()] ||
   |   divArray[selectBox()].style.visibility=="hidden")
3 |     return alert("No Box")
4 |   else if(url==" " || url=="NULL") {
5 |     return alert("Enter URL") }
6 |   else {
7 |     if(url.substr(0,7)!="http://") {
8 |       url="http://" +url;
9 |       linkInputForm.linkURL.value="
   |       http://" +linkInputForm.linkURL.value; }
10 |   document.getElementById(selectBox()).
   |   innerHTML="<iframe src='"+url+"' >"
```

Fig. 3. Interface algorithm

linkForm. On the second line, check whether the corresponding box exists or not. If there is no window, call a return alert for the box on the line 3. The fourth line is to check whether Url is entered into a URL box. When URL is not entered, call a return alert on the fifth line. If URL and window number are correctly entered, check to ensure http:// is added to the entered URL on the sixth line. On the tenth line, access the entered URL in the selected box.

3.3 Query Process

To evaluate the system suggested by the study, its query process was designed as the following by using an advanced search service of Yahoo Search API and Yahoo Widget engine [14].

For processing of a query, PSMD (personalized search manager data) is created based on values set by a user and a searchList, which is created with searchText values that are entered by a user and search words that are set and entered by a user, as an object.

On the second line, a search word is entered by a user, and the entered search word is compared with a searchList to check whether then entered value is a word that has been used and/or set previously on the third line. If there is no history for the entered value, it is recorded onto a searchList on the fourth line, and the entered SearchText is transmitted to API on the fifth line. As for the study, only Yahoo Open API is used. Thus, URL is requested by just entering a SearchText into Yahoo Search API URL [16].

```

1  var SearchText, SearchList, PSMD
2  get(SearchText)
3  If(SearchList != ""){
4      SearchList = add SearchText
5      OpenURL("OpenAPI URL + SearchText)
6  }
7  Else {
8      SearchList = join SearchText
9      Run Personalized_Search_Manager()
10     If(Searchlist.Function == "None of these word")
11         put PSMD = "%2D" + SearchText
12     else If(Searchlist.Function == "Any of these word")
13         put PSMD = "%2B" + SearchText
14         .....
15     get(PSMD)
16     OpenURL("OpenAPI URL + SearchText + PSMD)
17 }

```

Fig. 4. Query processing

For instance, when a user enters ‘Apple’, the search word is entered into a SearchText and then, it is compared with a SearchList to check whether the entered search word has been previously searched or set. If there is no such history, the entered search word is added to a SearchList and it is immediately sent to the search API. However, if it is not, the search word is combined so the search word can be referred by a customized search administrator.

3.4 System Realization

Yahoo Widget can be developed by using a framework and library provided by the Widget engine. The system has been realized based on the web document search API and search Widget provided by Yahoo. For obtaining the personalized search results that this study aims for, additional buttons and menus are composed as shown in the Fig. 5.

To realize the query process depicted in the Fig. 4, the existing advanced search has been realized with GUI and it also has been designed to allow entering, modifying and managing with various events, including typing, clicking, drag and drop and hovering, to have more intuitive interface. Even if a searchList is written based on the entered values, it is still possible to examine and filter out the searched results with various limit values. To support this function, checks, scrolls and textboxes were added for selecting the desired ranges of dates, sites, file formats and areas. Advanced search functions and search limitation functions that are realized with GUI are saved onto a SearchList upon setting of a user and then converted into operations of which API can understand as in the Table 1. For the final, a customized search administrator



Fig. 5. Customized search screen

Table 1. Operators of yahoo web search API

Function	Operator
None of these word	%2D
Any of these word	%2B
Within the past 3 months	&n=10&fl=0&vm=i&x=wrt&vd=m3
Within a year	&n=10&fl=0&vm=i&x=wrt&vd=y
PPT file format	&vst=0&vf=ppt
Excel file format	&vst=0&vf=xl

combines operators and values of SearchList for the corresponding functions to create resulting values as PSMD that API can understand.

Even though Widget operates independent of internet connection and/or web browser, it provide an intuitive UI which allows of drag of drop of a desired word to the list even during web surfing as shown in Fig. 6.

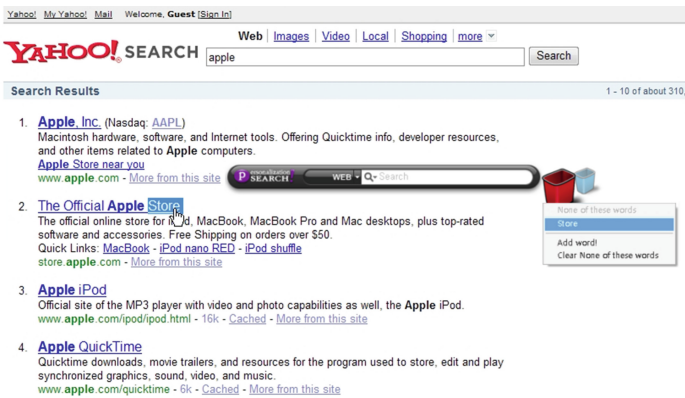


Fig. 6. Drag and drop of excluding words

3.5 System Evaluation

To evaluate how the structure suggested by the paper can provide sufficient UE to users, a user scenario was prepared, and the evaluation was made based on the scenario. The scenario is as the following.

A user who prefers news entered ‘Apple’ as a search word. He/she wanted to exclude a word ‘Store’ from the search results and to rank the search results in an order of the frequency of a word ‘IT’ and select the desired data range of 3 months. The web document address that the user searched for was “Apple.slash dot.org”.

Results of the general web search, advanced web search and personalized search administrator system of which suggested by the paper were evaluated and compared. To use the same search engine, the search results were compared with the use of the customized search administrator using Yahoo Web Search, Yahoo Web Document Advanced Search and Yahoo Web API and they are listed in the Table 2.

Table 2. Results of evaluation with the use of yahoo search engine

	Page rank	User’s entered value	No. of searched pages	Setting time
General search	6	apple	327,000,000	T*
Advanced search	2	apple, news, IT, store	127,000,000	6T
Customized search	1	apple	91,800,000	T

* T = A period of time require for entering a single word and setting one entry for advanced search

For general search, a user is required to enter only a short keyword of ‘Apple’ but she/he has to search 5 unnecessary upper rank pages to reach to the desired page. On the other hand, with advanced search, best matching web pages are ranked first but it requires more entries from a user. As shown in the Table 2, when a period of time required for setting one entry is presumed to be T, 6T is required for setting of entries for date and file format for the advanced search. However, when a user enters a new search word, all the previously entered entries are initialized and the user requires entering the same entries again.

However, it becomes possible to set the search entries in a simple manner with drag and drop with the use of an intuitive and dynamic UI when the aforementioned customized search administrator is applied onto the advance search. It also allows excluding of pages with titles that a user already checked and/or overlapped links by dragging them to the customized search administrator. Furthermore, only a script that a user changes gets re-transmitted rather than an entire page, it can be flexibly applied onto a new search word to give back the search results that that suitable for the user’s intention.

Another experiment was to calculate a period of time starting from a point of calling a page to of completion with Ethreal, a network analyzer [10]. To check the connection time of the existing web sites per data capacity, we accessed to arbitrary web pages corresponding to data capacity of 3,202–177,414 (byte). The Table 3 is a list of accessed sites and the Fig. 7 shows results of the measurement of webs.

The experiment of the Fig. 7 shows the connection time of web for visiting one site, but the connection time of AJAX-based page division is for visiting multiple pages.

Table 3. List of accessed lists

Site URL	Data capacity
www.7-eleven.co.kr	3202
www.google.com	5349
www.tnccompany.com	5429
www.fox.com/home.htm	15649
code.google.com	16794
ajax.asp.net	23991
www.kaist.ac.kr	30612
www.lge.co.kr	37708
java.sun.com	40200
www.ebay.com	66987
www.cjmall.com	94128
www.yahoo.com	107946
news.google.com	145410
www.joins.com	177414

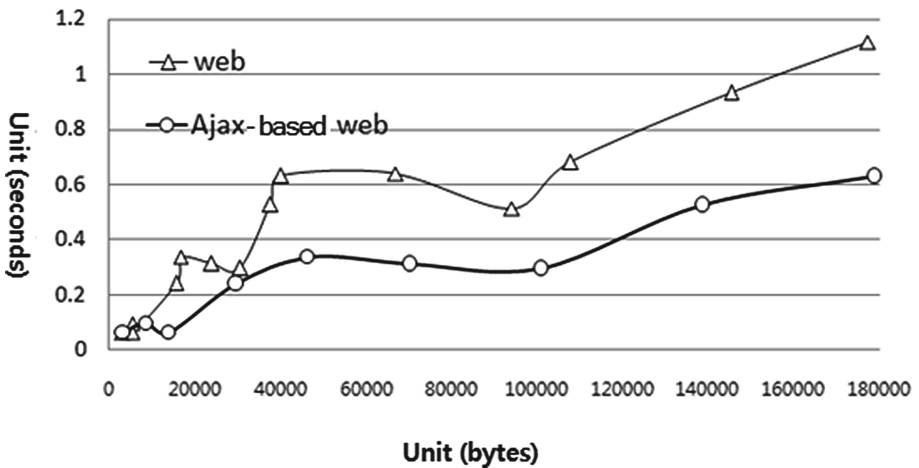


Fig. 7. Comparison of speeds of web and AJAX-based web

Despite of such difference, it is revealed that the AJAX method requires the less connection time in case of the same data capacity. That is because of that even though the size of total data shown on the page is the same, the AJAX-based page division transmits data on the requested part of a page and it decrease the total connection time.

When AJAX-based page division is used, there is no difference in terms of a period of time required for accessing to a single site. However, its access speed can be kept steady regardless of accumulated data even when it access to multiple sites since only data from the requested part of a page are transmitted.

The existing advanced searches provide various options but none of them allows of setting a priority list of entries. Nevertheless, by using interactive RIA and AJAX

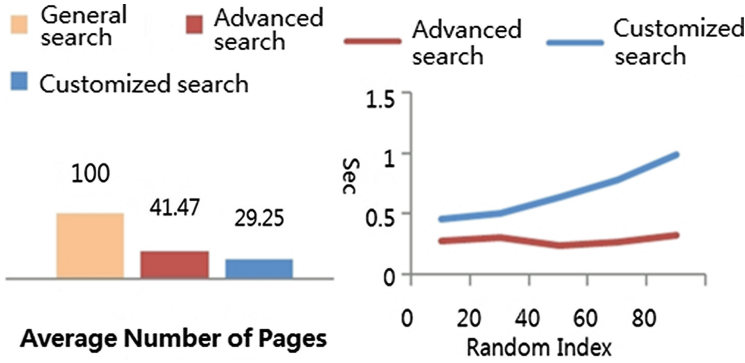


Fig. 8. Average decrease in the number of pages

technologies, each of search entries, which are options for the advanced search, can be integrated into a priority list for search. To evaluate it, 100 random search words were searched with 3 advanced search options just like the aforementioned scenario. The Fig. 8 shows the average numbers of searched pages. Comparing to the advanced search, the number of pages searched with the use of the customized search administrator is far less.

On the other hand, the advanced search of 30 random words without a priority list showed that the utilization of a priority list does not significantly affect the search time, but search time is increased for the customized search as a priority list gets more mixed.

4 Conclusion and Future Research

This paper suggests a customized search structure which employs filtering built by the web application such as RIA and AJAX techniques where user’s keyword preference to provide a list of best-matching web pages according to users’ intention. Since RIA and AJAX techniques can provide dynamic UI but still perform various features including client applications, they can be used to enhance performance of the existing advanced search services. Moreover, it has proved that the customized search administrator can save the preferred and non-preferred keywords by the user profiling feature and, can filter out the searched results with the saved preferred and non-preferred keywords, and can limit the desired file format and/or can update date range to give back more specific search results.

For the future research, each and every search sites should save the user profiling on its server and realize the suggested customized search in order to provide customized services effectively for the user preferences. In other words, there is much room for RIA and AJAX applications to be employed for server sides. Also, for more sophisticated customized search, it would be necessary to study on how the customized search administrator recognizes users’ patterns of search keywords, so that a graph oriented big data pattern can be figured out effectively.

Acknowledgement. This research was partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (2011-0011824).

References

1. Arora, N.R., Lee, W., Leung, C.K.-S., Kim, J., Kumar, H.: Efficient fuzzy ranking for keyword search on graphs. In: Liddle, S.W., Schewe, K.-D., Tjoa, A.M., Zhou, X. (eds.) DEXA 2012, Part I. LNCS, vol. 7446, pp. 502–510. Springer, Heidelberg (2012)
2. Dincturk, M.E., Choudhary, S., von Bochmann, G., Jourdan, G.-V., Onut, I.V.: A statistical approach for efficient crawling of rich internet applications. In: Brambilla, M., Tokuda, T., Tolksdorf, R. (eds.) ICWE 2012. LNCS, vol. 7387, pp. 362–369. Springer, Heidelberg (2012)
3. Garrett, J.J.: The Elements of User Experience: The User-Centered Design for the Web and Beyond. New Riders Publishing, Thousand Oaks (2010)
4. Chambers, M.: Developer relations for apollo at adobe (2007). <http://labs.adobe.com/wiki/ind-ex.php/Apollo>
5. W3C Rich Web Clients (2010). <http://www.w3.org/2010/rwc/>
6. Grigorik, I.: Making the web faster with HTTP 2.0. Commun. ACM **56**(12), 42–49 (2013)
7. Choudhary, S., Dincturk, M.E., Mirtaheeri, S.M., Jourdan, G.-V., Bochmann, G.V., Onut, I.V.: Building rich internet applications models: example of a better strategy. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 291–305. Springer, Heidelberg (2013)
8. Lee, W., Choi, C., Yoon, S.: Synchronous web browser fragmentation for structural web searches. JITA **6**(2), 161–170 (2009)
9. Moroney, L.: Getting started with WPF/E (2013). <http://msdn2.microsoft.com/en-us/library/bb190632.aspx>
10. Park, C., Lim, T., Lee, W.: RIA based personalized search with widget implementation. J. KIISE **13**, 402–406 (2007)
11. Ritter, A., Mausam, Etzioni, O., Clark, S.: Open domain event extraction from twitter. In: KDD. pp. 1104–1112 (2012)
12. Flickr Open API. <http://flickr.com/services/api/>
13. Yahoo Open API. <http://developer.yahoo.com/>
14. Google Open API. <http://code.google.com/>
15. Naver Open API. <http://openapi.naver.com/>
16. Yahoo! Widgets. <http://widget.yahoo.com/>

Leveraging Enterprise Application Characteristics to Optimize Incremental Aggregate Maintenance in a Columnar In-Memory Database

Stephan Müller^(✉), Paul Möller, and Hasso Plattner

Hasso Plattner Institute,
University of Potsdam, Potsdam, Germany
{stephan.mueller,paul.mueller,hasso.plattner}@hpi.uni-potsdam.de

Abstract. An analysis of database workloads generated by enterprise applications revealed a mixed workload of short-running transactional and long-running analytical queries. With the latter type of queries containing many aggregate operations, we implemented an efficient aggregate caching mechanism. But the incremental materialized view maintenance is very costly for aggregate queries joining multiple tables. To overcome this problem, we analyzed the characteristics of enterprise applications with respect to the creation of business objects and their persistence in the database layer. We evaluated how the detected patterns can be leveraged to reduce the join operations between the main and delta partitions of the involved tables in a columnar in-memory database. The resulting performance improvements are significant and close to using the caching mechanism with a denormalized schema.

1 Introduction

Until recently, enterprise applications have been separated into online transactional processing (OLTP) and online analytical processing (OLAP). The drawbacks of this separation are complex and costly ETL processes, not up-to-date and redundant data. Further, the analytical applications are often limited in their flexibility due to pre-calculated data cubes with materialized aggregates.

With the rise of columnar in-memory databases (IMDB) such as SAP HANA [1], Hyrise [2] and Hyper [3], this artificial separation is not necessary anymore as they are capable of handling mixed workloads, with transactional and analytical queries on a single system [4]. In fact, a modern enterprise application executes a mixed workload with both – transactional *and* analytical – queries [5]. While the transactional queries are mostly inserts or single selects, the analytical queries are often comprised of costly data aggregations [6]. Having the possibility to run flexible, adhoc analytical queries directly on transactional data with sub-second response times will further lead to an increased workload of aggregate queries.

To speed up the execution of analytical queries with aggregates, *materialized views* have been proposed [7]. Accessing tuples of a materialized aggregate is

always faster than an aggregation on the fly. The overhead of materialized view maintenance to ensure consistency for changing base data has to be considered, though [8]. Apart from temporary transactional inconsistencies, a downtime is not acceptable in during materialized view maintenance in mixed workload environments.

While existing materialized view maintenance strategies are applicable in columnar IMDBs [9], their specific architecture is well-suited for a novel strategy of caching aggregate queries and applying incremental view maintenance techniques [10]. This is because the storage of columnar IMDBs can be separated into a read-optimized main storage and a write-optimized delta storage. Since the main storage is highly-compressed and not optimized for inserts, all data changes of a table are propagated to the delta storage in order to ensure high throughput. Periodically, the delta storage is combined with the main storage in a process called *merge operation* [11]. The materialized aggregates do not have to be invalidated when new records are inserted to the delta storage, because they are only based on records from the main storage. Instead, the final, consistent aggregate query result, is retrieved by aggregating the newly inserted records of the delta storage on the fly and combining them – using a SQL UNION ALL statement – with the materialized aggregate.

One challenge of the proposed aggregate caching mechanism and the involved incremental materialized view maintenance is to handle aggregate queries that are based on joins of multiple tables. These queries require a union of joining all permutations of delta and main partitions of the involved tables, excluding the already cached joins between the main partitions. For a query joining two tables, three subjoins are required, and query joining three tables already requires seven subjoins. This may result in very little performance gains over not caching at all the query on the main partitions. After analyzing the characteristics of enterprise applications, we identified several schema design and data access patterns that can be leveraged to optimize the overall database performance. While these business semantics could potentially be applied to several other aspects for data processing in a columnar IMDB, this paper focuses on an approach to reduce the incremental view maintenance by explicitly leveraging business semantics of applications.

After discussing related work in Sect. 2, we describe the identified enterprise applications characteristics in Sect. 3. Section 4 describes the aggregate cache and strategies to reduce the number of joins for cached queries. We then outline in Sect. 5 how the database engine can obtain information about application characteristics. Our benchmarks in Sect. 6 support the significant speedup potential and Sect. 7 concludes the paper with the main contributions and an outlook on future work.

2 Related Work

A database can have different design goals depending on the application and its characteristics. The CAP theorem is an example of how different design

trade-offs have to be balanced [12]. In fact, there is an emergence of databases that are custom-built for specific applications such as Cassandra¹ or Amazon DynamoDB², each with its own design goals according to the characteristics of the application.

The enterprise application characteristics identified and discussed in this paper are used to reduce the incremental view maintenance inherent when introducing materialized views to speed-up analytical queries [8]. The maintenance of materialized views has received significant attention in academia [13, 14] and industry [15, 16], and the problem of incrementally maintaining aggregate queries with joins has been widely identified [17, 18]. However, neither of these approaches use the characteristics of the application to reduce the maintenance effort.

3 Enterprise Application Characteristics

In this section we give an overview of identified enterprise application characteristics, that can be utilized to speedup processing of join queries for the aggregate cache. Two aspects are of essential relevance: what are common patterns of database schema design and workloads.

3.1 Schema Design

In different domains, we identified tables with similar design patterns, namely header, item, dimension, text, and configuration tables.

A *header* table describes common attributes of a single business transaction. E.g., for a sale in a financials system it stores who made the purchase and when the transaction took place. In materials management the header stores attributes common to a single movement of goods like who initiated the movement and also the time it took place.

To each header table entry, there are a number of corresponding tuples in an *item* table. Item entries represent entities that are involved in a business transaction. For instance, all products and the corresponding amount for a sale or materials and their amount for a goods movement are stored in the items table.

Additionally, attributes of the header and item tables refer to keys of a number of smaller tables. Based on their use case we categorize them into dimension, text and configuration tables. *Dimension* tables manage the existence of entities, such as accounts and materials. Especially companies based in multiple countries have *text* tables to store strings for dimension table entities in different languages and lengths (e.g., product names). *Configuration* tables enable system adoption to customer specific needs and business processes.

¹ Distributed key value store focusing on scalability and high availability, <http://cassandra.apache.org/>.

² Managed NoSQL database focusing on cost efficiency, <http://aws.amazon.com/dynamodb/>.

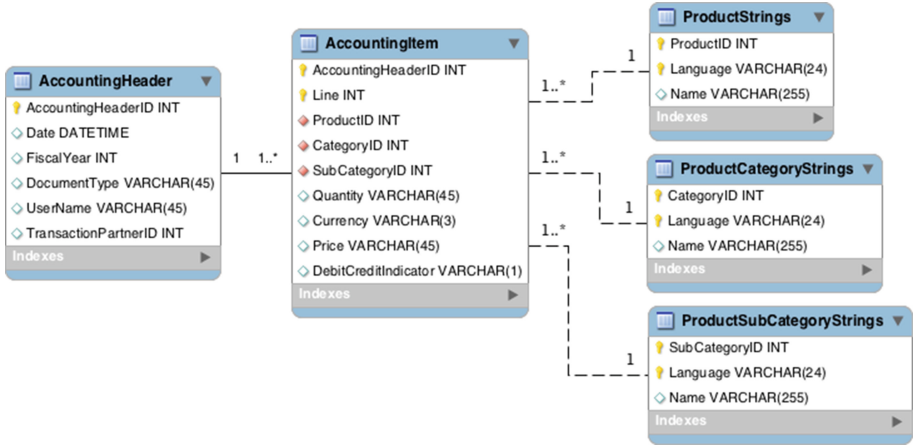


Fig. 1. Simplified schema extract of a financials application.

Figure 1 shows a simplified extract of an example schema of a SAP financials application from an international company producing consumer goods. An accounting header entry refers to a single business action, e.g. a sale or a purchase. It includes the specific time, what kind of accounting document this is, what system user entered the document and with whom the transaction took place. The accounting item table lists all invoiced or billed items. The three text tables store the real names in different languages for the involved products and other item properties.

3.2 Workload Patterns

We also see patterns in how the previously described schemas are used. There is a high insert load from enterprise systems persisting business transactions. Each transaction is represented by one header and a number of item tuples. Therefore the header and item tables have a high insert load and a large tuple count.

In many domains entire static business objects are persisted in the context of a single transaction. Therefore the header and corresponding item tuples are inserted at the same point in time. E.g. sales or goods movement transactions are persisted as a whole in the database. In some domains such as sales order management, items may be added or changed at a later point in time, e.g. when a customer adds products to his order. As [4] analyzed a number of enterprise systems, there is only a small amount of updates and deletes compared to inserts and selects on the header and item tables. Looking at aggregation join queries, we can almost always see that header entries are joined with their corresponding item entries.

Additionally, the analytical queries extract strings from dimension or text tables. Item tuple values are aggregated according to methods described in configuration tables. The number of involved smaller tables varies between none to

five. Those three table categories do have a number of properties in common. There are rarely inserts, updates or deletes and they contain only a few entries compared to header and item tables.

Starting in Sect. 4.2 we describe how each mentioned characteristic allows to reduce the number of table joins necessary when processing a query with a materialized view.

4 Optimizing Incremental Aggregate Maintenance

In this section we give a brief overview of how the aggregate cache utilizes the main-delta architecture to handle mixed workloads as explained in [10] and how enterprise application characteristics can be applied to improve the incremental maintenance of aggregation queries involving a join of multiple tables.

4.1 Architecture Overview

As depicted in Fig. 2, the query processor handles reads and writes to main and delta storage through the SQL interface from the application and delegates aggregate queries to the aggregates caching manager. In case the cache management table (CMT) indicates that the current query has not been cached yet, the query is processed on the main and delta storage. The query result set from the main is being cached and an entry in the CMT is created. Finally the unified result sets from main and delta are delivered back to the application.

As all new inserts are stored in the delta, an already cached query only needs to be executed on the delta storage. The final result set is obtained by unifying

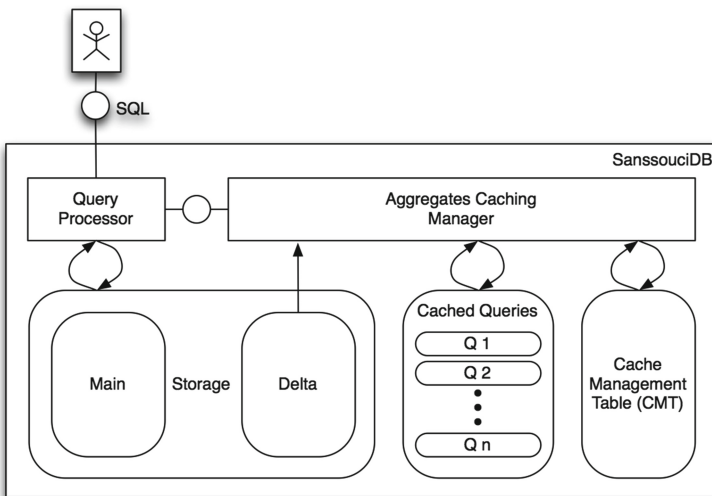


Fig. 2. Aggregate cache architecture.

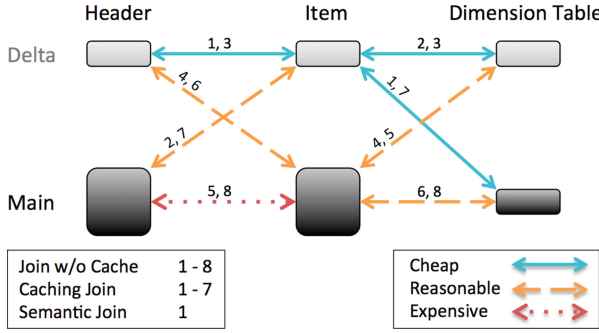


Fig. 3. Caching strategies for a three table join query.

the results from the delta with the cached entry that holds the content of the main storage. Since the delta is far smaller than the main and retrieving a cached result takes little time, the aggregate cache can speedup analytical queries by order of magnitudes. As there are only few updates and deletes in enterprise workloads as outlined in Sect. 3, we focus on insert only workloads in this paper.

4.2 Joins

Based on the header/item composition relationship and the use cases for the financials (see schema in Sect. 3) and materials management systems, we focus on inner joins, mostly with an equality operator in the join-predicate (equi-join) [19]. We define an aggregation query joining t tables between relations R with join conditions C as

$$Q_{Agg}(t) = R_1 \bowtie_{C_1} R_2 \dots \bowtie_{C_t} R_{t+1}$$

Each table consists of two partitions $\mathbb{P} = \{main, delta\}$. For a query involving a join of two tables, the database engine internally has to process more than just one join in order to retrieve a complete result set. The mains of both tables need to be joined, both deltas and both main-delta combinations of the two tables.

In the following subsections we show how to handle joins with different aggregate caching strategies. All variants are compared based on an example query involving a header, item and dimension table as depicted in Fig. 3. Each number represents a subjoin that needs to be unified with the UNION ALL SQL operator. The analytical queries of the financials application always included a join between the large header and item tables, and a varying number of smaller configuration and text tables.

4.3 Join Without Cache

Without caching, the database engine needs to run the join on all possible main-delta combinations $\mathbb{J}_{noCache}$ of all involved tables to build a complete result set:

$$\mathbb{J}_{noCache}(t) = \mathbb{P}^t$$

To evaluate Q_{Agg} joining t tables, that adds up to a total of 2^t subjoins to be unified:

$$ResultSet(Q_{Agg}) = \bigcup_{(p_1, p_2, \dots, p_t) \in \mathbb{J}_{noCache}} \left(R_{1_{p_1}} \bowtie_{C_1} R_{2_{p_2}} \dots \bowtie_{C_t} R_{t+1_{p_{t+1}}} \right)$$

As depicted in Fig. 3, for a join query involving three tables, this would mean unifying the result sets of eight sub joins.

Based on the size of the involved table components, the time to execute the subjoins varies. In our example the subjoins #5 and #8 require the longest time, since they involve matching the join condition of the mains of two large tables.

4.4 Caching Join

When using the aggregate cache, the result set from joining all main partitions is already calculated and the total number of subjoins is reduced to $2^t - 1$:

$$\mathbb{J}_{withCache}(t) = \mathbb{J}_{noCache}(t) \setminus \{main\}^t$$

For our example from Fig. 3, the subjoin #8 does not need to be rerun based on the cached result set. Since the database does not know anything about the semantics of the involved tables and therefore their usage characteristics, it has to assume there could potentially be newly inserted tuples in the delta of the dimension table, that create a new match for the join of the header-main and item-main. Based on their size, that subjoin requires a lot of time though. The header-main/item-main join needs to be run even more often, there more dimension, text or configuration tables are involved. Depending on the overhead induced by the caching mechanism (incremental update during merge, check of cache admission policy, ...), the regular caching join may not improve performance for analytical queries with three or more tables.

In case we have a cached query involving only a header and item table, only the header-delta/item-delta, header-main/item-delta and header-delta/item-main subjoins need to be computed. Since deltas get merged before they get to large, those subjoins take little time. Therefore the caching join delivers a speedup for analytical queries limited to joins involving only two tables.

4.5 Semantic Join

In this subsection we show which table components need to be joined, when the database is aware of the enterprise application characteristics introduced in Sect. 3. In Sect. 5 we explain how the database can become aware of those characteristics.

Let us assume a query joining a header and item table with a present cached result set representing the joined mains. As static business objects are inserted in the context of a single transaction, the header tuple and the corresponding

item tuples are inserted together. If there was no merge yet, both tuples that will match the join condition are both in the delta part of their table. Therefore we only need to run the header-delta/item-delta join and unify the results with the cached entry. The main-delta combinations of header and item table can be avoided. Same holds true for the subjoins #2, #4, #6 and #7 of our example from Fig. 3, since the header and item tuples that belong together are either all in the mains *or* deltas.

If there has not been an insert, update or delete on the dimension table in a long time, the delta of that table is empty. For inner joins, empty table components do not need to be included since they will not contribute to the result set. Therefore the subjoins #2 and #3 can be avoided. This elimination method could also be applied if there would be a greater number of involved dimension, text or configuration tables with empty deltas.

This only leaves the subjoin #1, between the header-delta, item-delta, and the main of the small dimension table. Using the semantic chaching strategy, an aggregation query

$$Q_{Agg_{HID}} = H \bowtie_{C_1} I \bowtie_{C_2} D$$

between a header H , item I and dimension table D is reduced to process the single subjoin

$$Changes(Q_{Agg_{HID}}) = H_{delta} \bowtie_{C_1} I_{delta} \bowtie_{C_2} D_{main}$$

compared to

$$ResultSet(Q_{Agg_{HID}}) = \bigcup_{(p_1, p_2, p_3) \in \mathbb{P}^3} (H_{p_1} \bowtie_{C_1} I_{p_2} \bowtie_{C_2} D_{p_3})$$

without an aggregate caching mechanism. Since all involved table components are small, the subjoin can be executed with little effort.

The concept of the semantic join can also be applied to extendable business objects such as sales orders, with item tuples possibly being added to an existing header tuple at a later point in time. In that case we additionally have to include the subjoin matching header-main, item-delta, and the mains of the smaller static tables.

For static business objects, the semantic join always only executes one subjoin using the header-delta, item-delta and dimension-, text- and configuration-table-mains. Next to the schema usage characteristics it requires a different method of handling the merge process as outlined in the following subsection.

4.6 Merge

The incremental maintenance of the aggregate cache takes place during the online merge process which propagates the changes of the delta storage to the main storage. When employing a semantic join between a header and an item table, there are two ways to merge. One way is to synchronize the merge of both tables. This way the tuples that match the join condition will always be all in

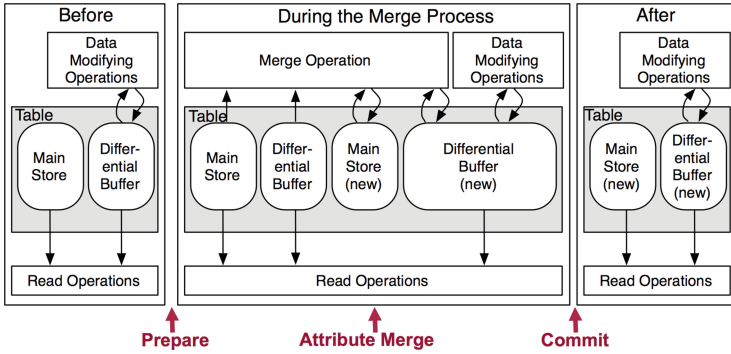


Fig. 4. Stages of the delta merge process.

the delta *or* main storage. Figure 4 shows the three phases of a merge process [11]. Specifically, the prepare steps that switch inserts to run into new blank delta storages need to happen in between the same two SQL queries. Depending on the database architecture, this may be a challenging implementation task without the introduction of a lock that cues transactional queries.

On the other hand, the header and item table could be merged independently by maintaining the aggregate cache at the same time. For static business objects the incremental update would need to process all subjoins that include the delta that is being merged. If we merge the header-delta of our example from Fig. 3, that would be #1. When rerunning a query after the header table has been merged, the item tuples of the merged header would not find a join partner using the inner join. But they also should not find a partner since they are already considered in the cached result set. For extendable business objects, the incremental update would only be done for merging the item table, since the semantic join also processes the item-delta/header-main subjoin for every analytical query.

5 Annotating Enterprise Application Characteristics

In this section we list schema usage characteristics the database requires to process the semantic join from Sect. 4.5. For each information aspect we introduce a number of ways of explicit and implicit character, how the database could obtain that information. In this Section we limit our annotation examples to static business objects, even though similar methods can be used to extendable business objects.

5.1 Empty Delta

To avoid the subjoins with the deltas of dimension, text, and configuration tables, the database engine needs to know that they are empty. That simple check should be a trivial implementation for most database architectures.

5.2 Associations

The caching engine needs to know which table attributes are used as join condition to the key of other tables. There are three methods with different strengths and weaknesses.

First, *foreign keys* could be defined on database level during the design time of the schema [19]. They are a well established mean in many database systems. A column is marked to match the key of another table. New inserts, updates and deletes are checked for data integrity of defined foreign keys. The checking mechanism may decrease transactional throughput performance.

Another way would be to use a *domain specific language (DSL) to model objects on database level*. The database would create the `CREATE TABLE` statements from the abstract data modeling language. The DSL supports syntax to explicitly express associations between objects. Listing 1.1 shows an example syntax similar to the CDS-DDL³ from SAP HANA [1]. An `AccountingItem` can, but does not have to be associated with a `Product`.

```
entity AccountingItem{
  Product: association [0..1] of Product;
  Quantity: int}
entity Product{
  Name: string}
```

Listing 1.1. DSL example to model objects and associations on DB level.

A third way would be to look at meta data repositories of present systems. Some enterprise application landscapes keep schema information in a central place. One example of those repositories is the SAP Data Dictionary. Each table column has a specific domain. Such a domain can be defined by a data type, a value range or the column of another table. The latter case indicates an association used as join condition.

5.3 Single Transaction Inserts

As explained in Sect. 3, static business objects are inserted in the context of a single transaction. There are two fundamentally different ways to communicate the insert behavior to the database. The schema could be annotated at design time or the database access could be restricted to insert entire business objects.

Design Time Annotation. During data modeling phase the designer defines how the schema will be used. This might be a challenging programming paradigm for environments where a large number of developers are involved. Application programmers might not know about the restrictions implied by the data modelers and be surprised about the errors returned by the database. The annotation could be done in two ways.

³ Core Data Services - Data Definition Language, a DSL to model objects on SAP HANA.

The *DSL for data modeling*, as introduced in Sect. 5.2 could support syntax to explicitly model a composition relationship. That relation is stronger than an association, meaning that entity *foo* consist of some entities *bar*. The composition relationship implies that they are inserted in the context of single transaction. In Listing 1.2 the AccountingHeader is composed of a number of AccountingItems.

```
entity AccountingHeader{
  FiscalYear: int;
  Items:      composition [1..*] of AccountingItem}
entity AccountingItem{
  Product:    association [0..1] of Product;
  Quantity:   int}
```

Listing 1.2. DSL example to model composition relationship.

Another way would be to slightly extend the in many databases already present concept of *SQL constraints*. Typically they are defined on a schema level, checked on SQL statement level and sometimes with a leaner execution time on transaction level. The available constraint enforcement levels need to be extended with a new *transaction* level, that explicitly checks for constraint consistency within a transaction. It checks if a transaction inserting new tuples is valid by itself. The defined foreign keys are validated among the tuples that are inserted together.

High Level APIs for Data Manipulation. Inserts into databases are typically done by using SQL commands. The database could restrict data manipulation to higher level APIs. Those commands could e.g. look like `StoreAccountingObject()`, `RegisterMaterialMovement()` or `ReleaseSalesOrder()`. In that case all information of header and item tuples would be inserted with a single command. By restricting data manipulation to such higher level APIs, inconsistent data states could also be prevented, that could otherwise be caused by improper usage of SQL commands. The application developer would access those ORM⁴ like methods directly on the database.

In the context of currently available database technology, one implementation strategy would be to do all data manipulation with *Stored Procedures* (SPs). The database would offer SPs to persist entire objects. The procedure describes in detail how the object attributes are transformed into tuples for different tables. A SP for e.g. an invoice may store a AccountingHeader tuple and multiple AccountingItem tuples in the corresponding tables.

Another way would be to use a *DSL for business object persistence and manipulation on database level*. That DSL would also only offer high-level commands as previously mentioned. One example would be the CDS-DML⁵ currently in development for SAP HANA.

⁴ Object Relational Mapper, a framework to easy access to relational databases from object oriented programming languages.

⁵ Core Data Services - Data Manipulation Language.

6 Benchmarks

In this section we evaluate the potential speedup of the semantic join (see Sect. 4.5) compared to the join not using schema usage characteristics, the caching mechanism used with a fully denormalized schema and using no caching mechanism at all.

For the evaluation we use a real customer data set of an SAP financials application of an international-operating company producing consumer goods. The schema – limited to the benchmark relevant tables and columns – looks similar to the one illustrated in Fig. 1. The data set consists of 35 million Accounting-Header tuples, 310 million AccountingItem tuples and the text tables have each less than 2000 entries.

We modeled a mixed OLTP/OLAP workload, based on input from interviews with that customer. The analytical queries simulate multiple users, using a profit and loss statement (P&L) analysis tool. The SQL statements calculate the profitability for different dimensions like product category and subcategory (as mentioned in Sect. 3) by aggregating debit and credit entries. Listing 1.3 shows a simplified sample query that calculates how much profit the company made with each of its product categories. We simulate a drill down into the (P&L) by applying a specific dimension value as filter and then grouping by another dimension.

```
SELECT pc.Name AS Category, SUM(i.Price) AS Profit
FROM AccountingHeader AS h,
      AccountingItem AS i,
      ProductCategory AS pc
WHERE i.AccountingHeaderID = h.AccountingHeaderID
      AND i.CategoryID = pc.CategoryID
      AND pc.Language = 'ENG'
GROUP BY i.CategoryID;
```

Listing 1.3. Simplified benchmark sample query.

All benchmarks are run on a server with 64 Intel Xeon QPI⁶ enabled processor cores and 1 TB of RAM running SansoucciDB [5], an in-memory column-oriented research database.

6.1 Delta Size

The speed up of the aggregate caching mechanism greatly depends on the number of records in the delta storage. The smaller the delta in respect to the main storage, the less tuples need to be aggregated when rerunning cached queries. How large the peak delta size is just before merging, depends on the insert rate and how long it takes to merge the table.

Figure 5 shows the speedup factor of the different caching strategies outlined in Sect. 4.2 compared to the caching mechanism running on a single, denormalized table. For the denormalized caching, the speedup is calculated by comparing

⁶ Quick Patch Interconnect, a direct communication system for processor cores that replaces the Front Side Bus (FSB).

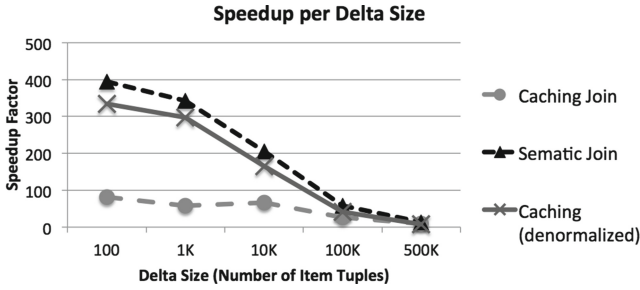


Fig. 5. Aggregation with header-item join benchmark.

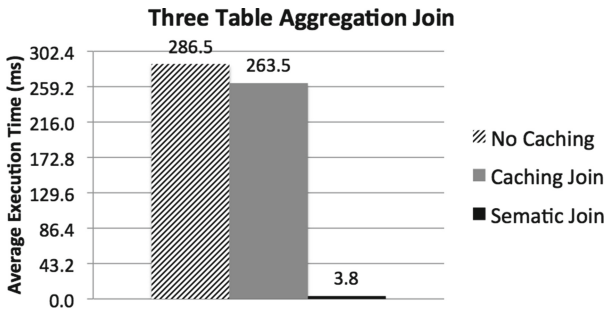


Fig. 6. Benchmark for aggregation queries joining header, item, and one dimension table.

it to the runtime on the denormalized table without caching. For this specific benchmark we only use a two table join between the header and item table. In that case the strategy not leveraging enterprise application characteristics also performs better by magnitudes since it never has to do the header-main/item-main subjoin.

The semantic join enables a speedup of greater than 200 for item deltas smaller than 10 thousand tuples and greater than 50 with less than 100 thousand tuples. Even for larger deltas with half a million entries, cached queries are calculated thirteen times faster than without caching (0.12 compared to 1.58 s).

6.2 Three Tables

For an aggregation query joining three tables as illustrated in Fig. 3, the caching mechanism has to join the large header-main and item-main (see Sect. 4.2). In this benchmark we use deltas with 50,000 items and their corresponding header tuples. The dimension table consists of 150 entries. Figure 6 shows the importance of utilizing schema usage characteristics once there are three or more tables involved. The analytical queries of the analyzed customer typically involve three to seven tables. Since the semantic caching strategy only joins rather small table

components, its execution time remains faster by an order of magnitudes, even if more tables are involved.

7 Conclusions and Future Work

With growing requirements on data analysis, the aggregate cache enables IMDBs to handle an even higher aggregation query throughput in enterprise system environments with mixed workloads. However, with queries joining two or more tables, the benefit of the aggregate cache is reduced as the needed incremental view maintenance is very expensive.

Our analysis of enterprise applications revealed several patterns with respect to schema design and resulting workloads. Most importantly among them, it is a very common practice to split business objects into a header and item table and schemas having many small rather static tables. These patterns can be leveraged to reduce the incremental view maintenance and run more efficient aggregate caching strategies. Especially for small delta sizes, they enable a speed-up by order of magnitudes.

With having a clear understanding of the speedup potential of the caching mechanism for aggregation queries joining tables, one direction of future work is to predict the runtime improvements for a columnar IMDB with a main-delta architecture. Based on cardinalities of main and delta partitions, unique value count, filter selectivity, and possibly other metrics, a cost model of the cache admission policy should decide what aggregate queries with joins are most valuable to be cached.

References

1. Färber, F., Cha, S.K., Primsch, J., Bornhövd, C., Sigg, S., Lehner, W.: SAP HANA database: data management for modern business applications. In: SIGMOD (2011)
2. Grund, M., Krüger, J., Plattner, H., Zeier, A., Cudre-Mauroux, P., Madden, S.: Hyrise: a main memory hybrid storage engine. In: VLDB, pp. 105–116 (2010)
3. Kemper, A., Neumann, T., Informatik, F.F., München, T.U.: Hyper: a hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In: ICDE, D-Garching (2011)
4. Plattner, H.: A common database approach for OLTP and OLAP using an in-memory column database. In: SIGMOD, pp. 1–2 (2009)
5. Plattner, H.: SanssouciDB: an in-memory database for processing enterprise workloads. In: BTW (2011)
6. Smith, J.M., Smith, D.C.P.: Database abstractions: aggregation. *ACM Commun.* **20**, 405–413 (1977)
7. Srivastava, D., Dar, S., Jagadish, H., Levy, A.: Answering queries with aggregation using views. In: VLDB (1996)
8. Gupta, A., Mumick, I.S.: Maintenance of materialized views: problems, techniques, and applications. *IEEE Data Eng. Bull.* **18**, 3–18 (1995)
9. Müller, S., Butzmann, L., Höwelmeyer, K., Klauck, S., Plattner, H.: Efficient view maintenance for enterprise applications in columnar in-memory databases. In: EDOC (2013)

10. Müller, S., Plattner, H.: Aggregates caching in columnar in-memory databases. In: 1st International Workshop on In-Memory Data Management and Analytics (IMDM), in conjunction with VLDB (2013)
11. Krueger, J., Kim, C., Grund, M., Satish, N., Schwalb, D., Chhugani, J., Plattner, H., Dubey, P., Zeier, A.: Fast updates on read-optimized databases using multi-core CPUs. In: VLDB (2012)
12. Brewer, E.A.: Towards robust distributed systems. In: PODC (2000)
13. Buneman, O.P., Clemons, E.K.: Efficiently monitoring relational databases. *ACM Trans. Database Syst.* **4**, 368–382 (1979)
14. Blakeley, J.A., Larson, P.A., Tompa, F.W.: Efficiently updating materialized views. In: SIGMOD, pp. 61–71 (1986)
15. Bello, R.G., Dias, K., Downing, A., Feenan, Jr., J.J., Finnerty, J.L., Norcott, W.D., Sun, H., Witkowski, A., Ziauddin, M.: Materialized views in oracle. In: VLDB, pp. 659–664 (1998)
16. Zhou, J., Larson, P.A., Elmongui, H.G.: Lazy maintenance of materialized views. In: VLDB, pp. 231–242 (2007)
17. Gupta, H., Mumick, I.S.: Incremental maintenance of aggregate and outerjoin expressions. *Inf. Syst.* **31**(6), 435–464 (2006)
18. Larson, P.A., Zhou, J.: Efficient maintenance of materialized outer-join views. In: 2007 IEEE 23rd International Conference on Data Engineering, pp. 56–65. IEEE (2007)
19. Garcia-Molina, H., Ullman, J.D., Widom, J.: *Database Systems: The Complete Book*, 2nd edn. Prentice Hall Press, Upper Saddle River (2008)

MaiterStore: A Hot-Aware, High-Performance Key-Value Store for Graph Processing

Dong Chang^(✉), Yanfeng Zhang, and Ge Yu

Northeastern University, Shenyang, 110819 Liaoning, China
cdaspirin@gmail.com,
zhangyf@cc.neu.edu.cn,
yuge@mail.neu.edu.cn

Abstract. Recently, many cloud-based graph computation frameworks are proposed, such as Pregel, GraphLab and Maiter. Most of them exploit the in-memory storage to obtain fast random access which is required for many graph computation. However, the exponential growth in the scale of large graphs and the limitation of the capacity of main memory pose great challenges to these systems on their scalability.

In this work, we present a high-performance key-value storage system, called MaiterStore, which addresses the scalability challenge by using solid state drives (SSDs). We treat SSDs as an extension of memory and optimize the data structures for fast query of the large graphs on SSDs. Furthermore, observing that hot-spot property and skewed power-law degree distribution are widely existed in real graphs, we propose a hot-aware caching (HAC) policy to effectively manage the hot vertices (frequently accessed vertices). HAC can conduce to the substantial acceleration of the graph iterative execution. We evaluate MaiterStore through extensive experiments on real large graphs and validate the high performance of our system as the graph storage.

Keywords: Graph store · Key-value store · Hot-aware cache · SSDs · Maiter

1 Introduction

In the era of big data, a huge amount of graph data is being collected, e.g., Twitter follow graph, Amazon consumer-purchase-record bipartite graph, LinkedIn social graph etc. A lot of state-of-the-art graph computation frameworks [9, 17–21, 27, 29] are proposed to process and mine these big real-world graphs. In many graph algorithms, the computation of a graph vertex only depends on its neighbors' values. It could involve a substantial amount of non-contiguous access when

This work was partially supported by National Natural Science Foundation of China (61300023, 61272179), Fundamental Research Funds for Central Universities (N120416001, N120816001), China Mobil Labs Fund (MCM20122051), and MOE-Intel Special Fund of Information Technology (MOE-INTEL-2012-06).

accessing values of the neighbors. Therefore, most of these graph computation frameworks [9, 18, 19, 21] are designed to maintain the graph data in memory for fast random access.

However, real-world graphs might be too large to fit into memory. Even though high-end servers with high-capacity memory are available in recent years, it may not be a typical solution for the widespread adoption of memory storage. The steep price and relatively small capacity of DRAM will continue to be a concern for a long time. On the other hand, many big data processing systems [1, 2] and several graph computation systems [20, 29] utilize external storage devices such as HDDs (hard disk drivers) to store these large graphs, but the costly I/Os correspondingly render these systems' performance inefficient. Thus, in graph computation frameworks, we believe that the graph-structured data should not be simply stored in the disk-based storage or the memory storage, but instead be placed in the fittest storage position while exploiting both advantages of disk and memory.

To address this challenge, we introduce **MaiterStore** for storing large graphs by using the state-of-the-art solid state drives (SSDs). We observe that many graphs are stored in *in-memory key-value tables*, such as Maiter [21]. The states of the graph vertices/edges are required to update many times during the computation. On the other hand, the graph structure, e.g. adjacency list, is immutable during the whole iterative computation. In addition, the size of graph-structured data is usually far larger than the state data of graph. MaiterStore keeps the frequently updated volatile graph state data in main memory for fast random access, while store the read-only immutable graph-structured data on SSDs. This is mainly because SSDs are capable of gaining better I/O performance than the HDDs, and are cheaper than DRAM. Using SSDs to store the immutable graph-structured data will bypass the I/O bottleneck of HDDs.

MaiterStore equips the graph data with key-value pairs and has an efficient *page manager* to manage these key-value pairs on SSDs. In addition, considering the graph data access pattern, MaiterStore adopts a page-based *prefetching buffer* in memory, which prefetches the anticipating graph-structured data from SSDs. The prefetching buffer can adapt to the access pattern changes and exchange the prefetched SSDs pages dynamically. Specifically, the graph computation usually exhibits skewed access patterns where some vertices' edges are accessed frequently and excessively. Based on this observation, we design a *hot-aware cache (HAC)* in memory to cache the hot vertices and their edges from SSDs intelligently. HAC can conduce to the substantial acceleration of the graph processing.

We summarize our main contributions as follows:

- MaiterStore, a high-performance key-value store for large-scale graph processing.
- HAC, a hot-aware cache for caching the hottest part of immutable graph-structured data during the graph computation.
- Implementation of MaiterStore and evaluation of its performance with graph applications on a local cluster as well as Amazon EC2.

The rest of this paper is organized as follows. Section 2 introduces the graph computation framework—Maiter and SSDs characteristics. Then we present the system design of MaiterStore, and its components as well as APIs in Sect. 3. The experimental results are shown in Sect. 4. We outline the related work in Sect. 5 and conclude the paper in Sect. 6.

2 Preliminaries

2.1 Introduction to Maiter

In this paper, we will present MaiterStore to support a recently proposed in-memory graph processing framework Maiter [21]. Note that, although MaiterStore is currently designed to make Maiter memory-efficient, the techniques used are also suitable for other graph processing systems if they maintain graph data based on key-value tables. The APIs can be easily extended and be compatible with other systems.

Maiter is built on the basis of a novel computation model, called delta-based accumulative iterative computation. By exploiting this new model, Maiter shows better performance than many of the state-of-the-art graph processing systems. Maiter follows a master-slave architecture for distributed computation. For graph processing, Maiter firstly splits the input graph data into multiple partitions and sends to different workers.

On each worker, the received graph partition is loaded in an in-memory key-value store, *state table*, as shown in Fig. 1. The first field represents the id of a vertex; the 2nd, 3rd, 4th fields correspond to the vertex state, which are updated during computation; the last one is vertex adjacency list, which is always immutable and static but frequently accessed during the iteration. We will explore this mutable/immutable property of graph data to design our graph storage. In addition, Maiter exploits a *priority scheduling* policy, which identifies the key vertices during graph processing and assigns them higher execution priority. Consequently, these vertices with higher priority are the hot-spot of the graph. We will design a caching mechanism to identify and cache these hot vertices for fast access.

2.2 SSD Characteristics

SSDs have the same purpose as conventional mechanical hard drives, but there is one crucial difference: they are electronic devices without any mechanical

vid	v	Δ	priority	adjacency list
⋮	⋮	⋮	⋮	⋮

Fig. 1. In-memory state table in Maiter

moving parts. Unlike HDDs, SSDs do not store data on spinning platters, but use flash memory instead. Therefore, SSDs are capable of providing *fast random access speed*, and the time to access data is almost proportional to the amount of data irrelevant for the physical locations of data in SSDs. As opposed to disk, accessing two sequential pages is no faster than accessing two random pages.

The basic I/O unit of SSDs is a *page*, typically 4 KB in size. These pages are organized into blocks which are between 256 KB or 1 MB in size. However, when data stored in the SSDs will be updated in place, SSDs have to perform an erase operation which cannot erase selectively on a specific data record, but on an entire block containing the data record. This leads to redundant effort and further slows the speed of update operation. MaiterStore is a hybrid storage system using SSDs. Considering the *erase-before-write* [11] limitation, MaiterStore is designed to bypass this limitation and optimize SSDs I/Os.

3 MaiterStore Design

In this section, we present the system architecture of MaiterStore and the related techniques. The design of MaiterStore is driven by the Maiter’s memory-inefficient state table.

In Maiter, gigantic graph data exhausts the limited memory of each worker. In order to get rid of the bottleneck caused by limited memory, we ought to *destage the huge static graph-structured data to SSDs*, reducing pressure on the memory. After migration, the state table in Maiter could become relatively smaller than before. Meanwhile, using SSDs could not lead to a sharp slowdown of random access compared to the memory. From the fact that every vertex in graph storage is an index to its neighboring vertex, thus we regard the vertex and its adjacency list as static key-value pair storing the SSDs.

3.1 System Overview

Overall, MaiterStore is composed of the following key components, as shown in Fig. 2.

Page Manager: The page manager serves to equip the static graph-structured data as key-value pairs and store them on SSDs in units of page. Meanwhile, it selects the first key for each page as *page number* and constructs these page numbers as the *BST index*. We will describe this in Sect. 3.2.

Prefetching Buffer: The prefetching buffer is a page-based storage structure that is maintained in memory. It is used for fetching the to-be-accessed pages from SSDs early enough so that they are available in the buffer when required. More details will be described in Sect. 3.3.

Hot-aware Cache (HAC): This is a fixed-size read cache of key-value entries that is maintained in the memory. We use a novel replacement strategy to evict the key-values pairs when inserting items into the full HAC. We will present this in Sect. 3.4.

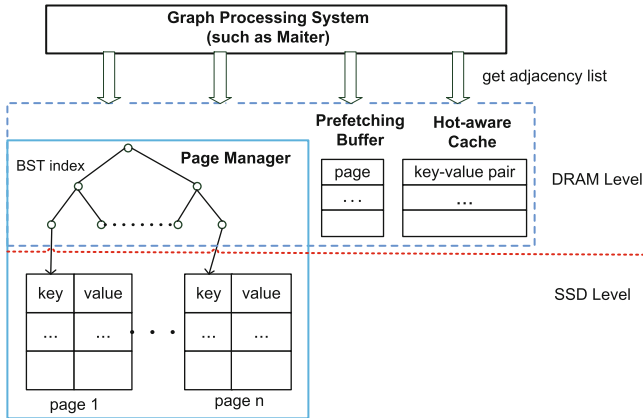


Fig. 2. MaiterStore architecture on each worker

For better understanding how these components work together, we describe a simple example, a lookup operation of the adjacency list.

In this example, we divide the graph of six vertices into three equal pages. These vertices are flash resident and paged in and out of the memory as needed. Assume also that prefetching buffer and HAC are empty from the beginning. When graph processing systems, such as Maiter, compute the PageRank of vertex 1, requiring to query the adjacency list L for vertex 1, the HAC is consulted first. If L is in the HAC, MaiterStore returns the cached adjacency list. If L is not in the HAC but it is in the prefetching buffer, it is read into the HAC from the prefetching buffer. If L is neither in the HAC nor in the prefetching buffer, the page manager firstly finds the page number `page0` via the BST index to locate the position of the adjacency list on SSDs, and then fetches the corresponding page into the prefetching buffer and inserts the adjacency list into the HAC. As shown in Fig. 3. If the HAC is full when a new adjacency list is read in, the HAC must evict an item according to its replacement policy. In the following, we will outline the three components in detail.

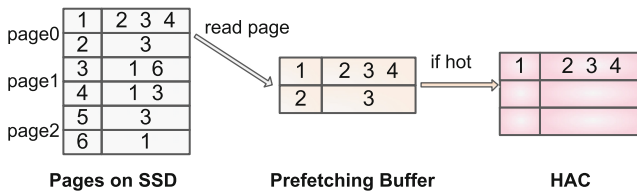


Fig. 3. Example of the lookup operation on MaiterStore

3.2 Page Manager

Before iterative computation, Maiter splits the input graph data into multiple partitions and sends to different machines. For each worker machine, the page manager is responsible for decomposing the graph partition into contiguous equal-size pages. Then these pages are sequentially logged in SSDs. The log-structured manner and its benefits have been reported in earlier work [12, 15, 30]. Typically, a page is a collection of graph vertices and their outgoing edges; in other word, edges are indexed for their source vertices. For each page, we choose the key of first pair as *page number* and the page manager saves it on the local machine. Every key-value entry in the page is associated with this page number. Before iterative execution, the page manager constructs these page numbers a BST (balanced search tree) index.

BST index is a simple but efficient in-memory index. We can quickly locate the page by binary search in logarithmic time. Also note that the BST index is a sparse and memory-efficient data structure since we only maintain the first key of the page.

Since the key-value pairs on SSDs are maintained implicitly in a sorted order, we can easily obtain the page number via the BST index and locate the retrieved page. For example, the request of a key-value pair (k, v) is sent to the page manager, the page manager can readily map the key k into the page number via BST index. And then, the page manager can quickly find the corresponding page on SSDs according to the page number.

The page in our key-value system is the basic unit that can be read or written. Generally, the page size must be an integral multiple of the SSDs default page size 4 KB to keep I/O aligned. The page size is set as 8 KB by default in MaiterStore. We will further study the effect of page size on performance in Sect. 4.2.3.

For page management, the page manager adopts a *write-once-read-many* [7] model. This model could not only enable high throughput of data access but also potentially avoid SSDs write pitfalls.

3.3 Prefetching Buffer

Prefetching is aimed at making data available in the memory before it is requested, thereby hiding the effect of latency resulting from poor reference locality. However, prefetching is not cost-free and it has to manage the prefetched data. If the prefetched data is not subsequently used, it obviously reduces the efficiency of system. To maximize the performance of system, the prefetching technique needs to predict the access pattern, minimizing the number of useless prefetches.

In Maiter, all local vertices stored in state table are processed iteratively round by round. In each iteration, the vertices are processed in the order that they are stored in the state table. Further, these vertices' edges are stored in SSD pages. We may pin a set of pages to memory for prefetching buffer.

In MaiterStore implementation, we adopt the following prefetching method: when the read of the key-value pair is not present in memory, the page manager requests the page containing the key-value entry and several pages adjacent to

that page with multi-threads in advance. In general, the number of prefetching pages and prefetching buffer size are application configurable by users in MaiterStore.

The prefetching buffer can be managed in many way. Finding the locality of graphs is still a big challenge [31], so the prefetching buffer has no information about which page will be reused if it remains in the prefetching buffer. For simplicity, we use FIFO (First In First Out) policy to evict the pages when inserting new pages into full prefetching buffer in current implementation.

In our experiment, it is also shown that prefetching technique can reduce excessive I/Os efficiently and hide its latency effect.

3.4 Hot-Aware Caching Policy

Typically, workloads of many graph computation systems exhibit excessive access skew. This is mainly because most real-world graphs obey the power law distribution [28]. For example, the social network can naturally be abstracted as a graph, where pages correspond to vertices and hyperlinks correspond to directed edges. In such real-world graph, most vertices have a relatively small degree but some outliers, such as celebrities in a social network, are much larger. This is important, because computation converges slower on these important vertices than others, and it is desirable to focus computation on them. For graph processing, the vertices with high degree are hot and accessed frequently, but others with small degree are cold and accessed infrequently. Obviously, it would make sense to reside such hot and high-degree vertices in memory. Cold vertices should be migrated to external memory such as flash to mitigate the memory pressure.

Additionally, updating a part of the important vertices selectively [20] rather than entire vertices can lead to more efficient graph computation, because not every element in the intermediate result changes at each iteration. To this end, we should focus on the hot vertices and keep such performance-critical vertices in memory.

HAC attempts to take into account the hotness of graph vertices based on the priority-based iterative execution of Maiter [21], which could avoid loading inactive vertices from SSDs. The hotness of the vertex is normally set to *the priority field* in Maiter state table. In the graph computation framework, Zhang et al. [20] proved priority scheduling: given an execution priority to vertices can potentially accelerate the convergence, because some of vertices play a decisive role in determining the final converged outcome. Under these conditions, we should keep these decisive vertices in cache. Our hot-aware cache occupies small available portions of memory, but holds the top hot key-value pairs and does not incur high penalties to main memory. Figure 4(a) shows the hot-aware cache with the triple(hotness, vid, adjacency list).

Conventional LRU (Least Recently Used) replacement strategy evicts the objects in cache that have not been accessed for the long time, and it only considers the time of the last access to objects. Unlike LRU, HAC can perform hot-aware policy to cache key-value pairs with higher hotness. Therefore, the

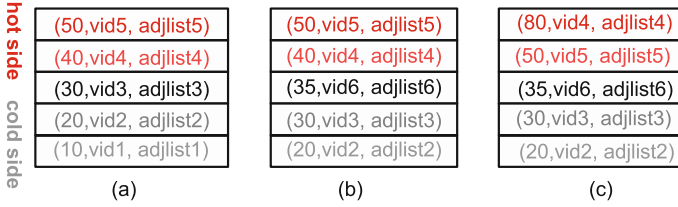


Fig. 4. (a) hot-aware cache. (b) evict the pair with lowest hotness and insert new pair with higher hotness (35, vid6, adjlist6) when miss. (c) update the hotness of the pair (80, vid4, adjlist4) when hit

relatively small memory can cache more valuable key-value pairs, which can make a big difference when Maiter executes iterative computation.

Looking-up operation with a key and a hotness value h in MaiterStore needs to read the cache. Upon a miss, the current key-value pair read from prefetching buffer or SSDs will be inserted into cache after evicting another key-value pair (if cache is already full). To decide which key-value pair to evict when cache is full, HAC firstly finds the key-value pair whose hotness is lowest, say h_{lowest} . If $h > h_{lowest}$, HAC evicts the key-value pair with lowest hotness to make room for the current key-value pair in Fig. 4(b). If the key-value pair is already in cache, HAC needs to only find the pair and update the hotness as shown Fig. 4(c). With the in-memory hash table, HAC achieves $O(1)$ time for hit and eviction.

3.5 API in MaiterStore

MaiterStore is implemented in circa 1800 lines of C++ code using boost library. It currently provides Maiter with graph storage, though it could later be expanded to support other in-memory graph processing frameworks. Basic operations in MaiterStore are as follows:

```
virtual void parseKV(string line, K* vid, V* adjList) = 0;
V get(const K vid);
void put(const K vid, const V adjList);
```

where K and V are the type of key and value respectively. The interface *parseKV* is a user-defined function for converting string record to users' data structure and implemented by developers. The function *get* and *put* are two basic operations in MaiterStore. Both of them are invoked by the upper graph processing frameworks when obtaining or storing graph adjacency lists.

4 Performance Evaluation

In this section, we evaluate our system with extensive experiments. MaiterStore is equipped to Maiter to show the performance in the context of two typical applications PageRank and Connected Components.

4.1 Environment Setting

The experiments are conducted on a cluster of local machines consisting of 4 commodity machines, as well as on Amazon EC2 Cloud [3]. Each machine in the local cluster has Intel Core i3-2120 3.3 GHz CPU, 3 GB of RAM, Seagate Barracuda 500 GB hard drive and Samsung 840 Series SSD [6]. It is running Ubuntu 13.04 with the Linux kernel 3.8.8 and Ext4 file system with default configuration. The Amazon EC2 cluster involves 30 m3.xlarge nodes. Each node has 15 GB of memory, and 80 GB of SSD.

We evaluate MaiterStore using three datasets: Web-Google [5], Web-BerkStan [5] and Web Graph [4]. The dataset statistics are presented in Table 1.

Table 1. Statistics of datasets

<i>Dataset</i>	<i>Nodes</i>	<i>Edges</i>
<i>Web-Google (55 MB)</i>	916,428	6,078,254
<i>Web-BerkStan (62 MB)</i>	685,230	7,600,595
<i>Web Graph (11 GB)</i>	50,000,000	686,231,717

4.2 Experimental Results

In this section, we present our experimental results. Without loss of generality, we fetch a page into the prefetching buffer and set buffer size as 2 page.

4.2.1 Performance of MaiterStore

To study the effects of different techniques used in MaiterStore, we evaluate MaiterStore with different configurations on the local cluster. For comparison, we also configure Maiter to store graph data in memory only and in disk file only. There are totally seven settings considered for comparison: (1) in memory only (S1); (2) in SSD + Prefetching Buffer + HAC, i.e., MaiterStore(S2); (3) in HDD + Prefetching Buffer + HAC (S3); (4) in SSD + Prefetching Buffer (S4); (5) in HDD + Prefetching Buffer (S5); (6) in SSD only; (7) in HDD only. Figure 5 shows the running time of different configurations on PageRank and connected components. Due to long running time in the final two settings, we do not plot them in Fig. 5. In this experiment, we use Web-Google dataset. By comparing S1 with S2 or S3 in Fig. 5, although S1 outperforms S2 or S3, our dataset is much smaller than the capacity of memory. S1 is not scalable for processing huge amounts of the graph data in real applications with limited memory.

Interestingly S2 and S3 have the same running time approximately. This is not surprising, since MaiterStore is not read-intensive for SSD or HDD and most of read operations are cached in the prefetching buffer or the HAC, which causes the read operations concentrating on the hot vertices and hides the high I/O cost. In Sect. 4.2.4, we show that the reads for SSD or HDD are rather few in number. In addition, in such setting most or all of the hot vertices of dataset fit into the HAC, and there is no need to read data from SSD or HDD.

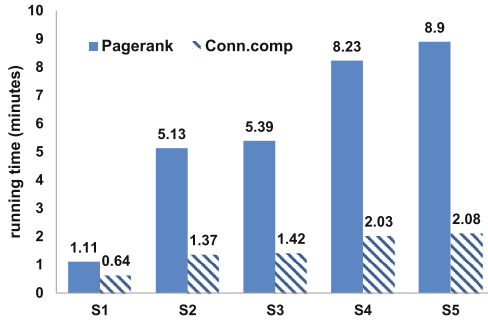


Fig. 5. Comparison of running time under different settings.

S4 or S5, by contrast, only caches the neighboring vertices and may not cache the hot vertices, and this causes miss ratio increased. We will study the effect of HAC further in Sect. 4.2.4.

In SSD or HDD only settings, we first locate the page and read it from the SSD or HDD, and then find the corresponding key-value without caching and buffering. In PageRank, the running time in SSD only setting is more than 13 times greater than MaiterStore (SSD + Prefetching Buffer + HAC) and the running time in HDD only setting is more than 17 times. This also shows the performance of MaiterStore is affected by the prefetching buffer and hot-aware cache closely.

Based on the above experiments, we can validate that, although the performance of MaiterStore has a certain gap compared with in-memory only, MaiterStore can process much bigger graphs, as it is not limited by the capacity of DRAM. It also shows that MaiterStore performs well on both SSD and HDD.

4.2.2 Scalability of MaiterStore

To show the scalability of MaiterStore under large-scale distributed environment, we conduct connected components on dataset Web Graph using EC2 cluster. Figure 6 shows the running time of MaiterStore when the number of workers is

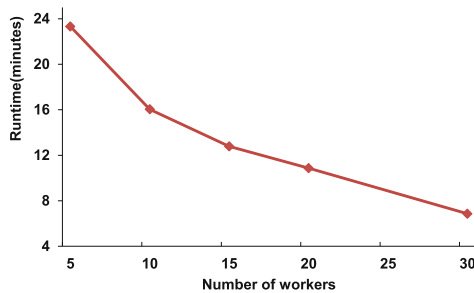


Fig. 6. Connected components: varying number of workers on EC2 cluster

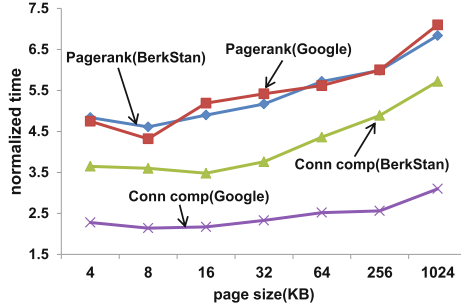


Fig. 7. Effect of page size on performance

varied. We can see that, the drop from 23.32 to 6.86 min using 6 times as many workers represents a speedup of about 4. It demonstrates that for MaiterStore the runtime decreases linearly in the worker size.

4.2.3 Effect of the Page Size

Page size is a vital parameter in MaiterStore. The setting of page size determines the granularity of buffer unit, which has effect on prefetching advantage. A large page size is desirable for reducing index size. On the other hand, a small page size can effectively eliminate reading unnecessary key-value pairs, since the page may contain both hot and cold key-value pairs. To specify the page size, the page size must be an integral multiple of the 4 KB, the default page size of the SSD.

We compare performance of different page sizes in stand-alone environment, including the Web-Google dataset and Web-BerkStan dataset in Fig. 7. We only cache 10% vertices in HAC. We get the results with various page sizes and show the execution time normalized to that of running on the memory-only setting. We can see that with the large page size, the running time increases significantly, because both hot and cold vertices may co-exist in a page, whereas sometimes we need the hot data only, which causes the unnecessary reads and prolongs the runtime. For the small page size (4 KB), the running time is larger than that with page size (8 KB). This is mainly because sometimes Maiter requires to compute a wide set of vertices on different pages, whereas small page contains less vertices than large page, thus causing the miss rate to increase.

The experiment also illustrates that choosing a proper page size can optimize performance. In MaiterStore, we use a page size of 8 KB in default.

4.2.4 Comparison with the HAC VS. FIFO VS. LRU

In this section, we compare the effect of different caching techniques in stand-alone environment using the Web-Google dataset. LRU and FIFO could not cache any vertices without any settings, because they are prone to the recently accessed objects. In the experiment, we filter the vertices with smaller hotness in

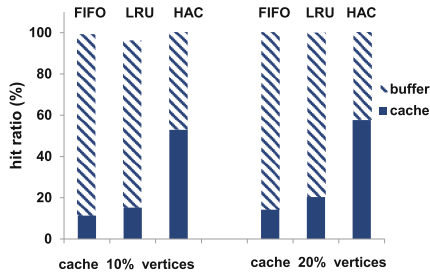


Fig. 8. Effect of cache size under HAC, FIFO and LRU

FIFO and LRU. That is, if the hotness of vertex exceeds a predefined threshold, we cache such the vertex. However, in HAC, we have no such setting.

Figure 8 shows the experimental results while varying the cache size. Even though all the cache policies exhibit better performance as the cache size increases, the HAC is more sensitive to the size of cache than the others. On the other hand, the hit ratio of HAC is more than threefold as high as the two other cache policies. In addition, the hit ratio of buffer is 97.2 % when cache is zero, and when cache has 10 % vertices size, the total hit ratio exceeds 99.97 %. It shows that below 0.03 % retrieving data is obtained by accessing SSDs.

4.2.5 Effect of the HAC and Prefetching Buffer

Finally, we evaluate the effect of buffer and cache in stand-alone environment using the Web-Google dataset. The results are shown in Fig. 9. The X axis shows the time interval of execution time, and the Y axis shows the hit times during the corresponding time interval. As iteration goes on, the hit times are increased significantly and our HAC plays an increasing important role. The hit times in the last time interval are declined because the execution is terminated during this time interval and corresponding the running time is less than 60s. It shows HAC is gradually caching the hot and performance-critical vertices.

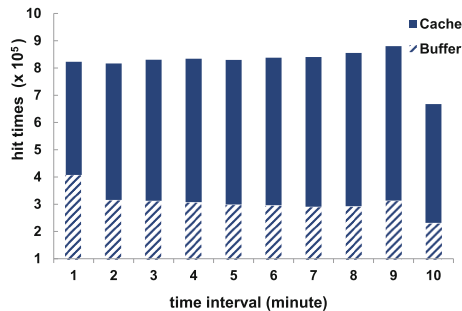


Fig. 9. Impact of prefetching buffer and HAC

5 Related Work

Recently, several key-value storage systems have sprung up in conjunction with flash storage such as BufferHash [22], SILT [24], FlashStore [23] and FAWN [12] etc. FAWN [12] is a power-efficient cluster architecture for data-intensive computing and uses SSDs as a replacement for HDDs. BufferHash [22] builds a content addressable memory system using flash for networking applications. It keeps the key-value pairs in the hash table and flushes the key-value pairs to flash when buffer is full. FlashStore [23] is a persistent key-value store using flash as a non-volatile cache between RAM and HDDs. SILT [24] is a memory-efficient key-value store by using partial-key cuckoo hashing and entropy-coded tries to reduce the per-key memory consumption. These key-value storage libraries are considered inserting SSD as another layer in the storage hierarchy to improve existing application performance. In contrast, MaiterStore treats the SSDs as an extension of the memory and pushes the SSDs upward in the memory hierarchy. Thus, using MaiterStore, existing shared-memory graph computation framework can easily and transparently enlarge the memory capacity to hundreds of gigabytes. Furthermore, by adopting the write-once-read-many model, we can easily eliminate many challenges associated with designing and implementing high-performance key-value store.

Apache Hadoop [1] is an open-source reincarnation of Google’s MapReduce [26]. It mainly consists of two components: Hadoop Distributed File System (HDFS) [7] and MapReduce framework. Yet HDFS is inappropriate for storing the structured graph data in its current implementation, and MapReduce is inefficient for processing graph iterative computation. In contrast, MaiterStore can be used as the structured graph storage for efficiently supporting the Maiter graph processing. Moreover, MaiterStore has faster access speed than HDFS, because almost all the data are accessed in memory.

GraphChi [27] is a disk-based system on a single machine that can efficiently perform advanced computation on billion-node graphs. However, if there exists excessive iterations, GraphChi may involve a high number of I/O operations when updated edges are flushed to disk at each iteration. Whereas, MaiterStore aims to mitigate the memory pressure with a simple design by separating huge graph data from memory. Furthermore, we need not worry about high I/O latency, because the graph data on SSDs is non-volatile and read-only, and MaiterStore performs a minimal portion of data reads on SSDs.

Spark [18], Piccolo [25] and GraphLab [9,16] are the graph iterative computation frameworks and they keep graph data in memory to achieve high computation speed. However, for huge graphs that do not fit in memory, such share-memory approach is constrained. Unlike these in-memory frameworks, MaiterStore separates the static graph-structured data from the memory, and mitigates the memory pressure. We need not worry about the Maiter performance of a sharp slowdown after using MaiterStore as the graph storage, because in MaiterStore the times of accessing SSDs count a surprisingly small percentage of all the data access.

6 Conclusion

This paper presents the design and evaluation of MaiterStore. MaiterStore is specialized for large-scale graph processing framework in the cloud. For efficiently managing the graph data, we separate the immutable key-value pairs from the in-memory state table in Maiter and store them on SSDs. In order to reduce the number of the SSD read operations and accelerate the graph algorithm execution, we adopt the page-based prefetching technique for buffering the to-be-queried data, and propose a hot-aware cache (HAC) for caching the hot and performance-critical data on SSDs. Our results show that MaiterStore is able to support graph processing more efficiently. Our ongoing work aims at extending MaiterStore with a more general I/O model and expects to apply MaiterStore to other in-memory graph processing frameworks beyond Maiter.

References

1. Hadoop. <http://hadoop.apache.org>
2. Hama. <http://hama.apache.org>
3. Amazon EC2. <http://aws.amazon.com/ec2/>
4. Web Graph. <http://lemurproject.org/clueweb09/>
5. Stanford dataset collection. <http://snap.stanford.edu/data>
6. Samsung SSD. <http://www.samsung.com/cn/business/business-products/ssd-card>
7. HDFS. <http://hadoop.apache.org/core/docs/r0.16.4/hdfsdesign.html>
8. Chen, F., Koufaty, D., Zhang, X.: Hystor: making the best use of solid state drives in high performance storage systems. In: Proceedings of ICS, pp. 22–32 (2011)
9. Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., Hellerstein, J.M.: Distributed graphlab: a framework for machine learning in the cloud. PVLDB **5**(8), 716–727 (2012)
10. Hu, Y., Jiang, H., Feng, D., Tian, L., Luo, H., Zhang, S.: Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity. In: Proceedings of ICS, pp. 96–107 (2011)
11. Lee, S.W., Moon, B., Park, C., Kim, J.M., Kim, S.W.: A case for flash memory SSD in enterprise database applications. In: Proceedings of SIGMOD, pp. 1075–1086 (2008)
12. Andersen, D., Franklin, J., Kaminsky, M., Phanishayee, A., Tan, L., Vasudevan, V.: FAWN: a fast array of wimpy nodes. In: Proceedings of SOSP, pp. 1–14 (2009)
13. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. **30**(1–7), 107–117 (1998)
14. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: a distributed storage system for structured data. In: Proceedings of OSDI, pp. 205–218 (2006)
15. Rosenblum, M., Ousterhout, J.K.: The design and implementation of a log-structured file system. ACM Trans. Comput. Syst. **10**(1), 26–51 (1992)
16. Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., Hellerstein, J.M.: Graphlab: a new framework for parallel machine learning. In: Proceedings of UAI, pp. 340–349 (2010)

17. Malewicz, G., Austern, M.H., Bik, A.J.C., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: Proceedings of SIGMOD, pp. 135–146 (2010)
18. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: HotCloud (2010)
19. Shao, B., Wang, H., Li, Y.: Trinity: a distributed graph engine on a memory cloud. In: Proceedings of SIGMOD, pp. 505–516 (2013)
20. Zhang, Y., Gao, Q., Gao, L., Wang, C.: PrIter: a distributed framework for prioritized iterative computations. In: Proceedings of SOCC, pp. 1–14 (2011)
21. Zhang, Y., Gao, Q., Gao, L., Wang, C.: Maiter: an asynchronous graph processing framework for delta-based accumulative iterative computation. In: IEEE Computer Society (2013)
22. Anand, A., Muthukrishnan, C., Kappes, S., Akella, A., Nath, S.: Cheap and large CAMs for high performance data-intensive networked systems. In: Proceedings of NSDI, pp. 433–448 (2010)
23. Debnath, B., Sengupta, S., Li, J.: FlashStore: high throughput persistent key-value store. In: Proceedings of VLDB, pp. 1414–1425 (2010)
24. Lim, H., Fan, B., Andersen, D.G., Kaminsky, M.: SILT: a memory-efficient, high-performance key-value store. In: Proceedings of SOSP, pp. 1–13 (2011)
25. Power, R., Li, J.: Piccolo: building fast, distributed programs with partitioned tables. In: Proceedings of OSDI, pp. 1–14 (2010)
26. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Proceedings of OSDI, pp. 137–150 (2004)
27. Kyrola, A., Blelloch, G., Guestrin, C.: Graphchi: large-scale graph computation on just a PC. In: Proceedings of OSDI, pp. 31–46 (2012)
28. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the internet topology. *Comp. Comm. Rev.* **29**, 251–262 (1999)
29. Kang, U., Tong, H., Sun, J., Lin, C.Y., Faloutsos, C.: Gbase: a scalable and general graph management system. In: Proceedings of KDD, pp. 1091–1099 (2011)
30. Chen, S.: Flashlogging: exploiting flash devices for synchronous logging performance. In: Proceedings of SIGMOD, pp. 77–86 (2009)
31. Leskovec, J., Lang, K., Dasgupta, A., Mahoney, M.: Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. *Internet Math.* **6**, 29–123 (2009)

Vertical Bit-Packing: Optimizing Operations on Bit-Packed Vectors Leveraging SIMD Instructions

Martin Faust¹, Martin Grund², Tim Berning¹,
David Schwalb¹ (✉), and Hasso Plattner¹

¹ Hasso Plattner Institute, Potsdam, Germany
{martin.f Faust, tim.berning, david.schwalb,
hasso.plattner}@hpi.uni-potsdam.de

² University of Fribourg, Fribourg, Switzerland
grund@exascale.info

Abstract. Today's in-memory column stores make heavy use of bit-packed data structures in order to reduce the required amount of main-memory and to improve the performance of memory-bound algorithms by trading more CPU cycles for less data that needs to be transferred over the memory-bus.

In this paper, we propose vertical bit-packing as a slightly modified alternative compared to classic bit-packing approaches, compressing an array of integer values with a known and finite value set so that each value is stored using the minimal required amount of bits. Vertical bit-packing aims to fully exploit the data parallelism provided by the existing on-chip vector processing units of modern x86-64 CPUs as they provide speedup potentials at no additional hardware cost.

In particular, we propose Vertical Bit-Packing and Aligned Vertical Bit-Packing as an alternative to the classic approach called Horizontal Bit-Packing. We show that the proposed techniques can save between one and two instructions per decompressed value block, outperforming the classic approach in some bit-cases with up to 12%.

1 Introduction

Research and development over the past decades has led to an ongoing improvement and evolution of computer technology in regards to architecture and performance of processors, memory and storage components. Simultaneously, both the ability to and the demand for storing vast amounts of data has been increasing, entailing the necessity to manage and analyze the information in a fast and efficient way. With the price of main memory declining, allowing for systems equipped with large amounts of RAM at relatively low cost, database systems have started to migrate from disk-based operation to keeping all data in memory at all times.

However, compared to the vast amounts of persistent memory such as disks available at hardly any cost, main memory is however still a scarce and critical

resource. Keeping all data in memory therefore becomes a task of intelligent data management and compression in order to minimize the memory footprint.

As a means of reducing memory consumption, in-memory database engines such as SAP Hana [4] and Hyrise [5] facilitate light-weight compression methods like dictionary compression and bit-packing. While conserving space is important due to the aforementioned reasons, another important aspect is the divergence of processor and memory speeds. With the rate of improvements in processor speed having outpaced advances in main memory latency, memory accesses are becoming the new bottleneck [11].

Consequently, research has been targeted at optimizing data structures and algorithms in order to mitigate the effects of low latency and make efficient use of the available bandwidth [10]. But the limited bandwidth combined with main memory latency, especially in high load scenarios, requires optimization of the data structures.

A common approach to this problem is further compressing the data in order to reduce memory accesses. In-memory column-oriented databases using dictionary compression [8] make heavy use of so called bit-packing techniques [14]. Using dictionary compression, each unique value is stored in a per-column dictionary and assigned a value ID, which then is stored in the actual column vector instead of the value.

Especially enterprise data has proven to have little distinct values in most columns, such as for instance a column containing countries or even just boolean values [6]. Assuming a four byte integer being used as a value ID, this also means that only a subset of the value range representable in 4 bytes is actually needed. Bit-packing exploits that fact by storing multiple values inside the four byte block. Bit-packing thus increases the information content per byte, allowing for a reduction of memory transfers and mitigating the effects of main memory as a bottleneck. Since the packed values cannot be processed by the CPU directly and require prior unpacking, bit-packing imposes computational overhead as a downside to the reduced memory footprint.

However, with the ever-increasing performance of modern multi-core CPUs and slow improvement of memory latency, operations on packed data are becoming less costly and the utilization of better compression techniques shifts operations from being I/O-bound back to being CPU-bound. While exploiting multi-core architectures to increase database operation performance is a common approach, using a single core's on-chip vector processing units to fully exploit existing data parallelism becomes essential when striving for optimal performance on modern architectures.

The SIMD-Scan as presented in [14] provides a fast and efficient way of performing common database operations such as the full table scan and predicate evaluation on bit-packed column data using commodity CPUs' SIMD units.

In this paper, we propose Vertical Bit-Packing (VBP) and Aligned Vertical Bit-Packing (AVBP) as alternatives to the classic approach of Horizontal Bit-Packing (HBP). Both proposed approaches make slight modifications to the data layout compared to the classic approach as described in Sect. 3, with the goal of

reducing the number of required CPU instructions for decompressing the stored data and further improving the performance of operations on bit-packed data.

2 Related Work

Techniques for efficient bit compression and decompression of integer arrays are discussed by Lemire and Boytsov [7], Schlegel et al. [12] and Wilhalm et al. [14] among others. They present various compression schemes such as null suppression or Elias encoding and present approaches using SIMD implementations such as Intel SSE to improve compression, as well as decompression rates. Test results show in all cases the potential of SIMD supported parallelization. Chhugani et al. [3] present a Merge Sort algorithm that heavily utilizes SIMD to achieve notable speedup over the scalar version.

The potential of using the CPU's built-in vector processing units to increase the performance of compressed data access is discussed in [14]. The authors are focusing on dictionary compressed column data and apply fixed length compression to further reduce the space required for storage. In order to perform scans on the compressed column vector, they introduce a SIMD approach, which decompresses 4 values in a 128 bit block of compressed data at a time. In their performance evaluation, this approach revealed an average speedup of factor 2 over a highly optimized, scalar vector implementation. In [13], the authors extend their approach to support vectorized table scans even for complex predicates.

Li and Patel [9] follow a different approach. They, too, operate on fixed-length compressed value IDs of an in-memory database column vector. In contrast to [14] however, they do not focus on efficient decompression of the data but instead provide a system for efficient, in-place predicate evaluation on the compressed data. Even without SIMD support they achieve higher scan speeds than [14]. Their two approaches come at costs each, however. Their horizontal methods require an additional storage bit for each value, doubling the space for a boolean vector, which only requires one bit per value. The vertical approach requires expensive reads, potentially crossing cache-line boundaries to retrieve single values.

3 Concepts of Bit-Packing

The concept of the VBP and AVPB are based on the techniques proposed by Lemire and Boytsov [7] and Wilhalm et al. [14] and represents an alteration of the original approach with the goal of performance optimization. This is achieved through a change in the storage layout of the compressed values in memory, followed by a modification to the decompression algorithm. The combination of storage layout and decompression algorithm is referred to as bit-packing in this paper. In this section, we first describe the classic approach of HBP followed by a description of the proposed VBP and AVBP algorithms.

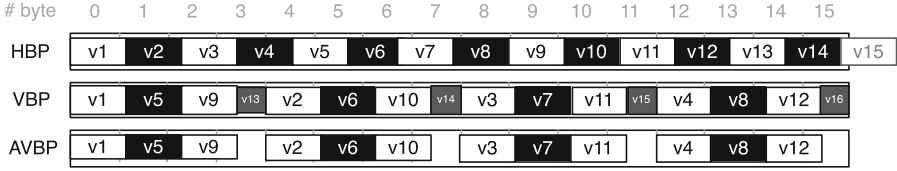


Fig. 1. Storage layouts for HBP, VBP and AVBP for bit-case $k = 9$.

3.1 Horizontal Bit-Packing

The SIMD Scan from [14] operates on fixed-length bit compressed values, which are stored in a contiguous block of memory in a consecutive fashion. This storage layout is referred to as Horizontal Bit Packing (HBP) for the sake of discussion. Analogous to the paper, an array of 64-bit integers is assumed as the storage block. Given a vector of n values $\{v_1, v_2, \dots, v_n\}$, each v_i being k bits in width, v_1 is encoded in the first k bits of the first array element, v_2 occupies the following k bits and so on. Depending on k , a single value can span across Quad-Word (QW) boundaries, potentially requiring an additional element at the end which is not entirely filled. For 64-bit integers that means in order to store n values, the required space $S_{HBP}(n)$ is calculated as follows:

$$S_{HBP}(n) = \left\lceil \frac{n * k}{64} \right\rceil \cdot 8 \text{ bytes} \tag{1}$$

For example, with $k = 9$ values v_1 through v_{14} can be fully packed into two QWs, while only $128 - (14 * 9) = 2$ bits of v_{15} can be fitted. The remainder overlaps into the next QW.

The HBP vectorized decompression algorithm extracts four 32-bit integer values simultaneously from the compressed block. In order to do so, several steps are necessary. Reference [14] describes the steps in detail and we will only summarize the steps here. In addition, Fig. 2 illustrates the basic steps.

1. One 128-bit block of compressed data is loaded into a 128-bit SIMD register using a single load instruction.
2. If a value spans across the boundaries of two consecutive 128-bit blocks, instead of step 1 a 256-bit load instruction is necessary in order to load the

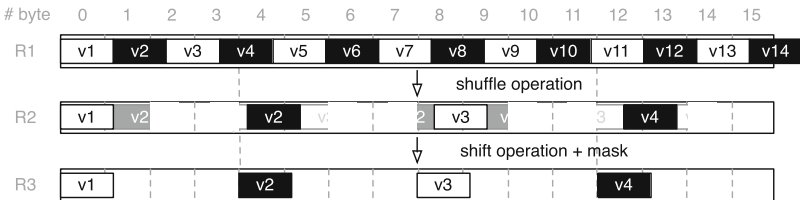


Fig. 2. Example of vectorized decompression for HBP with bit-case $k = 9$.

next block as well, followed by a shift operation to align the spanning value, thus requiring two operations for loading the data. Alternatively, more recent SIMD implementations also support unaligned access at no additional cost, which can make this step obsolete.

3. Next, four compressed values (or rather their corresponding bytes in the register) need to be copied to four separate Double-Words (DWs) in a second SIMD register. Modern SIMD implementations typically provide such an operation to perform the so called shuffle operation on all four values simultaneously.
4. If a compressed value spans across DW boundaries as for example the case with $k = 27$, an additional selective shift instruction might be necessary to correct the error.
5. Using a single SIMD shift instruction with variable offsets, the four values are positioned at the beginning of their respective DWs.
6. As a last sanitizing step a 128-bit SIMD AND instruction with an appropriate bit mask is used to mask out the remaining $32 - k$ bits in the four DWs, which were copied over by the shuffle operation.

Not including writing the decompressed values back into memory or performing further operations on them, the algorithm requires at least four and up to six operations for the extraction of values. All these steps are necessary for the extraction due to the horizontal storage layout.

3.2 Vertical Bit Packing

Instead of storing compressed values consecutively in a contiguous block of memory, VBP places a group of four values in four subsequent DW blocks of one 128-bit block of memory as this allows for a simplified decompression. Again, an n -element vector $v = \{v_1, v_2, \dots, v_n\}$ with k bits used for each value is assumed. The first 128-bit memory block will then contain $\{v_1, v_5, v_9, \dots, v_2, v_6, v_{10}, \dots, v_3, v_7, v_{11}, \dots, v_4, v_8, v_{12}, \dots\}$ with v_1, v_2, v_3 and v_4 marking the beginnings of the 32-bit sub blocks. This will store $\lfloor \frac{32}{k} \rfloor$ full values in the first block, leaving $p = 32 \bmod k$ bits at the end of each DW block. If $p < k$, the next value can only partially be fit. The remaining bits are then carried on to the corresponding DW sub block in the next 128-bit block.

Figure 1 shows how for $k = 9$ only the first 5 bits of v_{13}, v_{14}, v_{15} and v_{16} can be fitted into the first block with the rest overlapping into the next block. Similar to HBP, the values are tightly packed, resulting in roughly the same storage space necessary.

Based on the vertical storage layout, the decompression algorithm from [14] summarized in Sect. 3.1 can now be altered to take advantage of this structural change. As mentioned above, compressed values can still span across 128-bit block boundaries. Thus, loading the next block into a register might be necessary as well. Thanks to the vertical layout with four subsequent values stored in four separate DW blocks, the shuffle operation from step 3 becomes obsolete. The problems arising from shuffling values wider than 27 bits as described in step 4 is

Algorithm 1. VBP Vectorized Decompression

```

i ← 0
while i < n do
  block ← ((i/4) * k)/32
  R1 ← load128(input[block])
  offset ← ((i/4) * k) mod 32
  while offset < 32 do
    R2 ← pRShift(R1, offset)
    R2 ← pAnd(R2, maskk)
    if 32 − offset < k then ▷ overlap?
      R1 ← load128(input[block + 1])
      R3 ← pLShift(R1, k − (32 − offset))
      R2 ← pOr(R2, pAnd(R3, maskk))
    end if
    store128(R2, output[i])
    offset ← offset + k
    i ← i + 4
  end while
end while

```

subsequently eliminated as well, by default the values are effectively positioned as they would in the horizontal decompression process after step 4. Skipping these steps, now the remainder of the algorithm is unchanged. All four values are shifted and masked, leading to four decompressed 32-bit integer values per DW block.

Algorithm 1 explains the vectorized decompression for VBP more precisely. n denotes the number of packed values in the compressed vector *input*, k the bits used per value and *output* a regular 32-bit array buffer for the unpacked values to be stored in, *mask* _{k} a 128-bit bit mask with the lower k bits of each 32-bit block set to 1 and the R_i represent 128-bit SIMD registers. The functions **load128** and **store128** load and store 16-byte blocks from and back to main memory. **pRShift**, **pLShift**, **pAnd** and **pOr** are parallel SIMD methods operating on four 32-bit values in a register simultaneously. Most variables in the algorithm are constant or can be predetermined with only *block* and *offset* depending on i . Due to the optimized storage layout, the VBP decompression algorithm reduces the total number of operations for block decompression from 4–6 instructions per block down to 3–4 instructions per block.

3.3 Aligned Vertical Bit Packing

A slight variation of VBP is the AVBP approach. A further modification to the storage layout allows for a more simple decompression algorithm as outlined below. However due to the inserted padding, more space is required to store the values.

AVBP places a group of four values in four subsequent DW blocks of one 128-bit block of memory. However, as opposed to VBP, compressed values are stored

DW aligned. Again, an n -element vector $v = \{v_1, v_2, \dots, v_n\}$ with k bits used for each value is assumed. The first 128-bit memory block will then contain $\{v_1, v_5, v_9, \dots, v_2, v_6, v_{10}, \dots, v_3, v_7, v_{11}, \dots, v_4, v_8, v_{12}, \dots\}$ with v_1, v_2, v_3 and v_4 marking the beginnings of the 32-bit sub blocks. Since the beginning of each DW contains a value, there must not be values spanning across DW boundaries.

Thus, only $\lfloor \frac{32}{k} \rfloor$ values can be stored per DW and the other bits remain unused. The worst case scenario for $0 < k \leq 32$ is $k = 17$, in which case only one value can be stored per DW and 15 bits are wasted. As a result, all $16 < k \leq 32$ will require the same amount of space and are effectively equivalent to a 32-bit integer array. Since 32-bit integers are assumed to be available on the system, this renders bit compression irrelevant as far as VBP is concerned. The required amount of bytes for storing n values, denoted as $S_{AVBP}(n)$ can be calculated as:

$$S_{AVBP}(n) = \left\lceil \frac{n}{\lfloor \frac{32}{k} \rfloor \cdot 4} \right\rceil \cdot 16 \text{ bytes} \tag{2}$$

Figure 1 visualizes the storage layout for AVBP for $k = 9$. This bitcase demonstrates the padding at the end of each 4 byte block, leading to decreased storage capacity. In contrast to HBP and VBP, packing over 14 values into the 128-bit block, AVBP can only store 12 values in this particular case.

The AVBP storage layout allows to simplify the HBP decompression algorithm even more, as outlined in Fig. 3. Potential misalignment needs to be taken care of with both the HBP and VBP storage layout. AVBP in contrast guarantees by design that no overlapping values will occur due to the DW alignment. AVBP stores every four consecutive values in four consecutive DWs, resulting in a 128-bit block that is loaded into the SIMD register, thus eliminating the necessity for additional DW alignment and the loading of a second block. From this point on, the decompression process is the same for both VBP and AVBP.

In conclusion, the aligned vertical storage layout allows for at least the shuffle operation to be skipped in the decompression process and never requires additional data loads or alignment of spanning values, reducing the number of operations to a fix three. All other aspects of the SIMD-Scan algorithm are not affected by this change, suggesting a likely performance improvement as far as computational costs are concerned. However, it comes at the cost of potentially increased storage requirements, which may slow down the algorithm depending on k .

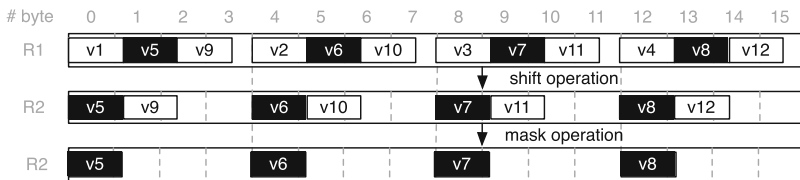


Fig. 3. Example of vectorized decompression for AVBP with bit-case $k = 9$.

3.4 Vectorized Predicate Evaluation

Now with the values decompressed, they can be written back into memory as a 128-bit block using a single SIMD store operation. There, they can be used for lookups in the column specific dictionary in order to materialize the values. However, simply materializing the entirety of all packed values is hardly ever necessary. Instead, a full scan of the column vector will usually only be performed in order to evaluate predicates on it and generate a position vector of values matching the specified condition. As described in [14], predicate evaluations can be performed using SIMD instructions inside the decompression process without writing the decompressed data back into memory.

Assuming four values decompressed and placed into their respective DWs in a 128-bit SIMD register, a single SIMD instruction can be used to compare all four values against a reference value in a single step. This is performed loading the latter as a 32-bit value into a second register, replicated so that each DW contains the reference value. The compare operation then sets the bits of another register accordingly, meaning that if the comparison evaluates to true, all bits in that DW are set to 1, otherwise 0. This is visualized in Fig. 4, register $R2$ still contains the four decompressed values $v5$, $v6$, $v7$ and $v8$ from Fig. 3 and the comparison value $comp$ is loaded into register $R3$, replicated over all DW blocks. The result is then stored in another register $R4$, in this case both $v5$ and $v7$ match the predicate specified through the comparison value $comp$ and the specific compare instruction. Algorithm 2 shows the process integrated into the AVBP decompression process, which however can be included into both HBP and VBP as well.

With the addition of a one-time load of the comparison value $comp$ and the addition of a single instruction **pCompare**, which represents a parallel SIMD comparison operation on four 32-bit values, the decompression algorithm is adjusted to include predicate evaluation. The result vector from $R4$ can now be written back to memory for further operations. The vectorized predicate evaluation process is the same for HBP, VBP and AVBP.

Considering the result vector's format of a set of 32-bit values with all bits set to either 0 or 1, the use cases without modification are limited. In order to generate a position list of matching entries in the vector, the result vector must be parsed again for elements unequal to zero and their position noted. In many cases though, predicate evaluation will consist of multiple conditions to be

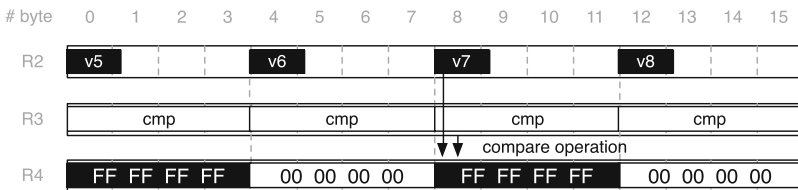


Fig. 4. Example of vectorized predicate evaluation for HBP with bit-case $k = 9$.

Algorithm 2. AVBP Predicate Evaluation

```

i ← 0
R3 ← load128(comp)                                ▷ Comparison Value
while i < n do
  block ← i/(128/k)
  R1 ← load128(input[block])
  offset ← 0
  while offset < 32 do
    R2 ← pRShift(R1, offset)
    R2 ← pAnd(R2, maskk)
    R4 ← pCompare(R2, R3)                        ▷ Predicate Evaluation
    store128(R4, output[i])
    offset ← offset + k
    i ← i + 4
  end while
end while

```

tested, for instance when searching for values between two constants. Lacking a corresponding SIMD instruction, the latter would be implemented using a greater than and a less than comparison operation. Here the result vector of the first operation can be reused in the second without modification. Keeping in mind the all zero or all one structure, a simple AND operation on the respective parts of the existing result vector and the result of the current operation produce an intersection of both result vectors with minimal overhead. This even holds true for operations on different column vectors, provided they use the same result vector format.

The predicate evaluation could also be adjusted to support pipelined operator execution by having the algorithm always process a chunk of values at a time. The resulting partial vector can then already be used within another scan operator, as it will not be written to again. Once a chunk has been processed, the next operator can process the same value range with little overhead, potentially just overwriting the vector through in-place AND-operations. With a small enough chunk size and the result vector still in cache, multiple operators can be executed with little overhead compared to execution of a single operator.

4 Implementation

We implemented prototypical versions of the algorithms for HBP, VBP and AVBP. Analogous to [14], Intel SSE [1] was used. The intrinsics programming method was used for seamless integration of the optimizations into the code.

The decompression steps explained in Sect. 3 for all three concepts map directly onto single SSE instructions. A single instruction loads one 128-bit block of memory into an SSE register. The 4-byte alignment step necessary for HBP is executed using an 8-bit shuffle instruction with an appropriate bit mask to move four consecutive values into separate 32-byte blocks in a second register. In order

to avoid spanning value issues during the shuffle step, the HBV implementation will default to 32-bit for $k > 26$.

In order to shift all four values to the beginnings of their 32-bit blocks, VBP and AVBP use a regular 32-bit shift instruction. The vertical storage layout makes sure, that the offsets into the blocks are always the same for all four values. This is different for the horizontal method. Here, shift offsets can be different depending on k . Shifting a value by n bit however is the same as multiplying it by 2^n . Thus, variable shifting can be realized as a 32-bit multiplication instruction with different factors, essentially performing the desired shift.

The final masking step is again the same for HBP, VBP and AVBP. Using a bit mask with only the first k bits of each 32-bit block set to 1 and a 128-bit AND instruction, all remaining $32 - k$ bits are set to 0 and the decompression is finished and the values can be written back to memory with a 128-bit store instruction.

All bit-packing algorithms feature two means of value access, a single value *get* method and a range access *mget*, which uses the SIMD decompression technique. Both expect the index of a value to be extracted as a parameter. While *get* will only extract the specified value, *mget* is designed for full scans of the vector. Therefore it will always extract a range of 128 values, starting with the first value in the 128-bit block in which the specified value resides. Since HBV and VBV can have spanning values, extracting 128 values at a time ensures that the compressed values will always align at 128-bit boundaries, no matter how many bits per value are used. Stopping before the end of a block would require the same block to be loaded again for the next range of values.

VBP and AVBP also implement a comparison method, which allows for scanning a vector for values equal to a passed in value. The first part of the comparison algorithm requires the values to be decompressed and therefore follows above steps. After the last masking step, before the values are written back to memory, the comparison is performed. The 32-bit value *cmp* to be checked against needs to be loaded once and stored replicated in a separate 128-bit register as $\{cmp, cmp, cmp, cmp\}$. It can then be used for the scan over the entire vector. The Intel SSE2 instruction set provides a set of comparison operations for 32-bit operands, covering basic comparison tests. This set of instructions checks four 32-bit values in a register against their counterpart in another register and sets all bits of the respective 32-bit block in another 128-bit register to either one or zero, indicating whether the comparison returned true or false. With the negligible one-time load of the comparison value, these instructions can now be used to perform predicate evaluation on the unpacked values using only one additional instruction. Now, instead of writing the decompressed values back to memory, the comparison result vector is written back with no additional overhead.

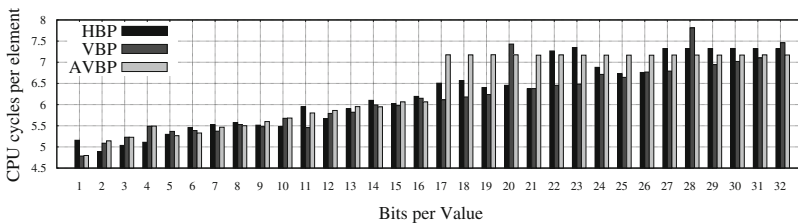
5 Evaluation

The evaluation of the three bit-packing approaches HBP, VBP and AVBP was performed on a server with two Intel Xeon E5450 processors (3.0 GHz) and

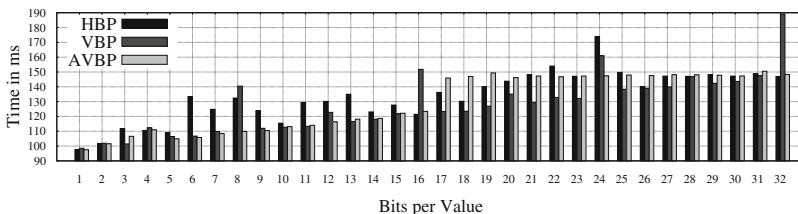
12 MB last level cache each. The system was equipped with 64 GB of DDR2-ECC main memory and running Ubuntu Server version 11.10. All benchmarks were compiled using the GNU gcc version 4.8.1 with the highest level of optimization. Performance measurements were performed using the PAPI library [2].

5.1 Decompression Performance

Figure 5a shows the decompression performance of HBP, VBP and AVBP normalized as CPU cycles per element for all bit-cases from 1 to 32. For this benchmark, an instance of each vector type was created for each bit-case and filled with 200 million values. Therefore, even if only one bit per value is used, the vector will exceed the processor's last level cache, to benchmark memory access effects. In the benchmark, all values are decompressed using the respective *mget* methods of the bit-packing algorithms and stored in a result buffer. The graph shows clearly how decompression speed generally decreases with more bits being used to encode values. While all vectors contained the same number of elements and thus the number of decompression steps necessary did not change, more bits per value increases the overall vector size and requires more load instructions. This has a notable impact on the decompression performance for all three variants. The graph furthermore shows the consistent decompression speeds of AVBP starting at 17 bits per value and HBP starting at 27, both of which default to their 32-bit variant in these cases. Depending on the bit-case, the vectors also greatly diverge in their performance.



(a) Decompressing 200 million values on a system with slow DDR2 memory.



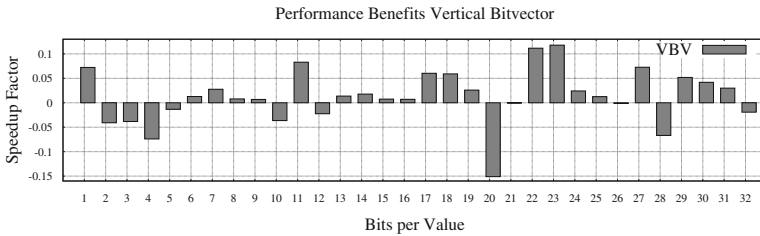
(b) Decompressing 200 million values on a system with faster DDR3 memory.

Fig. 5. Block-wise decompression performance for HBP, VBP and AVBP.

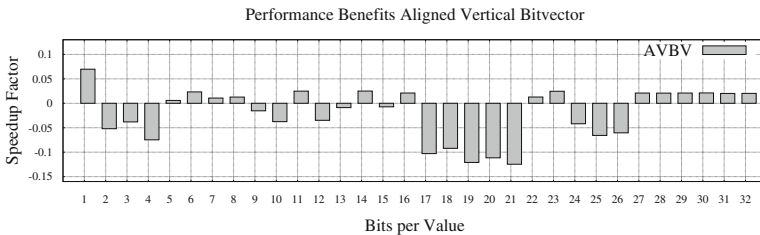
Figure 6a and b quantify the performance gain or loss of VBP and AVBP respectively over HBP. VBP shows great improvements over a wide range of bit-cases, most notably for 1, 11, 22 and 23 bits per value. Interestingly, there is hardly any speedup in bit-cases that are a power of two. AVBP shows similar speedup for 1 bit per value and decreased performance in the range of 2 to 4 bits as compared with VBP. In all other cases, the speedup is comparably low at under 2.5% over the HBP and generally inferior to VBP. This behavior is explained by the increased memory consumption due to the 32-bit alignment. Despite the lower computational overhead as detailed in Sect. 3.3, the increase in storage necessary renders the latter irrelevant.

To demonstrate this aspect more clearly, the same benchmark was compiled and run on a different system with one Intel® Core™ i7-2820QM processor with 8 MB last level cache and running Mac OS X 10.8.4. This system was equipped with 8 GB DDR3 memory running at 1600 MHz, whereas the server's memory used in the previous experiments were older DDR modules running at lower speeds. Despite the CPU's memory controller's lower memory bandwidth, the decreased access latency of the DDR3 memory modules resulted in a lower penalty for the AVBP as can be seen in Fig. 5b. With the effects of memory latency reduced, the differences in the bit-cases among HBP and VBP become more apparent.

Overall it can be noted though, that VBP decompression speed outperforms the horizontal version in most cases. The differences between the vertical versions however make it necessary to take the desired bits per value into account when



(a) Direct comparison for VBP and HBP.



(b) Direct comparison for AVBP and HBP.

Fig. 6. Direct decompression speedup comparisons.

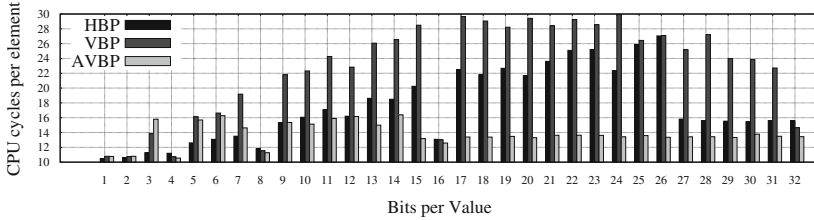


Fig. 7. Performance of single value extraction at 0.1 % Selectivity.

choosing between them. The AVBP generally provides less performance gain, but does in a few cases such as 8, 16 and 32 bits per value beat both HBP and VBP.

5.2 Single Value Access

While all three bit-packing variants are optimized for SIMD supported range access, single value lookups are necessary for tuple reconstruction. The value lookup is implemented differently in all vectors as it needs to comply with the underlying storage model. Figure 7 shows the vectors' performance when selecting 0.1 % random values out of 200 million compressed values.

Power of two bit-cases allow for easy extraction in all cases and show similar performance among the three vector types. While HBP achieves the lowest extraction speeds up until 9 bits per value, AVBP takes over starting at 10 bits per value and for the remaining bit-cases allows much faster single value access of up to a factor of 4 in case of 26 bits per value. Again, the graph shows roughly constant performance of HBP after bit-case 26 and AVBP after bit-case 16. The regular vertical method shows the slowest access speed in essentially all cases. This is due to the nature of the VBP storage layout, due to which a higher number of spanning values occur as values need to be aligned at 32-bit boundaries. While HBP encounters spanning values as well, only the next data block of for example 64 bits must be loaded. VBP requires the corresponding 32-bit block in the next 128-bit block, leading to more stride accesses and higher potential for cache misses. However, single value extraction is hardly ever performed on a larger scale but only to retrieve specific values determined by a scan.

6 Conclusion

In-memory column-stores strive to conserve space as a still scarce resource and rely heavily on the compression of data. In order to maintain access speeds similar to operations on uncompressed data, several techniques such as the SIMD-Scan [14] have been proposed, utilizing available on-chip vector processing units to improve performance.

This paper presents two new vertical storage layouts with according algorithms called Vertical Bit-Packing (VBP) and Aligned Vertical Bit-Packing

(AVBP), targeted at further improving the performance of the SIMD-Scan approach. The experiments carried out demonstrate the performance gain of VBP of up to 12% over the classic Horizontal Bit-Packing (HBP) approach in over half of the analyzed bit-cases, whilst maintaining the same memory footprint.

AVBP only offers minimal performance improvements in some bit-cases and is generally inferior to HBP. It thus nicely demonstrates that despite computational simplification the increase in storage space needed has a major impact on the performance. In some bit-cases however, HBP still offers the best performance. The optimal choice therefore depends on the necessary bits per value as to be determined from the column vector's value range.

References

1. Abel, J., Balasubramanian, K., Barger, M., Craver, T., Phlipot, M.: Applications tuning for streaming SIMD extensions. *Intel Technol. J.* **Q2**, 1–13 (1999)
2. Browne, S., Dongarra, J., Garner, N., Ho, G., Mucci, P.: A portable programming interface for performance evaluation on modern processors. *Int. J. High Perform. Comput. Appl.* **14**, 189–204 (2000)
3. Chhugani, J., Nguyen, A.D., Lee, V.W., Macy, W., Hagog, M., Chen, Y.-K., Baransi, A., Kumar, S., Dubey, P.: Efficient implementation of sorting on multi-core SIMD CPU architecture. In: *VLDB* (2008)
4. Färber, F., Cha, S.K., Primsch, J., Bornhövd, C., Sigg, S., Lehner, W.: SAP HANA database: data management for modern business applications. In: *SIGMOD* (2012)
5. Grund, M., Krüger, J., Plattner, H., Zeier, A., Cudre-Mauroux, P., Madden, S.: HYRISE: a main memory hybrid storage engine. In: *VLDB* (2010)
6. Krüger, J., Kim, C., Grund, M., Satish, N., Schwalb, D., Chhugani, J., Plattner, H., Dubey, P., Zeier, A.: Fast updates on read-optimized databases using multi-core CPUs. In: *VLDB* (2011)
7. Lemire, D., Boytsov, L.: Decoding billions of integers per second through vectorization. In: *CoRR* (2012)
8. Lemke, C., Sattler, K.-U., Faerber, F., Zeier, A.: Speeding up queries in column stores. In: Bach Pedersen, T., Mohania, M.K., Tjoa, A.M. (eds.) *DAWAK 2010*. LNCS, vol. 6263, pp. 117–129. Springer, Heidelberg (2010)
9. Li, Y., Patel, J.M.: BitWeaving: fast scans for main memory data processing. In: *SIGMOD* (2013)
10. Manegold, S., Boncz, P.A., Kersten, M.L.: Optimizing database architecture for the new bottleneck: memory access. In: *VLDB* (2000)
11. Plattner, H., Zeier, A.: *In-Memory Data Management: An Inflection Point for Enterprise Applications*. Springer, New York (2011)
12. Schlegel, B., Gemulla, R., Lehner, W.: Fast integer compression using SIMD instructions. In: *Proceedings of the Sixth International Workshop on Data Management on New Hardware*, pp. 34–40. ACM (2010)
13. Willhalm, T., Oukid, I., Mueller, I., Faerber, F.: Vectorizing database column scans with complex predicates. In: *AMDS* (2013)
14. Willhalm, T., Popovici, N., Boshmaf, Y., Plattner, H., Zeier, A., Schaffner, J.: SIMD-scan: ultra fast in-memory table scan using on-chip vector processing units. *Proc. VLDB Endow.* **2**(1), 385–394 (2009)

Efficient Streaming Detection of Hidden Clusters in Big Data Using Subspace Stream Clustering

Marwan Hassani^(✉) and Thomas Seidl

Data Management and Data Exploration Group, RWTH Aachen University,
Aachen, Germany

{hassani,seidl}@cs.rwth-aachen.de

Abstract. Recently, many data mining techniques were revisited to cope with the new big data challenges. Nearly all of these algorithms considered the efficiency of the mining algorithm to survive the increasing size of the data. However, as the dimensionality of the data increases, not only the efficiency but also the effectiveness of traditional mining algorithms is compromised. For instance, clusters hidden in some subspaces are hard to be detected using traditional clustering algorithms, as the dimensionality of the data increases. In this paper, we consider both the huge size, and the high dimensionality of big data by providing a novel solution that presents a three-phase model for subspace stream clustering algorithms. Our novel model, overcomes the huge size of the big data in its first phase, by continuously applying a streaming concept over the huge data objects, and summarizing them into micro-clusters. Then, after each certain batch of data, or after upon a user request, the second phase is applied over the data summarized in micro-clusters, to reconstruct the current distribution of the data out of the current summaries. In the third phase, a subspace clustering algorithm is applied to overcome the high dimensionality of the data, and to find hidden clusters within some subspace. An extensive evaluation study over different scenarios that follow our model over a big data set is performed.

Keywords: Subspace stream clustering · Streaming big data · Real-time streaming analysis of big data

1 Introduction

The continuously-increasing sizes and dimensionality of available data is a fact that is appearing in a lot of domains nowadays. Actually, it is very hard to find a data producing area where the output does not easily become big, with respect to the size, the dimensionality, or both. Recently, many data mining techniques were revisited to cope with the emerging big data challenges. Almost all of these algorithms considered the efficiency of the mining algorithm to survive the increasing size of the data. However, as the dimensionality of the data increases,

not only the efficiency, but also the effectiveness of traditional mining algorithms is compromised. In this paper, we present an algorithmic model that considers both the big size and the high dimensionality of the data using the subspace stream clustering concept. We shortly define the related concepts in Sects. 1.1 and 1.2, then we present the motivation with a paper structure in Sect. 1.3.

1.1 Subspace Clustering

Clustering is an example of a data mining algorithm that aim at grouping similar objects in the data set in the same group or cluster, and dissimilar objects in different groups or clusters. The definition of *similarity* here is introduced according to some distance function. Thus, far objects are dissimilar, and vice versa. However, for big data, as the dimensionality of the objects increases, the distances between objects in higher subspaces increases exponentially, making it very hard to find “similar” objects, and all objects of the data set appear as outliers. This problem was termed “curse of dimensionality” [5] where traditional distance functions are no more applicable.

In recent applications like network intrusion detection, objects (connections) are described using many dimensions. For example, each connection in the famous Network Intrusion Dataset [1] has 42 dimensions. Recent research introduced subspace clustering which aims at locally detecting relevant dimensions per cluster. If objects in a certain cluster are densely close to each other on some subgroup of dimension (also called *subspace*), then this subgroup of dimensions is relevant to that cluster. Otherwise, if they are scattered over other dimensions then those dimensions are irrelevant to that cluster. Thus, for each cluster, relevant dimensions are locally determined and irrelevant dimensions are ignored.

Figure 1 gives an example of applying a full-space clustering of objects with only two dimensions, for illustrative purposes. *Cluster 1* represents the only full space cluster (here *2-dimensional* cluster) as both dimensions are relevant. All other points are considered outliers, since they are not forming a dense area in the full space. However, as we see in Fig. 2, for the same dataset, although the purple objects are categorized as outliers by full-space clustering in Fig. 1, they form a dense cluster when projecting them over *Dim 2*. Objects of *Cluster 2* are highly scattered in *Dim 1* making *Dim 1* irrelevant for *Cluster 2*. Subspace/Projected clustering algorithms try to find all clusters hidden in any subspace, and not only in the full space, in addition, to the full space clusters.

The same discussion applies to the projections of *Cluster 1* over *Dim 1* and *Dim 2*.

1.2 Stream Mining

Streaming data on the other hand, is available in increasingly many applications. In modern IT-systems, there is a constant stream of data sprouting out of log and maintenance files. The number of attributes in these files still rises, together with the frequency of output generation. The same applies to connection protocols

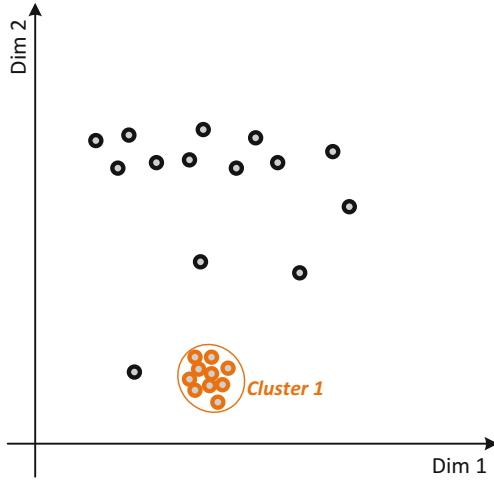


Fig. 1. An example of full-space clustering

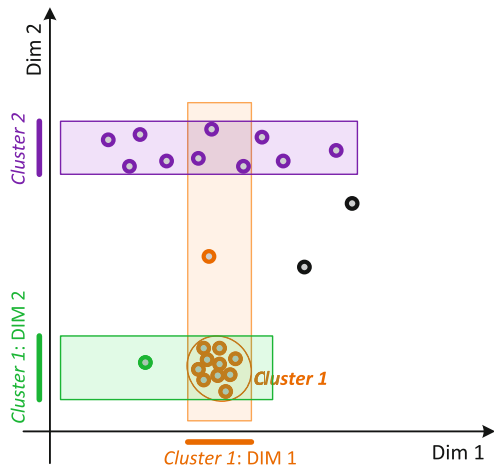


Fig. 2. An example of subspace clustering

of different user groups, be it phone-calls or TCP-connections, as well as monitoring user behavior of a software in a live environment. There are thousands of examples about sensors measuring different sorts of attributes like pollution, micro waves, humidity, heart beats, body temperature, acceleration, speed, location etc. In all of the above examples, there is constant stream of information which has to be processed in a timely manner. Apparently, it is impossible to save every bit of information getting caught in major telescopes, the same goes for large-scale physical experiments where saving all incoming data might not be possible or would delay the next experiment by a large margin.

All these applications have one thing in common: To further get a grasp of the information contained deep in the data, we have to handle a constant flood of data. Sometimes it is not applicable to save the data stream due to memory restrictions or it might be of massive importance to mine information as soon as they occur. In all these cases, we somehow have to work with streaming, volatile data and use procedures to maximize the information gain we can get along while paying attention to occupy just enough space to make it happen.

1.3 Motivation

The streaming model of mining data online was a suitable concept to be adopted when mining big *static* data. The idea of online processing of the data with the expectation of different speeds is very suitable for processing data over batches. Although, such solution might reduce the quality of the mining algorithm due to its possibility of real time processing and management of the data, the ability to handle huge data sizes batch-wise made them appealing for the big data community.

To handle the challenge of increasing dimensionality in big data, we will consider also the usage of subspace clustering to find hidden clusters of data within some subspace. Available subspace clustering algorithms (cf. Sect. 2) are known to be heavy-weighted when talking about scanning the whole data. This feature contradicts the stream mining requirements of having a light-weighted version of any algorithm, that is capable of processing the data online.

In this paper, we will consider both the huge size, and the high dimensionality of the big data by providing a novel solution that presents a model for subspace stream clustering algorithms. Our novel model, overcomes the huge size of the big data by continuously applying a streaming concept over the huge data, and summarizing them into micro-clusters. Then, after each certain batch of data, or after a user request, the data is reconstructed out of the micro-clusters summaries and forwarded to one subspace clustering algorithm, to overcome the high dimensionality of the data, and to be able to find hidden clusters within some subspace.

The remainder of this paper is organized as follows: Sect. 2 lists some of the related work in the areas of stream clustering and static subspace clustering. Section 3 introduces our model and brings a running example to explain it step-by-step. Section 4 presents a thorough evaluation that compares the running time and the clustering quality of different scenarios of our given model and compares them to one static model. Then we conclude this paper in Sect. 5 by giving also an outlook.

2 Related Work

The first grid based subspace clustering algorithm introduced was CLIQUE [4]. It works with a grid-based approach to identify subspace clusters in a bottom-up fashion using so called dense grid cells which contain more points than a

certain threshold. This is done with using the apriori method. To speed up the computation, monotonicity laws are used to prune the possible dense-regions. This idea is picked up by MAFIA [10] but instead of using a static grid, the grid is now adaptive and pruning is dropped in favor of heavy parallel optimization. Projected clustering uses a special distance function in conjunction with a common clustering algorithm. PROCLUS [3] is a three-phase algorithm, and this is the one we use in this paper as a running example. First we guess medoid candidates from objects spread over the data set. Then we improve these medoids and compute dimensions for each. In the last phase, we use an algorithm similar to k-medoid to refine the clusters. *P3C* [15] starts with one-dimensional intervals which might be approximate higher dimensional subspace clusters. Merging these in an a-priori bottom-up method nets high-dimensional subspace clusters. These maximal-dimensional clusters are used as cluster-cores for a refinement step using expectation-maximization-like algorithm. SubClu [14] uses the DBSCAN model [9] of connected sets in an apriori style. By applying DBSCAN over each subspace, SubClu requires a high runtime.

Our algorithmic model presented in this paper differs from the above algorithms by applying the subspace clustering algorithm over streaming data. Due to their huge complexity, a straight-forward utilization of the above algorithms over big datasets as in the streaming cases faces serious memory issues (cf. Sect. 4). Our algorithmic model gives a smooth way of applying subspace clustering algorithms in the streaming scenario.

There is a rich body of literature on stream clustering. Approaches can be categorized from different perspectives, e.g. whether convex or arbitrary shapes are found, whether data is processed in chunks or one at a time, or whether it is a single algorithm or it uses an online component to maintain data summaries and an offline component for the final clustering (as most of stream clustering algorithms). Convex stream clustering approaches are based on a k -center clustering [2, 13]. Detecting clusters of arbitrary shapes in streaming data has been proposed using density based clustering [7, 8]. Another line of research considers the anytime clustering with the existence of outliers [12].

The model we present in this paper follows a three-phase model that differs from the two-phase one which appears in most of the above stream clustering algorithms. This is done to be able to deal with big data requirements as well as the subspace clustering algorithms requirements.

3 Algorithmic Model

In this section we will present the model we are using for applying subspace stream clustering algorithms to efficiently and effectively find subspace clusters in big data. We will use one scenario of the combination: CluStream+SUBCLU as a running example, while the same will apply for other scenarios. In the experimental part, we will show a comparison between most scenarios in details.

The idea of stream subspace clustering is intuitive: We cluster the incoming data “live” and save features of the clustering for a defined period of time. We can

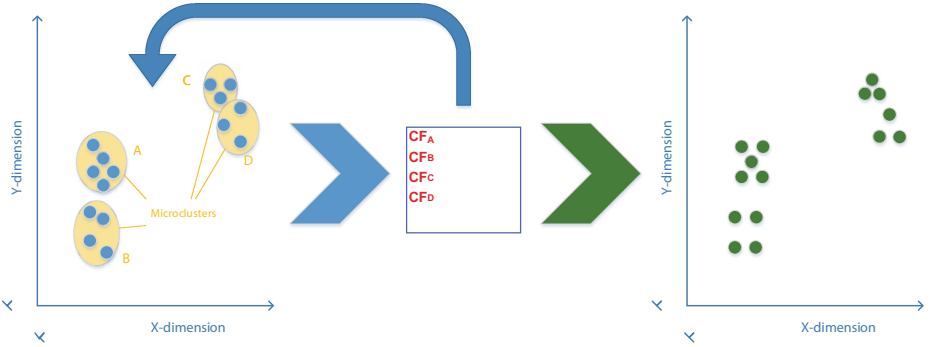


Fig. 3. The whole process of our model for stream subspace clustering for big data. The blue arrows represent the online phase while the green arrow represents the reconstruction and the offline phases (Color figure online).

then approximately reconstruct the data for a given time frame, the more recent the more accurate, and use a classic subspace algorithm to determine a clustering of this frame.

Going a bit more into detail, we use a three-stop approach in this paper: an online phase, a regeneration phase and then an offline phase. The data stream gets processed by CluStream [2] (or DenStream [7]) in the online phase, which produces micro clusters for the current input data (cf. Sect. 3.1). These clusters are then saved as cluster feature vectors as seen in Fig. 3. Then, upon some request from the user for a final clustering or after a certain amount of time, we regenerate the points out of the summaries in the regeneration phase (cf. Sect. 3.2). The regenerated data is then forwarded to the final subspace clustering algorithm which produces the final clusters (cf. Sect. 3.3).

3.1 Online Phase: A Stream Clustering Algorithm

In the online phase of our model, a summarization of the data stream points is performed and the resulting microclusters is given by sets of cluster features:

$$CF_A = (N, LS_i, SS_i)$$

which represent the number of points within that microcluster A , their linear sum and their squared sum, respectively. One of the two online algorithms (CluStream or DenStream) is responsible for forming these microclusters, deleting older ones or continuously maintaining the updated ones.

3.2 Regeneration Phase: Gaussian Out of Online Summaries

After reaching a predefined time threshold, we call it here (window size), we compute Gaussian distributed objects out of the statistics we got from the cluster features of the microclusters (green arrow Fig. 3).

This step is called the offline phase, where the clustering features are used to reconstruct an approximation to the original N points, for each microcluster, using Gaussian functions to reconstruct points over each dimension i .

$$c_i = \frac{LS_i}{N}$$

with a radius:

$$r = \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2}$$

where:

$$SS = \frac{1}{d} \sum_{i=1}^d SS_i$$

and

$$LS = \frac{1}{d} \sum_{i=1}^d LS_i$$

The generated N_A points for each microcluster will be now normally distributed. Thus, they will look a little bit differently distributed than the original distribution (compare the points in Fig. 3 Right with Left). Actually, this is the only approximation that we have in our model.

3.3 Offline Phase: A Subspace Clustering Algorithm

The generated N points are forwarded to one of the four subspace clustering algorithms. These are SubClu [14], ProClus [3], Clique [4] and P3C [15]. This results with 8 different combinations of algorithms that can be tested. These algorithms are applied to the streaming cases. Other than static data that do not vary over time, stream data are given in different rate and pattern changing dynamically, which makes it challenging to analyze its evolving structure and behavior. In streaming scenarios, we also often face limitations on processing time and storage, since a vast amount of continuous data are coming rapidly.

The offline part of our model uses a subspace clustering algorithm to deliver the final clusters. In our running example we use the subspace clustering algorithm: SUBCLU [14] over the regenerated points. The aim of SUBCLU is to find all clusters in all subspaces of the data in a greedy, bottom-up way. The rough work-flow of the algorithm is that we start in the 1-dimensional subspaces and find clusters using DBSCAN. Then we recursively compute a set of candidate subspaces with the dimensionality $k + 1$, prune these and test each candidate if there are still clusters left in the subspace. As there are a lot of range queries necessary for DBSCAN which make up the bulk of computation time needed in SUBCLU, the implementation introduced in [14] uses an efficient index support for range queries of single attributes in logarithmic time. In case of range queries on more than one attribute, the range query for each attribute is used on its own and the intersection of all intermediate results is computed as final result.

4 Experiments

4.1 Example

We test this combination of CluStream and SUBCLU on a very large dataset containing connection information. Each of these objects contains 41 different attributes from which we will try to cluster information. 4.8 million objects in total are included in this data set, To compare SUBCLU and other offline-algorithms, we will test a variety of other macro algorithms in regard of accuracy and performance. This will be done using the Subspace MOA framework [11], as it gives the possibility to work with a GUI and all common offline-algorithms, such as CLIQUE, PROCLU and P3C. All calculations were done on a AMD FX 8-core clocked at 4 GHz with 8 GB RAM.

4.2 Dataset

The real data set used in this experiment is the KDD CUP'99 Network Intrusion Detection data set [1] which has been used to evaluate several stream clustering algorithms [2] and [7]. The MIT Lincoln Labs recorded the traffic of a LAN network for two weeks. Each connection is labeled as either normal, or as an attack. There are 22 different types of attacks, which fall into the following four main categories:

1. DOS: denial-of-service
2. R2L: unauthorized access to a remote machine
3. U2R: unauthorized access to local root privileges
4. Probing: surveillance and other probing.

The data set consists of 494021 TCP connections where most of them are normal connections. Each connection has 42 attributes. These attributes can be discrete or continuous. As in [2] and [7], all the 34 continuous attributes for the clustering task are used. Table 1 gives a detailed view of the different attacks appearing within a horizon $H = 5$ and stream speed = 1000 of this dataset.

4.3 Framework

The software used as part of this paper is the java based Subspace MOA. Introduced in [11], this software uses the interface of the MOA framework [6] style and contains an additional tab which is solely for subspace clustering. As data input Subspace MOA supports a synthetic random RBF (subspace) generator or the option to read an external ARFF file as input stream. As for one-stop algorithms, seven total different algorithms can be chosen, for the three-phase method, three different online- as well as five common offline algorithms are available.

Table 1. Table of stream of Network Intrusion Data set within the horizon $H = 5$, stream speed = 1000

Normal or attack type	Objects within horizon $H = 5$ at time unit			
	150	350	373	400
Normal	4004	4097	892	406
Satan	380	0	0	0
Buffer overflow	7	1	2	0
Teardrop	99	99	383	0
Smurf	143	0	819	2988
Ipsweep	52	182	0	0
Loadmodule	6	0	0	1
Rootkit	1	0	0	1
Warezclient	307	0	0	0
Multihop	0	0	0	0
Neptune	0	618	2688	1603
Pod	0	1	99	0
Portsweep	0	1	117	1
Land	0	1	0	0
Sum	5000	5000	5000	5000

CluStream is based on snapshots, saved as extended cluster feature vectors which contain information about characteristics of the discovered micro clusters. Contained information be contained are the sum of data values and time stamps amongst others. The total amount of micro clusters is always maintained, as such new objects are inserted in the closest already existing micro cluster (MC) or a new micro cluster containing this object is created. As the amount of micro clusters has to be constant, a not-relevant micro cluster is then deleted or the two closest cluster are merged. Using a pyramidal time frame to store snapshots allows us to halt the computation at any given point and produce the input for SUBCLU.

For this part, we have to specify the time-horizon h . For this horizon, CluStream produces points for each micro cluster which are computed according to the variance and mean of the saved micro cluster information using a Gaussian distribution. This set of points is then passed to SUBCLU.

SUBCLU then produces the actual clustering using the input provided by CluStream. Since only the snapshot of the recent data is passed over, we will most likely not find all possible subset clusters in any given snapshot.

In the following, we will take a closer look what the used implementation of CluStream in conjunction with SUBCLU in the SubspaceMOA framework can accomplish. In the first section we will compare SUBCLU against CLIQUE, PROCLUS and P3C and get a first glimpse of accuracy between those. In the following section, we will compare different settings of SUBCLU in regards to three measurement methods.

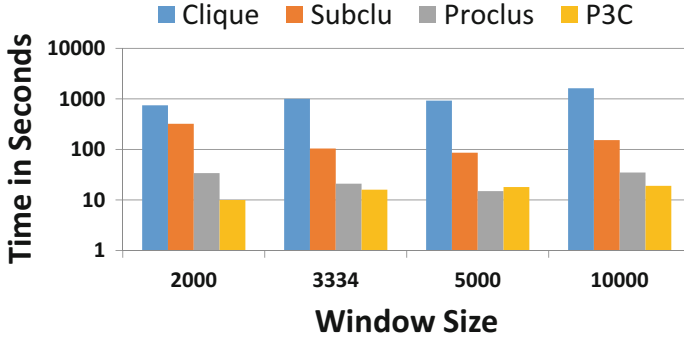


Fig. 4. The performance of the four offline algorithms after using CluStream in the online phase when changing the window size (the batch size).

4.4 Running Time Results

To evaluate the performance we will have a set micro algorithm, CluStream with a maximum of 30 micro clusters, and compare four different window sizes of 2000–10000 for four different macro clustering algorithms while keeping the number of overall processed objects at a steady 10000. CLIQUE with default settings in SubspaceMOA which are $\xi = 10$ and $\tau = 0.01$, SUBCLU with $\epsilon = 0.002$ and $m = 5$, PROCLUS with $c = 5, d = 2$ and P3C with $p = 10, \chi^2 = 0.001$ which also are default settings.

Note that in Fig. 4 the scale of the runtime in s is logarithmic. CLIQUE is too slow to have all values in one figure if using metric scale.

SUBCLU's predecessor CLIQUE is in every aspect the slowest algorithm going over this benchmark. At even the best case, it is slower than any other algorithm in this set. It also needs even more time to process all 10000 objects when using a larger window size peaking at a stunning 27 min needed for one run when using the whole 10000 objects as one window.

SUBCLU is vastly more efficient. Between 80 s and just over 5 min are the values for the different window sizes. Interestingly, a window size of 2000, and thus doing 5 computations of each, yields the worst results while two computation circles of 5000 objects each are still good runtime-wise.

PROCLUS and P3C are both extremely fast when compared to SUBCLU, peaking at 35 s tops for a worst-case window. PROCLUS shows the same preference for a 5000 object window as SUBCLU does, P3C prefers smaller windows.

To observe the huge improvement our model brings when compared to the static subspace clustering algorithms, we tried to apply the KDD CUP '99 dataset over the static PROCLUS algorithm. We have tried first to run the PROCLUS over the whole dataset size with 1 G memory allocated for the algorithms' heap. As it was crashing, we decided to try smaller versions of the dataset, by getting the first ones as they appear in the dataset. As shown in Fig. 5, the exponential increase of the runtime is obvious as the size of the dataset increases. The algorithms started to crash when trying a sub-dataset of size 200 K. Additionally,

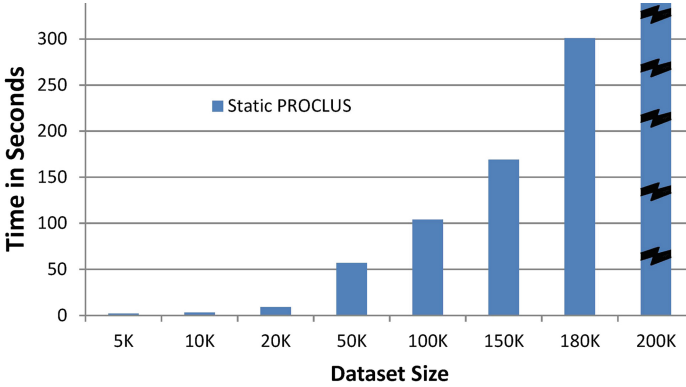


Fig. 5. The runtime performance of a *static* subspace clustering algorithm: PROCLUS. Beginning from a sub-dataset size of 200 K objects, the algorithm failed to successfully finish the running.

the runtime improvement that our algorithmic model causes over PROCLUS is obvious when trying any window size (cf. Fig. 4).

4.5 Accuracy Results

A total of three different evaluation measures are used in this section: The $F1$ measure, which gives an overview how well hidden clusters are represented by the output of the algorithm, RNIA [16], which measures how well hidden subobjects are covered by already found objects, and CE [16], which works similar to RNIA but also evaluates if a cluster is split up into several smaller clusters. From now on, when talking about RNIA or CE measures, we mean $1 - CE$ and $1 - RNIA$, so the nearer the measure to 1, the better, just as it is the case for the $F1$ measure. For the previous settings of the performance evaluation, we averaged for each algorithm the three accuracy evaluation measures. Figure 6 depicts the gained results as the batch size (window size) changes. As expected, the accuracy of almost all algorithms increases when the window size increases, with one exception with SubClu which is fluctuating a bit. The reason of this improvement of the accuracy is the fact that considering more data at a time, gives each algorithm more possibility to find even more hidden clusters. Another observation, is that nearly all of the algorithms who performed well w.r.t. the running time, are also accurate. While the slow ones are also delivering bad results. This makes P3C a winner algorithm when considering the running time and accuracy.

Going over the different settings of CluStream and SUBCLU, we first check for different ϵ to get a feel of how this parameter affects the performance. Just as in the performance section, we use 30 as maximum number of micro clusters for CluStream, 2000 as window size and $m = 5$ for SUBCLU.

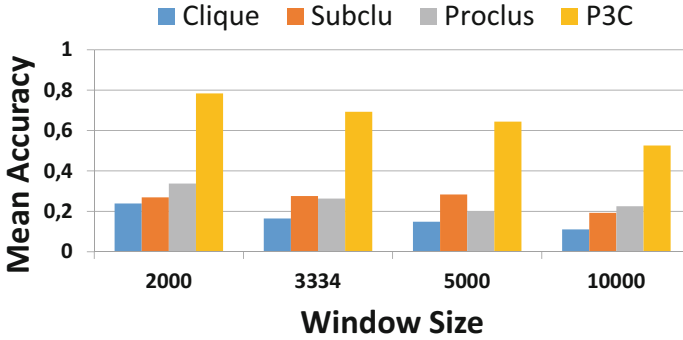


Fig. 6. The averaged accuracy of the four offline algorithms when using CluStream in the online phase when changing the window size (the batch size).

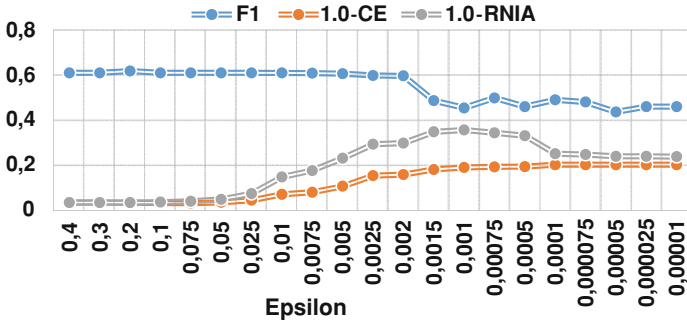


Fig. 7. Different accuracy measures of the SUBCLU algorithm after using CluStream in the online phase when changing ϵ (the neighborhood parameter).

Presented in Fig. 7, the $F1$ measures start to drop after $\epsilon = 0.002$. Prior to this value, smaller settings meant that there are more but smaller clusters holding the balance between precision and recall. After this, a bit of precision is lost, resulting in the 0.1 worse measure. Starting from a smaller $\epsilon = 0.0015$, both RNIA and CE have a maximum with RNIA falling a bit after $\epsilon = 0.0001$. It seems like the algorithm did not find too many objects after this part, however the found ones are clustered with little excessive clusters. Another parameter to check is the minimum points m , found in Fig. 8. For this benchmark, $\epsilon = 0.001$ was used. $m = 9$ seems to be an interesting point, resulting in a spike from both $F1$ and RNIA in opposite directions. We could assume this was a threshold for adding “bad” objects to a cluster without enabling DBSCAN to connect the cluster to an existing “good” one.

Overall, this setting seems to have lower impact on the accuracy of SUBCLU than the ϵ parameter. As these settings are mainly used for DBSCAN this is not surprising but still nice to confirm.

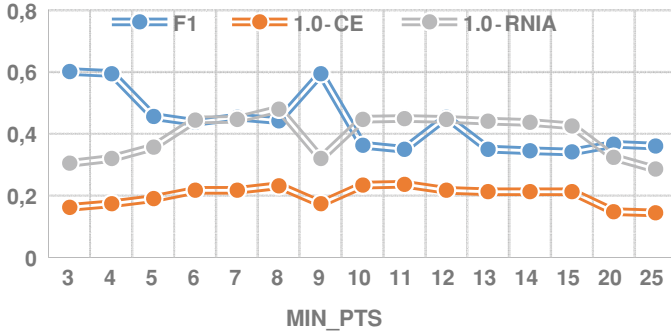


Fig. 8. Different accuracy measures of the SUBCLU algorithm after using CluStream in the online phase when changing m (the minimum number of points needed in the ϵ -neighborhood for an object to become core [14]).

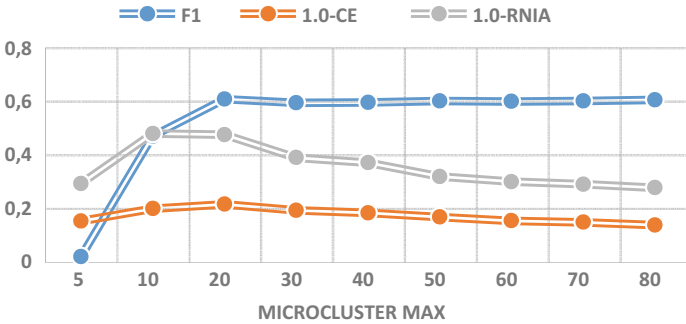


Fig. 9. Different accuracy measures of the SUBCLU algorithm after using CluStream in the online phase when changing the maximum allowed number of microclusters within CluStream.

In this last part of this accuracy evaluation we will take a short look to see how varying different the maximum amount of micro clusters within CluStream affects the quality of the results. Figure 9 depicts the results, as expected, a certain minimum amount of micro clusters has to be present to achieve good results. After this threshold, which seems to be around 20, the results do not change too much but steadily go worse. This holds true especially for the RNIA measure. Probably 23 is a good number because there are a total of 23 different connection types present in the data set.

5 Conclusion and Outlook

In this paper, we considered both the huge size, and the high dimensionality of big data by providing a novel solution that presents a three-phase model for subspace stream clustering algorithms. Our novel model, proved to overcome the

huge size of the big data in its first phase, by continuously applying a streaming concept over the huge data objects, and summarizing them into micro-clusters. Then, after each certain batch of data, the second phase is applied over the data summarized in micro-clusters, to reconstruct them by assuming a Gaussian distribution within the relatively small sized micro-clusters. Using a subspace clustering algorithm in the third phase, to overcome the high dimensionality of the data, we make the heavy subspace clustering feasible in finding hidden clusters within some subspace of big datasets.

In more details, we took a look at the density-based subspace clustering algorithm SUBCLU as a running example. We used important observations about monotonicity to develop the idea of a bottom-up greedy algorithm which could detect arbitrary shaped clusters hidden in subspaces. We then used this algorithm to process a high-dimensional data stream of connection information, using CluStream as micro algorithm in conjunction with SUBCLU as macro algorithm. Comparing SUBCLU to CLIQUE, PROCLUS and P3C, we observed the runtime requirements depending on different window sizes of the micro clustering. After that, we observed how different parameter settings played a large role in producing good results and finished in evaluating the current implementation in Subspace MOA. We have also shown that a direct application of huge datasets over static subspace clustering algorithms will face serious memory issues. Currently, an algorithm such as *P3C* is fast and accurate. The only problem is inherited from the density-based subspace clustering: the dependency on input parameters done by the user. This lead to a trial and error-style preparation before achieving good results.

One good future direction could be discussing two main ideas. First: how to minimize the effect of the input parameters by optimizing the selection of the density parameters such that they fit the distribution in the current batch of the data. Second: how to minimize the effect of the ordering of the data, such that we can read the big data in any streaming order without a dramatic change of the overall accuracy or performance.

Acknowledgments. This work has been supported by the UMIC Research Centre, RWTH Aachen University, Germany.

References

1. KDD Cup 1999 Dataset. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>. Accessed 22 Nov 2013
2. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: VLDB '03, pp. 81–92 (2013)
3. Aggarwal, C.C., Wolf, J.L., Yu, P.S., Procopiuc, C., Park, J.S.: Fast algorithms for projected clustering. In: ACM SIGMOD Record, vol. 28, pp. 61–72 (1999)
4. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: SIGMOD '98, vol. 27, pp. 94–105 (1998)

5. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is nearest neighbor meaningful? In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1998)
6. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. *J. Mach. Learn. Res.* **99**, 1601–1604 (2010)
7. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: *SDM' 06*, pp. 328–339 (2006)
8. Chen, Y., Tu, L.: Density-based clustering for real-time stream data. In: *KDD '07*, pp. 133–142 (2007)
9. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: *KDD '96*, vol. 96, pp. 226–231 (1996)
10. Goil, S., Nagesh, H., Choudhary, A.: MAFIA: efficient and scalable subspace clustering for very large data sets. In: *KDD '99*, pp. 443–452 (1999)
11. Hassani, M., Kim, Y., Seidl, T.: Subspace MOA: subspace stream clustering evaluation using the MOA framework. In: *DASFAA' 13*, pp. 446–449 (2013)
12. Hassani, M., Kranen, P., Seidl, T.: Precise anytime clustering of noisy sensor data with logarithmic complexity. In: *SensorKDD '11 Workshop in conj. with KDD '11*, pp. 52–60 (2011)
13. Hassani, M., Müller, E., Seidl, T.: EDISKCO: energy efficient distributed in-sensor-network k-center clustering with outliers. In: *SensorKDD '09 Workshop in conj. with KDD '09*, pp. 39–48 (2009)
14. Kailing, K., Kriegel, H.-P., Kröger, P.: Density-connected subspace clustering for high-dimensional data. In: *SDM'04*, pp. 246–257 (2004)
15. Moise, G., Sander, J., Ester, M.: P3c: a robust projected clustering algorithm. In: *ICDM '06*, pp. 414–425 (2006)
16. Patrikainen, A., Meila, M.: Comparing subspace clusterings. *TKDE* **18**(7), 902–916 (2006)

A Comparison of Systems to Large-Scale Data Access

Amin Mesmoudi^(✉) and Mohand-Saïd Hacid

CNRS, Université de Lyon, Université Lyon 1, LIRIS, UMR5205,
69622 Lyon, France
{amin.mesmoudi,mshacid}@liris.cnrs.fr

Abstract. With the amount of data produced in several application domains, it is increasingly difficult to manage and query related large data repositories (<https://www.lsstcorp.org/science/wiki/images/DC-Handbook.v1.1.pdf>). Within the PetaSky project, we focus on the problem of managing scientific data in the field of cosmology. The data we consider are those of the LSST project. The overall expected size of the database that will be produced will exceed 60 PB. This paper presents preliminary results of experiments conducted on PT1.1 (<http://lsst1.ncsa.uiuc.edu/schema/index.php?sVer=PT1.1> (with a size of 90 GB.)) and PT1.2 (<http://lsst1.ncsa.uiuc.edu/schema/index.php?sVer=PT1.1> (with a size of 145 GB.)) data sets in order to compare the performances of both centralized and distributed database management systems. As for centralized systems, we have deployed three different DBMSs: *MySQL*, *Postgresql* and *DBMS-X* (a commercial relational database). Regarding distributed systems, we have deployed *HadoopDB* and *Hive*. The goal of these experiments is to report on the ability of these systems to support large scale declarative queries. We mainly investigate the impact of data partitioning, indexing and compression on query execution performances.

Keywords: DBMS · Benchmark · Distributed systems · Centralized systems

1 Introduction and Context

Today, because the amount of data produced in several application domains has been exploding, DBMS vendors¹ are working towards design of advanced tools that could scale when it comes to manage and query very large data repositories. Some domains, like social networks, astronomy and Web ask for new approaches. For instance, Google provided the basis of Map/Reduce framework [3] which was designed basically to efficiently manage Web data. New technologies are influenced, not only by new applications, but also by the intensive use

This work is partially supported by Centre National de la Recherche Scientifique-CNRS. Under the project Petasky-Mastodons (<http://com.isima.fr/Petasky>).

¹ <http://www.adeptia.com/products/Gartner-Cool-Vendors-in-Integration-2010.pdf>

of some kinds of material architectures. One of these architectures is the shared nothing² clusters which is used to deploy many applications that need intensive storage and computation resources.

In many scientific fields, such as physics, astronomy, biology or environmental science, the rapid evolution of data acquisition tools (e.g., sensors, satellites, cameras, telescopes) as well as the extensive use of computer simulations have led in recent years to an important production of data. Modern scientific applications are then facing with new problems that are primarily related to the storage and use of these data. In addition to the growing volume of data to handle, their complex nature (e.g., images, uncertain data, multi scale, ...), the heterogeneity of their formats and the various processing to which they are subject are the main sources of difficulties. The problems are such that scientific data management is now recognized as a real bottleneck^{3,4} which at some extent slows down scientific research since it relies more and more on the analysis of massive data. In this context, the role of the computer as a direct way to improve the discovery process in science^{5,6,7} is important. This has led scientists from different disciplines to work together towards the design and testing of new approaches, tools and techniques for managing and analyzing large data repositories of the order of petabytes.⁸ The work presented in this paper focuses on the problem of managing scientific data in the field of cosmology.

In this paper, we report on these experiments. We mainly analyzed issues related to:

1. Performance: The time required to get all the answers to a query.
2. Speed up [4]: if we have more material dedicated to solve the same task, less time of processing is needed. We analyze speed up according to several configurations. For example, with a machine equipped with a main hard disk of 1 TB of storage space and 113 MB/s of transfer rate, the scan of 30 GB takes 5 min. Replicating data on 2 disks allows to accomplish the scan in less than 5 min.
3. Fault tolerance: The use of more equipments triggers more failures. Indeed, it often happens that a machine does not respond (software problems) or a disk crash (material problems). A task that requires a long processing time may never be finished if whenever a fault occurs, it is necessary to reset the task [5].
4. Latency: It stands for the time required to get the first answer to a query. A guaranty for a fault tolerance impacts latency. This is due, for example, to the requirement that all subtasks should be successfully completed [1].

² <http://db.cs.berkeley.edu/papers/hpts85-nothing.pdf>

³ <http://www.eecs.berkeley.edu/~culler/courses/cs252-s05/papers/DeepData.pdf>

⁴ http://www.cse.buffalo.edu/faculty/tkosar/papers/jnrl_philtrans_2011.pdf

⁵ <http://research.microsoft.com/en-us/um/cambridge/projects/towards2020science/>

⁶ http://www.nitrd.gov/pubs/200311_grand_challenges.pdf

⁷ <http://www.cs.purdue.edu/homes/ake/pub/CommunityCyberInfrastructureEnabledDiscovery.pdf>

⁸ XLDB (Extremely Large Data Bases, <http://www.xldb.org>) and SciDB (Scientific Data Bases, <http://www.scidb.org/>).

The rest of this paper is organized as follows: In Sect. 2 we provide some details regarding material and resources that are used in our experiments. Experimental results and their interpretation are presented in Sect. 3. We discuss our results in Sect. 4. We conclude in Sect. 5.

2 Experimental Environment

Our experiments have targeted two types of architectures:

- A centralized architecture, with 14 GB of RAM, 4 cores and 3 disks with a total storage capacity of 2.5 TB.
- A distributed architecture: We used three clusters composed of 3, 6 and 12 machines, respectively. Each machine has 4 GB of RAM, 660 GB of storage capacity and 2 cores. Network rate between machines can reach 1 GB/s.

According to `hdparm`⁹, the hard disks deliver 113 MB/s for buffered reads.

As for the centralized architecture, we deployed three different DBMS: `MySQL`, `Postgresql` and `DBMS-X` (a commercial relational database). The goal is to report on the ability of these systems to support declarative queries in the one hand and to support LSST requirements¹⁰ in the other hand.

Regarding the distributed architecture, we deployed `Hive`¹¹[6] and `HadoopDB` [2].

2.1 Hadoop

For our experiments, we used two versions of `Hadoop`: 0.19.1 and 1.1.1. Due to the incompatibility of `HadoopDB` with recent versions of `Hadoop`, `HadoopDB` is coupled with `Hadoop` 0.19.1, whereas `Hive` is coupled with `Hadoop` 1.1.1. The both versions run on Java 1.7. We deployed the system using the configuration mentioned in `HadoopDB` original paper.¹² Data in `HDFS` (`Hadoop Distributed File System`) is stored using 256 MB data blocks instead of the default 64 MB. Each `Map/Reduce` job executor runs with a maximum heap size of 1024 MB. Two `Map` instances and a single `Reduce` instance are allowed to be executed concurrently on each node. Buffer space for file read/write operations and the sort buffer are set to 132 MB and 200 MB respectively with 100 concurrent streams for merging.

2.2 Hive

`Hive` is based on `Hadoop`¹³ open source framework. It proposes `HiveQL`, a SQL-like language, to specify analysis tasks. Data is stored using `HDFS`. From a SQL-like query, a set of `Map/Reduce` tasks are generated. `Hive` also schedules the execution of generated tasks.

⁹ A tool that gives the average disc speeds (<http://en.wikipedia.org/wiki/Hdparm>).

¹⁰ <http://www.lsst.org/files/docs/SRD.pdf>

¹¹ <http://hive.apache.org/>

¹² The same configuration is used in [5], the popular `Hadoop` benchmark paper by Pavlou et al.

¹³ <http://hadoop.apache.org/>

2.3 HadoopDB

HadoopDB is based on Hive and uses HiveQL. Existing DBMS are used to store data in cluster nodes. The execution plan, generated by Hive, is processed to push more complex tasks (not only data scan) to nodes in the Map phase. HadoopDB uses an XML file¹⁴ to store access information to data. Essentially, information about chunks and sub-chunks, related to each table, is stored in this file. HadoopDB processes queries as follows. First, HadoopDB parses the query by resorting to Hive's mechanisms. Second, the catalog is parsed and the query is reformulated with respect to chunks and sub chunks information in the catalog. With respect to queries a set of Map/Reduce jobs are generated with a predefined execution order. After the execution of the job, Map and Reduce phases, HadoopDB checks if there is another job to be executed. If any, partial results are stored in HDFS. Otherwise, responses are transferred to the node with an Hive client. In the Map phase of HadoopDB job, a query is sent to the DBMS and responses are formatted in a <key, value> format. In our work, HadoopDB is coupled with PostgresQL in each node of the cluster. HadoopDB also offers a partitioning tool. This tool is used to minimize intermediate results of jobs generated by Hive. The use of existing DBMS for data storage and access layer allows to exploit exiting technologies such as indexing and compression.

2.4 Data Sets

Our experiments use PT1.1¹⁵ and PT1.2¹⁶ data sets. The data sets are organized as follows:

1. PT1.1: This data set contains two tables stored as "CSV" files. The Source table that has 92 attributes¹⁷ and contains 165 million tuples (for a total size of 85 GB). The Object table that has 227 attributes¹⁸ and contains 5 million tuples (for a total size of 5 GB).
2. PT1.2: This data set contains 22 tables stored as "CSV" files with a total size of 220 GB. For this stage of experiments, we make use of only 2 tables: The Source table that has 107 attributes and contains 180 million tuples (for a total size of 139 GB). The Object table that has 229 attributes and contains 5 million tuples (for a total size of 7 GB).

Data can be compressed during the final storage within the DBMS. For example, with DBMS-X and the PT1.1 data set, only 30 GB are used to store the data of the Source table. Table 1 summarizes the features of PT1.1 and PT1.2 data sets (first part of the table) and the features of the expected¹⁹ final Source

¹⁴ Called Catalog.

¹⁵ <http://lsst1.ncsa.uiuc.edu/schema/index.php?sVer=PT1.1>

¹⁶ <http://lsst1.ncsa.uiuc.edu/schema/index.php?sVer=PT1.2>

¹⁷ Note that the expected final Source table will have 125 attributes.

¹⁸ Note that the expected final Object table will have 470 attributes.

¹⁹ <http://www.icis.anl.gov/programs/file.php?id=303&obj=MultiFile&field=filename&attachment=yes>

Table 1. Data set description

Table	#attributes	#records	Size	#indexes	Expected size	Expected #attributes	Expected #records
PT1.1 Object	227	5 million	5 GB	2	109 TB	470	38 Billion
PT1.1 Source	92	165 million	85 GB	7	3.6 PB	125	5 Trillion
PT1.2 Object	229	5 million	7 GB	2	109 TB	470	38 Billion
PT1.2 Source	107	180 million	139 GB	7	3.6 PB	125	5 Trillion

Table 2. Selected queries

Q	Object	Source	Selection	Projection	Join	Group by	Order by
#1		yes	1 (indexed)	92			
#2		yes	2 (indexed)	2			
#3		yes	1 (indexed)	2			
#4		yes	1 (indexed)	3			
#5	yes		2 (range)	227			
#6		yes	1 (range)	1			
#7	yes		4 (range)	227			
#8		yes		1, count		yes	
#9	yes	yes	1 (indexed)	319	yes		
#10		yes		2			yes

and Object tables (the second part of the table). Some indexes are suggested for LSST²⁰ data sets. The source table has 7 indexes and the object table has 2 indexes.

2.5 Queries

Our query sample derives from LSST catalog of queries²¹ with a slight adaptation to comply with the schema²² of PT1.1 and PT1.2 data sets. Also, in the experimental phase, we left out the queries with *Scisql functions*.²³ Table 2 summarizes our set of (ten) queries. The complete specification of queries can be found in the project Website.²⁴

²⁰ <https://dev.lsstcorp.org/trac>

²¹ <https://dev.lsstcorp.org/trac/wiki/db/queries>

²² <http://lsst1.ncsa.uiuc.edu/schema/index.php>

²³ These functions need to be implemented. Such queries will be considered in another test campaign. We already verified that all the functions can be implemented within Hive and HadoopDB.

²⁴ http://com.isima.fr/Petasky/groups/sous-groupe1/queries-1/at_download/file

“yes” in the column *Object* (resp. *Source*) means that the query targets the *Object* (resp. *Source*) table. The column *selection* indicates the number of attributes involved in the WHERE clause of the query (together with the types: index or not). The column *projection* gives the number of attributes in the SELECT clause of the query. Query #9 performs a join. Query #8 has a GROUP BY and query #10 has an ORDER BY.

3 Experiments

We start by comparing **performances** of the selected tools using different configurations. Then, we analyze **latency**, **speed up** and **fault tolerance** of the distributed systems, namely Hive and HadoopDB.

3.1 Performances

The protocol of our experiments relies on three phases:

1. Distributed systems: In the first phase, we consider only distributed systems. We do not resort to secondary index and the partitioning schema is based on the primary keys of the two tables: Source id (resp. Object id) for the *Source* table (resp. *Object* table).
2. Optimization within distributed systems: in this phase, we change the partitioning attribute for the *Source* table. Both tables are partitioned using the object id²⁵. Also, all the suggested indexes for LSST data are considered. The objective of this phase is to analyze the ability of distributed systems to rely on traditional optimization techniques.
3. Distributed and centralized systems: In this phase, we compared the performances of centralized DBMS and distributed ones.

Distributed systems. Our objective is to analyze the behavior of such systems with regards to *performance*, *speed Up*, *latency* and *fault tolerance*. We start by comparing two systems that can be deployed in a cluster environment. Hive is a Map/Reduce based system and HadoopDB which is a hybrid system based on Map/Reduce technology in one hand and classical DBMS technologies in the other hand. We are interested in the response time for our set of queries. The results reveal which properties can be guaranteed by each system. At this stage, we do not resort to indexing.

Preprocessing

- **Hive:** The tables are loaded from the local disk to HDFS directly and they are stored as text files. This task took 25 min for the tables of the PT1.1 data set and 1 h 30 min for the tables of the PT1.2 data set. This task is constrained by the local disk speed which reflects the same time needed for the three clusters.

²⁵ This attribute is the primary key in the *Object* table and a foreign key in the *Source* table.

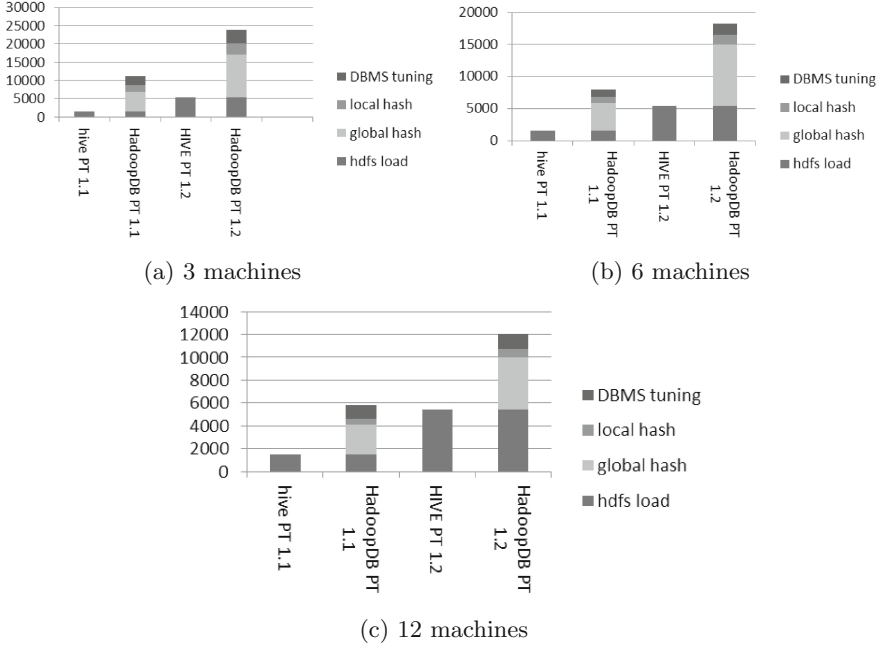


Fig. 1. Data load time for phase 1

- **HadoopDB:** The designers suggest to use two hash functions for partitioning raw data. In the first part of partitioning, we have partitioned raw data into three big chunks. We get for each table three parts. For **HadoopDB**, the recommendation is to use chunks with a size that could fit in the buffer.²⁶ We then did not perform a partitioning using the second function, that is, the Object table. However, for the source table, we used the second function to partition the obtained chunks. Each chunk is partitioned into sub-chunks of 1 GB. This task is done using the second hash function. Hash is done using the first attribute (the primary key) for each table. For example, for the cluster of three machines, the source table of PT1.1 is first partitioned into big chunks of 28 GB. By using the second hash function (local hash), each chunk is partitioned into 28 sub-chunks. Figure 1 shows the time needed to load PT1.1 and PT1.2 in the three clusters of 3, 6 and 12 machines respectively.

The first hash takes 37%–54% of the total loading time, while the second (local) hash takes 6%–16% of the total loading time. This task (second hash) can be triggered, in parallel, in all the machines of a given cluster. Loading data into DBMS takes 10%–21% of the total loading time, this task also can be performed in parallel. The choice of partitioning is done before making the

²⁶ <http://hadoopdb.sourceforge.net/guide/>

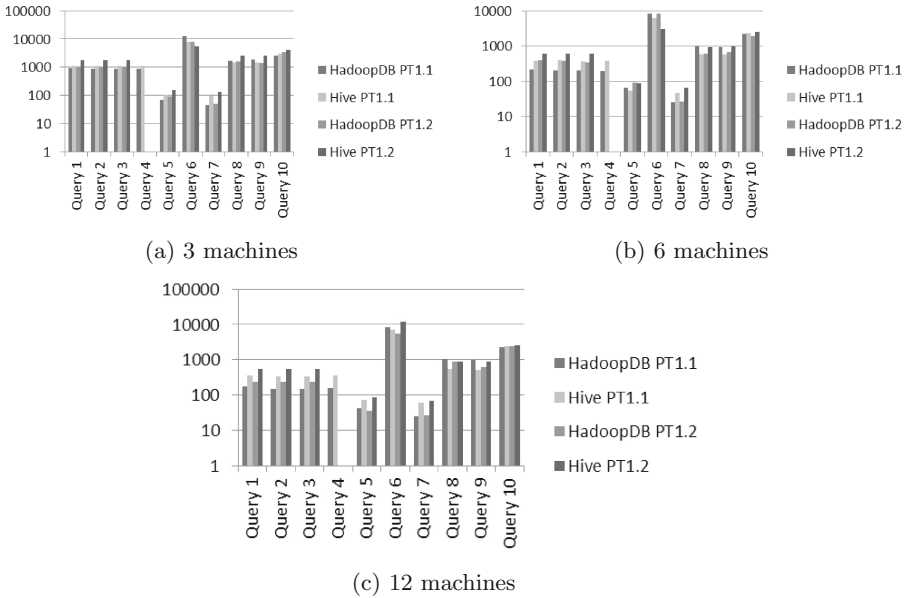


Fig. 2. Execution time for phase 1

decision about queries to be executed. HadoopDB needs between 120 % and 640 % of the loading time needed for Hive.

Analysis of the results. Execution time is shown in Fig. 2. Indeed, one can see that for the queries #1, #2, #3, #4, #5 and #7, HadoopDB outperforms Hive for all the clusters and for the both data sets. These queries express a selection with few results (the size of results does not exceed a few megabytes). Hive outperforms HadoopDB for the query #10 for all the clusters and for both data sets. In this query, an *ORDER BY* operation is used which will be performed in the Reduce phase of a Map/Reduce job. But, in the Map phase, HadoopDB does not apply the projection and prefers to transfer all data to reducers for applying the sort and projection. This decision induces a very high communication cost, which requires almost 2/3 of data (56 GB for PT1.1) to be transferred over the network. For the Query #6, Hive outperforms HadoopDB except for the PT1.2 data set and the cluster with 12 machines. Indeed, in this case, the size of the results is 100 GB which is saved in HDFS. Due to the use of Virtualization environment, the concurrent write access implies that HadoopDB outperforms Hive in this case. HadoopDB generates less writing processes than Hive. For the Queries #8 and #9, HadoopDB outperforms Hive for the PT1.2 data set, whereas Hive outperforms HadoopDB for the PT1.1. For the Query #8, HadoopDB performs a partial *GROUP BY* in the Map phase, which, in the case of PT1.2, is very useful to minimize execution time. This technique can be useful if data is stored using the attribute used in the *GROUP BY*. The same technique is used

in the Query #9. Indeed, a partial sort is performed in the Map phase, which implies less work for the reducers.

The execution time varies between 26 s²⁷ and 12000 s²⁸.

Optimization within distributed DBMS. In this phase, we consider all indexes suggested by LSST.²⁹ Also, we consider another partitioning schema for HadoopDB. Indeed, we compare the execution time for old and new configurations for the same queries.

Preprocessing

- **Hive:** We use the data already available in HDFS. The index is successfully created for small data samples (1000 records). However, for some millions of tuples of the Source table, this task cannot be accomplished correctly. Indeed, Hive loads the created index to the memory of each node. The size of the index is so important that it cannot fit in memory of one node. So, we decided to execute the queries without resorting to any index.
- **HadoopDB:** We partitioned raw data with respect to the primary key of the Object table. This task requires the same execution time as the first phase. All indexes are created in each table of the cluster. This operation is parallelized for all machines. The task took between 12 % and 30 % of the total time needed to load data.

HadoopDB requires between 130 % and 720 % of the loading time needed for Hive. For HadoopDB and in relation to the previous phase, we can see an increase of 3 %–10 % of the time needed to loading data.

Analysis of the results. Execution time is shown in Fig. 3. The queries #1, #2, #3 and #4 take advantage of the indexes. Indeed, these queries express selections on attributes serving as indexes and the selection predicate returns a few tuples. For the queries #8, a join operation is expressed on an indexed attribute. So, in this case, the Map phase generated for this query became faster, whereas we kept the same Reduce phase. For the queries #5 and #7, we obtain the same execution time, as reported in the previous phase. Indeed, these queries express a selection on attributes not serving as indexes. For the query #10, the use of an index has led to a small slowdown, because we make use of all tuples of the Source table. The use of the index requires more latency. So in this case the best operation that can handle the Map phase is the full-scan operation. In classical DBMSs, the index is used to perform a sorting of data. But, in the Map/Reduce framework, this task will be done mandatory in the Shuffle phase. The partial GROUP BY done using an attribute serving as an index had a positive impact on the execution time of this query.

Centralized and distributed systems. In this phase, we compared classical (centralized) DBMS with distributed systems.

²⁷ Query #7 is the less expensive one for HadoopDB.

²⁸ Query #6 is the most expensive one for Hive.

²⁹ <https://dev.lsstcorp.org/trac>

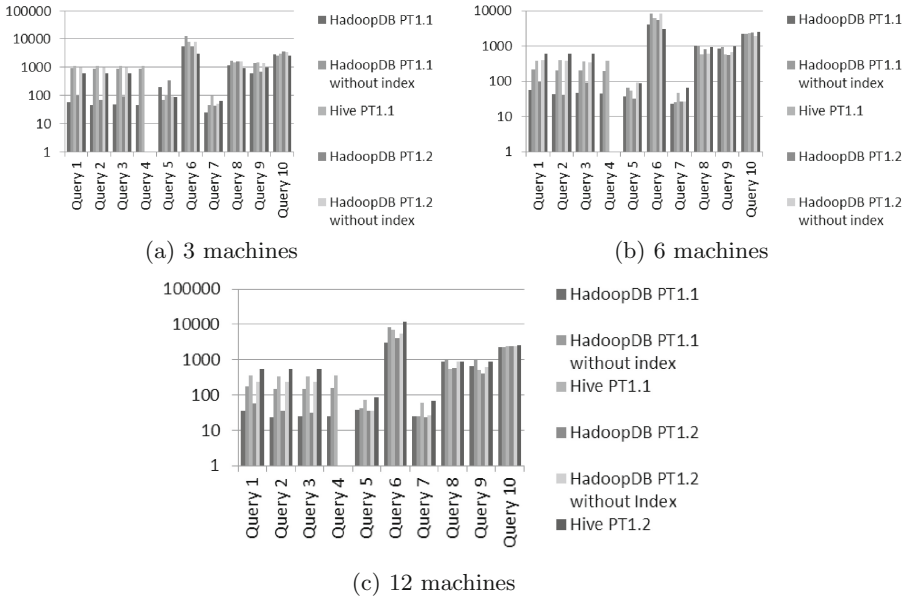


Fig. 3. Execution time: Hive vs HadoopDB by accommodating indexing and partitioning

Preprocessing. We used the configurations of Hive and HadoopDB from the second phase with the cluster of three machines and PT1.1 data set. For classical DBMS, we created all indexes and we loaded data in each node. For the DBMS-X, the creation of indexes took 2h. With mysql, it took 15h to load data and to create all indexes. With postgresql it took 15h to perform data loading and indexing.

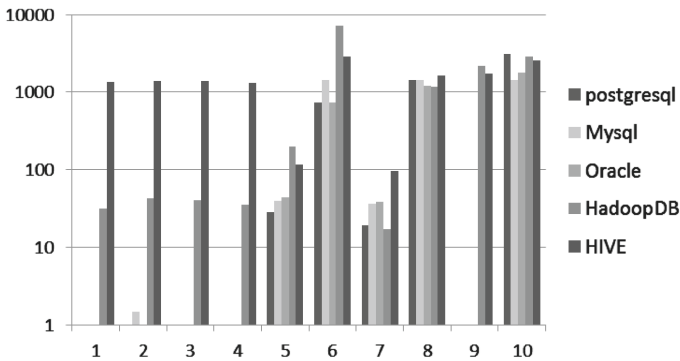


Fig. 4. Execution time: centralized vs distributed systems

Analysis of the results. Execution time is shown in Fig. 4. Classical DBMS are optimized to use disks directly, so no communication cost is induced and no fault tolerance mechanism is considered. This explains the low execution time needed for queries #1, #2, #3 and #4. The use of indexes leads to the manipulation of few disk pages in the memory.

For queries #5 and #6, two costs are considered: (1) communication and (2) disk access for distributed systems. Even if disk access cost is three times lower than disk access required for centralized systems, communication cost affects the total execution time for HadoopDB. The communication cost considered for these queries is the time needed for transferring data from the two machines where Hive was not installed to the node hosting Hive. For query #6, 95 million (32 GB) tuples are transferred. The additional time needed for HadoopDB in the case of queries #1, #2, #3, #4, #5 and #6 is due to the generation of Hadoop jobs, JDBC calls, transformation of partial results to <key, value> tuples and catalog parsing.

Queries #7 and #8 require a complete scan of Object and Source tables. HadoopDB outperforms centralized DBMS. For query #7, communication cost is dominated by disk access cost where only a few tuples (48) are returned as an answer to the query. In the case of query #8, GROUP BY is performed by three machines. Even with the use of network, communication cost for distributed systems is less than disk access cost needed for centralized systems. For the query #9, even with the change of partitioning (Hash attribute) schema and the creation of an additional index, execution time remains high. In this case (join) the communication cost is inevitable to guarantee the completeness of query evaluation. For the query #10, the bad performances of HadoopDB are due to the use of a bad strategy. Indeed, HadoopDB does not perform the projection in the Map phase, this operation is performed in the Reduce phase which induces a very high communication cost.

3.2 Speed Up

To measure the speed up of distributed systems, we consider three configurations: HadoopDB without index, HadoopDB with index and Hive. In each configuration, we change the number of machines in the cluster and we compare execution time of queries. Figure 5 shows the change in execution time of queries by changing, for every data set and every query, the number of machines to host data and process the query.

By analyzing the results of experiments, two facts can be revealed:

1. More data means more processing time: which can be seen when we execute one query, on a fixed cluster, on the both data sets PT1.1 and PT1.2. In both cases, with or without indexing, more time is needed to process more data, and this is the case for all queries. Without indexing, we need more time for data scan, whereas with indexing we need more latency to access to indexes.
2. More machines means less processing time: More disk means faster scanning and more machines means less workload for each machine. In the case of

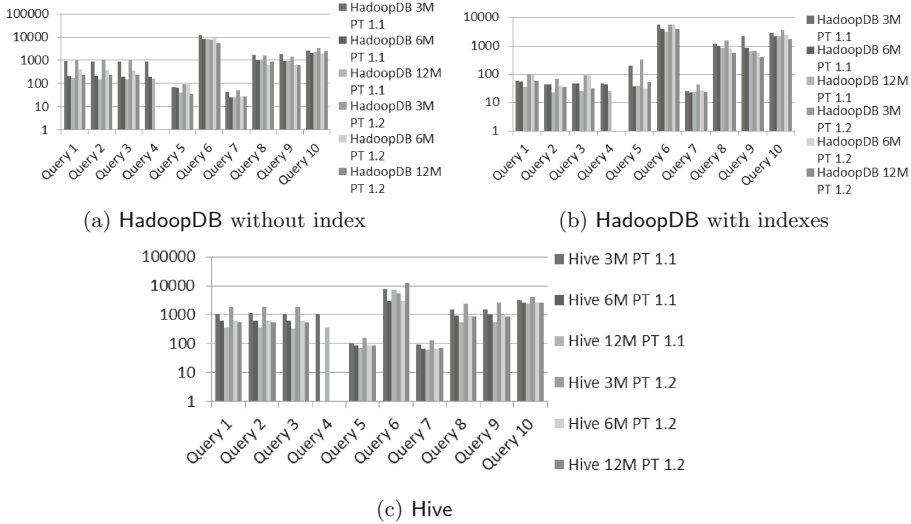


Fig. 5. Speed up

indexing, less latency is required to access the index. For HadoopDB, indexes are created separately for each chunk. For example, if we have 28 chunks and we need four indexes, 112 indexes will be created.

3.3 Fault Tolerance

For Hive and HadoopDB, fault tolerance is managed in Hadoop framework. In our case, we noticed 60 faults during our experiments. We analyzed the tasks with faults and we found that (1) if a task needs an important execution time, then it is subject to more occurrences of faults, and (2) more sub-tasks one has for a query, more faults occur. In average we had 2 faults per hour. For example, query #6 needs 339 sub-tasks to scan the Source table. The induced time was 22 min and 8 s and 2 tasks have failed.

3.4 Latency

To measure the latency of HadoopDB, we executed the queries #1, #2, #3, #4, #5 and #7. The results are shown Fig. 6. One can see that low latency is granted by classical DBMS for queries #1, #2, #3, #4 and #5. For the query #7, one can see that HadoopDB has the lowest latency. Indeed, HadoopDB outperforms centralized DBMS if the computation time (needed for Map/Reduce tasks) exceeds the time needed to set Map/Reduce jobs.

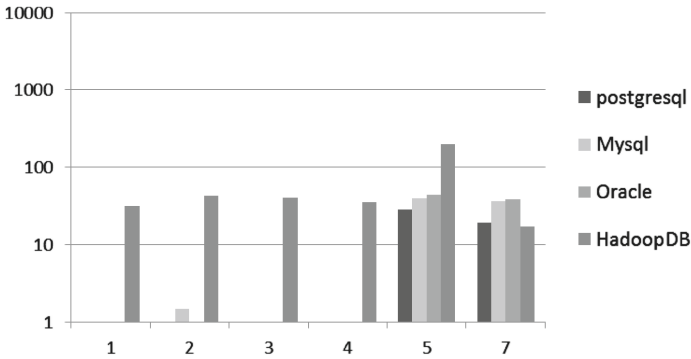


Fig. 6. Latency for HadoopDB

4 Discussion

In this paper we presented some preliminary results of experiments conducted to evaluate the ability of some data management systems to support specific queries in the area of **corpusecular physics** and **cosmology**. We deployed two categories of systems: centralized systems and distributed systems. The goal of these experiments is to report on the ability of these systems to support LSST requirements³⁰ from data management perspective. We compared the response time of queries by considering several configurations. In particular, we used two data sets: PT1.1 and PT1.2. We used three clusters of 3, 6 and 12 machines, respectively, and a set of 10 declarative queries. For each data set and each cluster, we first executed the 10 queries on non-indexed data sets, then on indexed data sets. Starting with comparing the performance of two distributed tools, namely Hive and HadoopDB on non-indexes data, we found that HadoopDB requires between 120 % and 640 % of the loading time required for Hive. Indeed, HadoopDB requires a preprocessing step devoted to the storage of data available on HDFS in a dedicated DBMS for each node. By comparing the execution time we noticed that: (1) for queries with a few results, HadoopDB outperforms Hive. Indeed, this is due to the use of conventional DBMS techniques such as buffering and compressing. Postgresql is configured to use 1 GB of RAM as a buffer, which is compatible with the size of the memory used by the Map/Reduce steps. This configuration allows HadoopDB to display a best latency. For Hive, the chunk size is set to 256 MB, which does not allow Hive to take advantage of the maximum size of memory devoted to the Map/Reduce steps. (2) When it comes to sort data, Hive outperforms HadoopDB. Indeed, sorting algorithms used by Hive are designed specifically for Hadoop. In this case, Hive takes advantages of the Shuffle step. (3) In all other cases, the performances of tools depend on the state (sorted, partitioned, ...) of data.

³⁰ <http://www.lsst.org/files/docs/SRD.pdf>

For our second step, namely the execution of queries on indexed data sets, we tried to use both Hive and HadoopDB. We noticed that Hive is not ready yet to make use of indexing in order to evaluate queries. Indeed, with a size of an index greater than the size of the memory devoted to Map/Reduce phases, Hive cannot use indexes to evaluate queries. On the other side, as HadoopDB is based on traditional DBMS, indexes can be used locally (each database in each node uses its own indexes). The index creation time varies between 12% and 30% of the total time required to load the data. Compared to the previous configuration, we noticed an increase between 3% and 10% of the time needed to load data. Additionally, we compared performances of HadoopDB by considering two types of data, namely indexed and non-indexed data. We noticed an improvement in terms of execution time for queries with selections on attributes serving as indexes. For the other queries, we noticed a slowdown due to the use of indexes. Indeed, there is an incompatibility between the query evaluation strategy adopted by the DBMS when an index exists and Map/Reduce processing within the Hadoop framework. A typical DBMS optimizer can be used to detect the best execution plan for a query. So, local optimization does not always guarantee better performances. It can even lead to poor performances, especially for execution plans that are not compatible with the Map/Reduce framework. In this phase, we noticed the importance of classical optimization techniques (index, partitioning, compression) and the necessity to integrate global optimizers, which are compatible with Hadoop. By comparing performances of conventional DBMSs with those of distributed systems, we noticed that, for queries with a few tuples to handle, traditional DBMS outperforms distributed ones. For queries that require a large number of tuples, it will be necessary to consider other architectures. For distributed architectures, the cost of transferring data from one machine to another is very important if one wants to estimate the cost of a query. Then, we analyzed the impact of resources by providing more equipments (machines, disk and RAM). We found that all queries take advantage of additional resources. By increasing the number of machines we decreased their workload.

The most important property guaranteed by classical DBMSs and not yet guaranteed by the distributed ones is latency. Indeed, fault tolerance mechanism built within the Hadoop framework does not allow Hadoop-based systems to guarantee low latency for queries.

5 Conclusion

In this paper we presented some preliminary results of experiments conducted to evaluate the ability of some data management systems to support specific queries in the area of **corpusscular physics** and **cosmology**. We deployed two categories of systems: centralized systems and distributed systems. The goal of these experiments is to report on the ability of these systems to support LSST requirements from data management perspective. We reported on the impact of data partitioning, indexing and compression on query evaluation performances. We also highlighted the need for new techniques to optimize emerging systems.

We are refining our experiments by considering new data sizes. Indeed, we are experimenting with datasets of sizes 500 GB, 1 TB and 2 TB. We are also setting up an experimental environment (virtual machines) with 10 nodes, 25 nodes, 50 nodes and 100 nodes. The impact of parameters of hadoop as bloc size, memory used by Map/Reduce tasks and the number of Map tasks per node will be studied. We will include the support of queries with user defined functions since they are very relevant in the framework of LSST.³¹ After this experiential campaign, we will move to Grid5000³² for a large scale experiments with 300 machines.

References

1. Abadi, D.: Consistency tradeoffs in modern distributed database system design: cap is only part of the story. *Computer* **45**(2), 37–42 (2012)
2. Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Silberschatz, A., Rasin, A.: Hadoopdb: an architectural hybrid of mapreduce and dbms technologies for analytical workloads. *Proc. VLDB Endow.* **2**(1), 922–933 (2009)
3. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. In: *OSDI*, pp. 137–150 (2004)
4. DeWitt, D., Gray, J.: Parallel database systems: the future of high performance database systems. *Commun. ACM* **35**(6), 85–98 (1992)
5. 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: *SIGMOD*, pp. 165–178. *ACM* (2009)
6. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H., Murthy, R.: Hive-a petabyte scale data warehouse using hadoop. In: *ICDE*, pp. 996–1005. *IEEE* (2010)

³¹ <https://dev.lsstcorp.org/trac/wiki/db/queries>, e.g., Q007, Q008, Q013

³² <https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home>

**Third International Workshop on Data
Management for Emerging Network
Infrastructure (DaMEN)**

A Framework to Measure Storage Utilization in Cloud Storage Systems

Xiao Zhang¹(✉), Wan Guo¹, Zhanhuai Li¹, Xiaonan Zhao¹, and Xiao Qin²

¹ School of Computer Science Northwestern Polytechnical University,
Xi'an 710129, Shaanxi, China
zhangxiao@nwpu.edu.cn

² Department of Computer Science and Software Engineering Auburn University,
Auburn, USA

Abstract. Cloud storage systems aim to offer cost-effective storage services. The key is sharing resources between multiple users by virtualization technologies. Storage resources in cloud systems can not be reclaimed even when users do not access their data for a long time. Storage resources must be shared through space sharing rather than time sharing. Existing technologies improve storage utilization at various layers and data sets, making it difficult to analyze the efficiency of a cloud storage in a holistic way. To address this problem, we propose an evaluation framework to study the impacts of a wide variety of I/O techniques on an enterprise-scale cloud storage. The framework offers storage utilization evaluation from both the users and the vendors' perspective.

1 Introduction

The growth rate in the volume of data has been sped up by the Internet and high resolution digital media. This is popularly known as information explosion. Martin estimates that the world's information storage capacity grew at a compound annual growth rate of 25 % per year between 1986 and 2007 [1]. McKinsey Global Institute estimates that nearly all sectors in the US economy had at least an average of 200 terabytes of stored data by 2009. In total, the study estimates that 7.4 exabytes of new data were stored by enterprises and 6.8 exabytes by consumers in 2010 [2].

Building on an "Allocate on Demand" mantra, cloud storage is changing the paradigm of storage economics by offering a high dependability storage service at a much cheaper price with small capacities. Apart from traditional online storage services such as SugarSync and Mozy, new deep-pocketed players in the area including Microsoft, Apple, and Google are forcing Dropbox to face the heat. Microsoft with Azure, Apple with iCloud, and Google with its Drive seem to offer more than Dropbox in terms of price-to-value ratio. The price per GB of Google cloud storage is \$0.085¹, and the price per GB of Amazon is \$0.095².

¹ <https://developers.google.com/storage/docs/pricingandterms>

² <http://aws.amazon.com/s3/#pricing>

Building a data center to provide cloud service represents a significant investment and ongoing costs. This reveals the importance of optimizing resource utilization per dollar. Unfortunately, past researchers focus on CPU and network utilization optimizing; seldom studies focus on utilization of storage system, which are full of fragmentation and replicas.

Cloud computing systems aim to provide low cost service. When a VM (short for virtual machine) is closed, all allocated resources (e.g., CPU, memory) will be released and can be re-used by new VMs. Computing resources are shared both in the space and time. These resources are typical pay-as-you-go resources. But in a cloud storage system, the storage resource cannot be released even if users do not access them in a long time. Comparison between local storage and cloud storage shows that it is economic only when the data stored in primary storage is less than 7 TB [3].

In this paper, we present a framework to analyze the utilization of cloud storage systems. Our main contributions are:

- We define a framework to evaluate the storage utilization from vendors’ and users’ perspectives.
- The broader perspective of cloud storage and a detailed analysis of its layers provide a clearer path toward an optimized cloud storage. This can be used to determine whether a technology will be applied. And it also can be used to analyze the effect for given data sets.
- We illustrate how to use our framework to evaluate the system scale storage utilization. We give an example to evaluate the efficiency of a given technology with various policies and data sets.

The rest of this paper is organized as follows. Section 2 reviews the related works about utilization. Section 3 analyzes the character of a cloud storage system. Section 4 divides the cloud storage into three layers, and it defines terms for each of the layers. Section 5 presents the framework to analyze storage utilization. We define eight ratios to describe the utilization of each of the layers. There are two overall metrics defined to measure the total storage utilization. Section 6 illustrates how the technologies and policies affect storage utilization. Section 7 gives some examples of how to analyze a whole cloud storage system.

2 Related Works

Cloud computing is tightly coupled with low cost. Many studies try to find an effective approach to manage the cost of cloud computing. Li *et al.* present a method and tool to analyze the cost of cloud computing [4]. Twinstrata *et al.* compare the cost of a private data center and public cloud storage for primary storage and disaster site. The results show that cloud storage appears considerably less expensive than local under small capacities (<1 TB) [3].

Albert *et al.* point out that the cost breakdown reveals the importance of optimizing work. But the resources inside the data centers often operate at low utilization due to resource stranding and fragmentation [5]. Many methods try

to shut down unused disks or concentrate data together to reduce the cost, such as MAID [6] and PDC [7]. EEVFS try to manage disk placement to improve energy efficiency [8].

Provisioning and placement issues in enterprise storage servers have been studied in the past [9, 10]. Compression improves the efficiency of network bandwidth and disk space. Cao *et al.* proposed data compression in order to increase the I/O performance of Hadoop [11]. Chen *et al.* analyze how compression can improve performance and energy efficiency for MapReduce workloads. For read-heavy text data, compression provides 35% to 60% of energy savings. Mark presented a stream de-duplication for large scale storage systems by breaking up an incoming stream into relatively large segments and de-duplicating each segment [12]. Most of these technologies analyze the effect of the technology itself; there is no previous paper working on how to analyze the utilization of storage resources in the data center scale.

3 Analysis of Cloud Storage Systems

It is important to make the system work more efficiently to reduce the cost. Users cannot share persistent storage resource at different times. Many technologies have been developed to improve the utilization of storage systems. All of these optimization technologies fall into three categories: share “free resource”, share “used space”, and save efficiently.

The first class shares “free resource”. Virtualization technology makes the free devices a resource pool. The resource can be dynamically added when needed in the future. Thin provisioning is the act of using virtualization technology to give the appearance of having more physical resources than are actually available. So free physical resource can be reserved for several logical file systems; this delays the request of adding disks. Another example is quota mechanism in file systems. In the file system shared by multiple users, the sum of quotas of all users may be greater than the available space of the file system.

The second class shares “used space”. Data de-duplication is a specialized data compression technique for eliminating duplicate copies of repeating data. Related and somewhat synonymous terms are intelligent data compression and single-instance data storage. It identifies the same files and blocks and only saves once. Some people like to save pop music and public documents in their own space. In these conditions, if the file has been saved by other people before, a link to the existing file will be created to represent sharing between different users. An interesting thing is that the utilization of a storage system with de-duplication can exceed 100%.

The third class tries to save data more effectively. Data compression can decrease the space requirement for files, especially files in text format. Meta-data refers to data about data. It saves the file name, size, and block positions on disks. It is important but needs extra space besides user data. Different file systems have different storage efficiency.

On the other hand, there are also technologies that decrease the utilization which are designed for high dependability and performance purposes. RAID is used

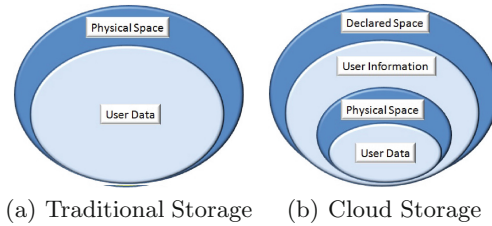


Fig. 1. Contrast between traditional and cloud storage

to protect data from disk failure, but it decreases the utilization by 20%–50%. CDN (content delivery network) is used to deliver data quickly via the Internet; the decrease rate of the technology depends on the number of replicas. Replicas are also created to prevent data from being unusable when the server is down. All of these technologies are mixed together, and this makes it difficult to analyze the utilization of storage systems.

These technologies take effect on different layers and data sets, which makes it difficult to analyze the efficiency of the whole system. There is no available method to analyze the utilization of cloud storage systems. This paper presents our efforts towards filling in the gap. In this paper, we discuss the concept of storage utilization and related metrics. The storage utilization can be described by ratios among these metrics. We split the cloud into the physical, logical, and user layers, and we develop measurable metrics for each layer.

Enterprise-scale cloud storage systems contain thousands of servers and storage devices and up to tens of thousands of disks, which are difficult to expand. Traditional design methods usually lead to solutions that are grossly over-provisioned. Cloud storage transfers the costs of over-provisioning and the risks of under-provisioning to cloud providers. In a traditional storage system, the IT manager buys disk arrays and servers before they save data in it (Fig.1(a)). The IT manager must buy more devices for future use. While in a cloud storage system, although the IT manager declares how much space every user owns (Declared space), they will add device capacity (Physical Space) while the size of user data reaches the capacity of the current systems (Fig.1(b)). Meanwhile, it uses compression and de-duplication technologies to store user files (User Information) in an effective way. In traditional storage system, declared space equals to physical space and user information equals to user data.

4 Storage Utilization Terms and Metrics

Before storage utilization can be measured, basic terms and metrics must be defined first. We divided the storage system into three layers. Physical layer refers to hardware devices such as disk array and tape library. Logical layer refers to software parts such as the file system and other data management technologies.

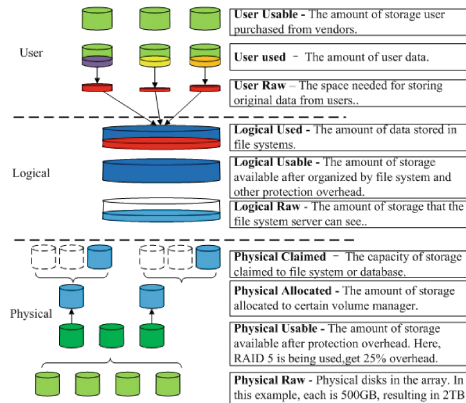


Fig. 2. Terms and layers of cloud storage

User layer refers to user space and user data, it represents the space announced to the user. The following nomenclature will be used to describe storage utilization metrics of the three layers (Fig. 2).

4.1 User Layer

User Usable – The amount of storage the user purchased from vendors. When the user tries to save files exceeding the quota, the user will receive an error. The quota system automatically monitors the disk usage for users on a file system by a file system basis. Many modern file systems support quotas for individuals. The NTFS (New Technology File System) allows administrators to control the amount of data that each user can store on the file system.

User Used – The amount of data of users. All cloud storage providers count the used space for user. DropBox sums up the size of regular files and shared files together as the used space. It also gives the used ratio by percentage. An interesting thing is that files shared with friends will be counted several times in Dropbox.

User Data – The space needed for storing original data from users. In most conditions, this is smaller than User Used. Sharing, compression and de-duplication technologies make it possible to save large data in smaller space. Files shared between different users only need to be stored one time. Compression file systems (e.g., NTFS, ZFS) support compression of some files to save space. De-duplication technology can find the same files between different users and store them only once. It is more efficient especially when users store pop music and movie files in their space.

4.2 Logical Layer

Logical Used – The amount of data stored in file systems. Besides user files, it also includes metadata and replicas. Metadata refers to data about content;

it records file position and other properties in the file system. Some metadata are created when we format the file system; the size can be calculated by capacity of the file system. The size of metadata associated with the file grows incrementally in pace with the number of files.

Logical Usable – The amount of space available after being organized by a file system and other protection technologies. Elasticity is an important character of cloud storage. Users may require more storage capacity whenever they want. So scalability is an important feature when we deploy a file system. If a file system is constructed over thin provisioning technology, it only needs to add physical disks when needed. For systems without thin provisioning, there are some tools that can resize the file system such as `resize2fs` in Linux. After increasing volume space by LVM (Logical Volume Manager), these tools can extend the usable capacity of a file system.

Logical Raw – The amount of storage that the file system server can see. Thin provisioning technology can ‘cheat’ a file system into appearing to have enough physical disks. In this condition, the logical raw is the size thin provisioning reported.

4.3 Physical Layer

Physical Claimed – The capacity of storage claimed to file system or database. For example, if a hard disk has a capacity of 2 Terabyte, then its Physical Claimed is 2 Terabyte.

Physical Allocated – The amount of storage allocated to a certain volume manager. Storage virtualization makes it possible to access without regards to physical storage or heterogeneous structure. LVM on Linux can resize volume groups online by absorbing new physical volumes (PV) or ejecting existing ones. So the available space can be reserved for any of the servers which can access it.

Physical Available – The amount of storage available after protection overhead. Space available after constructing RAID and spare disks. IT managers can allocate Logical Units from these spaces. This value can be collected by array management tools.

Physical Raw – The total physical space of all physical disks in the disk array and tapes in the tape library. It is the sum of the sizes of all installed disks. If we set the price as the weight of each media, we can get the cost of Physical Raw.

5 Storage Utilization Metrics

Utilization can be described by the ratios of standard metrics of different layers. We illustrate the key storage utilization ratios in Fig. 3. These ratios can indicate the effect of different data management technologies and policies.

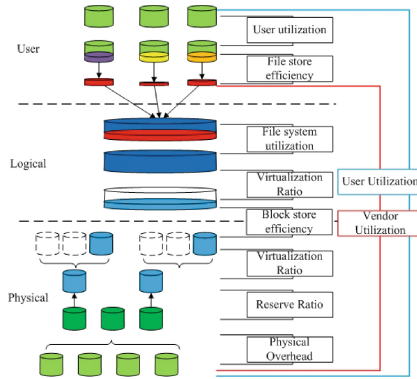


Fig. 3. Utilization metrics of cloud storage

The utilization ratios can be obtained by comparing one metric with another to obtain a percentage. These metrics are measured in capacities. The percentage may be greater than 100 % for systems with virtualization. The cumulative effect of these utilization results in two overall utilization metrics. User utilization can be used to calculate the cost of each user or all users. Vendor utilization represents the cost of storage vendors. The difference between user utilization and vendor utilization is the profit of cloud storage vendors. The most common feature of a storage system is capacity. These metrics are defined by ratios between the different layers. These metrics also can be expressed in dollars if we add the price as weight of capacity.

5.1 Physical Layer Metrics

Physical Overhead – Physical Overhead is the percentage of installed storage capacity that is not usable. Overhead is usually due to the desired level of data protection and performance (e.g. RAID, mirroring, spare disks). A high level of data protection (e.g., RAID1, spare disks) results in high protection overhead. Having too many spare disks in the system also results in high overhead.

Reserve Ratio – Comparison between the **Physical Claimed** space taken by a file system and the **Physical Allocated** space. Virtualization technology treats all the allocated logical disks as a resource pool. The ratio reflects the resource pool level. When a file system needs more space, reserved space can be added to the file system by volume managers like LVM (Logical Volume Manager) dynamically. It depends on how fast the size of the file system increases and the policy of how much space should be reserved for future use.

Virtualization Ratio – Comparison between claimed space and the space actually allocated. Thin provisioning is used to give the appearance of having more physical resources than are actually available.

5.2 Logical Layer Metrics

Block store efficiency – Comparison between available space to store data and the space of raw volumes. File systems use some special blocks, so called super blocks, to store meta-data, allocation bitmap, and log files. An analysis shows that the ext3 file system can only use 98.4 % of the space of a 1 TB volume³. This metric depends on the volume size and file system format; it varies enormously between file systems.

File system Utilization – Comparison between the actual space taken by user data and the total space. Elasticity refers to the ability to quickly scale up or down one's available storage capacity. This is an important economic benefit of cloud storage as it transfers the costs of resource over-provisioning and the risks of under-provisioning to cloud providers.

5.3 User Layer Metrics

File Store Efficiency – Comparison between the **User Raw** and **Logical Used**. Compression can use a small space to save files. The compression rate depends on the file format and the contents. To protect data or for performance purposes, some cluster file systems create replicas on different nodes. GFS (Google File System) divided files into chunks; each chunks is replicated at least once on another server, and the default is three copies of every chunk. If the file system creates two replicas for every file, the redundancy rate should be 50 %. On another hand, de-duplication technology eliminates duplicate copies of repeated data. If several users store the same video files in their own space, it is more efficient to save one file and create several links to the file.

User Utilization – For an individual user, comparison between **User Used** and **User Usable**. Storage providers show this metric to users as a reminder of how much available space is left. Some storage providers count the space of shared files between users for each user.

5.4 Overall Metrics

Overall User Utilization – Comparison between summation of all user usable space and the sum of physical space. It represents the price of cloud storage. From this metric, we can decide how much a user should pay for the storage capacity. Pricing is complex in the real-world. Most cloud storage vendors provide 2-6GB of free storage service. Dropbox hit 100 million users in November 2012, but 96 % of the customers are only using free services.⁴

Overall Vendor Utilization – Comparison between summation of logical used and the sum of physical disks. This metric reflects the cost of vendors. There are

³ <http://www.lisnichenko.com/articles/ext3-file-system-overhead-disclosed-part-2.html>

⁴ <http://www.forbes.com/sites/victoriabarret/2012/11/13/dropbox-hits-100-million-users-says-drew-houston/>

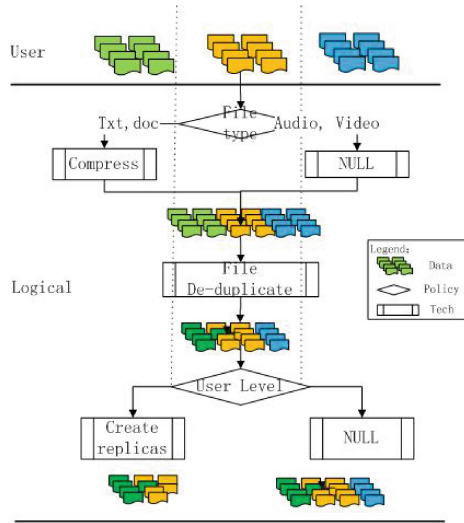


Fig. 4. Storage utilization analyze framework

not too many ways to decrease the overhead of a storage associate with specified dependability. In some conditions, the protection is being handled by a file system on the logical layer, such as Google GFS.

6 Storage Utilization Analytical Framework

In this section, we present the framework to analyze the storage utilization as a whole solution. Different vendors select different technologies and policies according to the file types, user SLA (Service Level Agreement). When users store files in the cloud storage system, multiple technologies are applied before these files begin to be saved on the disks finally. All of these technologies in one cloud storage system can be shown in a workflow chart as Fig. 4. Some manage files and replicas, such as de-duplication and CDN. Some map files to block devices, such as RAID and block level de-duplication.

There are three characteristics that affect storage utilization—user data characters, policies, and technologies. Cloud storage system vendors decide which technologies and policies should be used in their system. But it receives a different efficiency for different user data characters. Vendors need to provide service under certain SLA (Service Layer Agreement). They need to make a decision for dependability or performance purpose, such as how many replicas should be created in different servers.

Characterizing and understanding file system content and workloads is imperative for the design and implementation of effective storage systems [13]. There have been numerous studies over the past years of file system characteristics. Wallace collects the meta-data of EMC Data Domain systems, and statistical

analysis considers information such as file age, size, counts, de-duplication effectiveness, compressibility, and other metrics.

When we calculate the efficiency of a given technology, we need address the right position of the technology in the workflow chart. According to the input and output of a given technology, it will be addressed in the corresponding layer. If there are more than one technologies used in the same layer, they should be placed side by side if they handle different conditions, or they should be placed like a pipeline if process data one by one. The data characters impact the efficiency of each technology. When we analyze the efficiency of a given technology, we use data characters of the upper layer instead of raw user data characters.

We illustrate how to analyze three technologies used to map user raw file to cluster file system: file level de-duplicate, compression, and replicas on different site (CDN). These technologies are located on the same layer between User raw and Logical used. The sequence for applying these technologies is compression, de-duplication, and create replicas. The de-duplication ratio can be calculated by Formula 1, where N represents the number of the same files. We assume the metadata size is 255 bytes, file size is 1024 bytes, and there are 100 same files. The de-duplication ratio is 97.63%. But if the files were compressed before de-duplication and the compression ratio is 10%, then the de-duplication ratio is 80.44%. File store efficiency should be mixed results of these technologies.

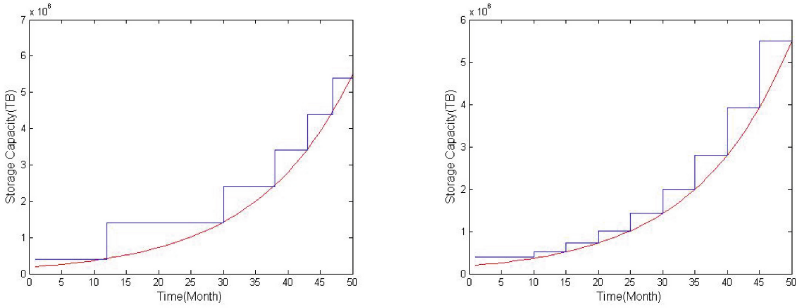
$$Dedupratio = \frac{N \times (filesize + metadatasize)}{filesize + N \times metadatasize} \quad (1)$$

7 Case Study

We illustrate how to use the framework to analyze a cloud storage system used on a campus. Teachers and students store lecture, video, pop songs and homework in the system. It is a private cloud system, so the character is easier than commercial cloud system. We use the framework to predict how many storage devices will be enough. The method also helps IT managers choose technologies and configurations. Cost of each users also can be calculated by it.

7.1 How Many Storage Devices are Enough

The task of a cloud storage manager is to allocate storage resources when users request. It means that they must ensure that the Logical Usable Space is bigger than User Data at any time. The amount of user data depends on user number, user data size, and overall vendor utilization. If we adopt a conservative estimation of the maximum size, the growth rate of the new user data can be estimated by the current user data. We use α to represent the growth rate of the current user data, β to represent the growth rate of users. Then the growth rate of user data can be estimated by $(1 + \alpha + \beta) \times S_{cur_userdata}$. Methods of adding more storage resources for a system fall into two broad categories: scale



(a) Add Storage at Constant Capacity (b) Add Storage at Constant Interval

Fig. 5. Available space and user data over time

out and scale up. It takes different times to make the resource available by different methods. Thin-provisioning technology can make the file system use newly added physical disks transparently. Logical Usable Space is a step function of time. According to when and which method is taken, logical usable space increases after scaling takes effect. If we assume the current user number is 15 thousand, data size is 200 TB. The growth rate of user data and user number are 2% and 5% per month. We can get the increasing curve of user data in Fig. 5(a). From the figure, we can find that the growth rate of user data increases over time. Figure 5(a) shows that if we extend the same capacity when needed, the interval of extension become shorter over time. Figure 5(b) shows that if we extend the amount of necessary capacity for the next 5 months, the capacity of extension grows over time.

7.2 Selection and Configuration of Technology

There are many data management technologies that can be applied in the system. We illustrate how to choose the technology and the best configuration. The first step is to address the technology in the process network shown in Sect. 6. Secondly we need to analyze the characteristics of the files to be handled by the technology. Then we can analyze the effect of the technology with a different configuration. At last, we need to analyze the impact of other technologies under it.

Most students save their lecture in the system. We assume there are 100 students in one class. The file size is 10 MB, and the metadata size is 512 bytes. It needs $(10\text{ M} + 512\text{ K}) \times 100 = 1050\text{ M}$ to store these files. We assume that the compression ratio is 50%; this means the compressed file size is 5 MB. If we apply compression alone, it needs $(5\text{ M} + 512\text{ K}) \times 100 = 550\text{ M}$. If we apply de-duplication alone, it needs $10\text{ M} + 512\text{ K} \times 100 = 60\text{ M}$. If we apply both compression and de-duplication together, it will need $5\text{ M} + 512\text{ K} \times 100 = 55\text{ M}$. In this case, compression has little effect on the overall optimization as shown in Fig. 6. We can find it is enough to use file deduplication in the system.

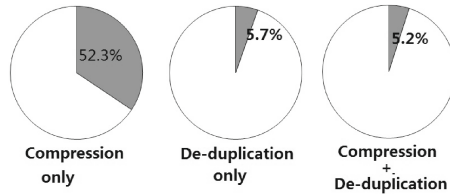


Fig. 6. Optimization results of different technologies

8 Conclusion

This study was motivated by the lack of a holistic way of evaluating the performance of various I/O techniques applied at multiple layers of a storage cloud. We proposed a novel evaluation framework, which aims at investigating the impacts of a wide range of policies and mechanisms on large-scale storage clouds. At the center of our framework is an array of evaluation metrics for the multiple storage layers.

We develop empirical models to quantitatively measure our proposed new metrics. To illustrate the correctness of the models as well as the usage of our framework, we conduct a case study to demonstrate how to apply our framework to evaluate storage utilization of cloud systems. We show that the framework can be deployed to investigate impacts of I/O policies, mechanisms, and data sets. Our experiments show that the framework provides both users and vendors with storage utilization evaluation from various perspectives.

There exist a few opening issues in this study. The current version of the framework can evaluate static storage utilization; however, we are unable to apply the framework to evaluate I/O activities. We intend to extend our framework to evaluate dynamic I/O workload conditions so it can be used to analyze utilization of hierarchical storage. Many data management techniques to promote utilization that have been proposed in the past decades. We plan to incorporate some modules into the framework to acquire the impact of each technique in the whole storage system.

Acknowledgment. This work was supported by the National High-tech R&D Program of China (863 Program) under Grant No. 2013AA01A215 and No. 2012AA011004; the NFS of China under Grant No. 61033007; the NFS of China under Grant No. 61272123 and No. 61303037.

References

1. Hilbert, M., López, P.: The worlds technological capacity to store, communicate, and compute information. *Science* **332**(6025), 60–65 (2011)
2. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.H.: *Big data: The next frontier for innovation, competition, and productivity*. McKinsey Global Institute, pp. 1–137 (2011)

3. Twinstrata: Economics of public cloud storage. <http://pt.slideshare.net/rinfantino/the-economics-of-public-cloud-storage>
4. Li, X., Li, Y., Liu, T., Qiu, J., Wang, F.: The method and tool of cost analysis for cloud computing. In: IEEE International Conference on Cloud Computing, CLOUD'09, pp. 93–100. IEEE (2009)
5. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The cost of a cloud: research problems in data center networks. *ACM SIGCOMM Comput. Commun. Rev.* **39**(1), 68–73 (2008)
6. Colarelli, D., Grunwald, D.: Massive arrays of idle disks for storage archives. In: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing, pp. 1–11. IEEE Computer Society Press (2002)
7. Pinheiro, E., Bianchini, R.: Energy conservation techniques for disk array-based servers. In: International Conference on Supercomputing: Proceedings of the 18th Annual International Conference on Supercomputing, vol. 26, pp. 68–78 (2004)
8. Manzanares, A., Ruan, X., Yin, S., Xie, J., Ding, Z., Tian, Y., Majors, J., Qin, X.: Energy efficient prefetching with buffer disks for cluster file systems. In: 2010 39th International Conference on Parallel Processing (ICPP), pp. 404–413. IEEE (2010)
9. Alvarez, G.A., Borowsky, E., Go, S., Romer, T.H., Becker-Szendy, R., Golding, R., Merchant, A., Spasojevic, M., Veitch, A., Wilkes, J.: Minerva: an automated resource provisioning tool for large-scale storage systems. *ACM Trans. Comput. Syst. (TOCS)* **19**(4), 483–518 (2001)
10. Anderson, E., Hobbs, M., Keeton, K., Spence, S., Uysal, M., Veitch, A.: Hippodrome: running circles around storage administration. In: Proceedings of the Conference on File and Storage Technologies, pp. 175–188 (2002)
11. Cao, Y., Chen, C., Guo, F., Jiang, D., Lin, Y., Ooi, B.C., Vo, H.T., Wu, S., Xu, Q.: A cloud data storage system for supporting both oltp and olap. In: IEEE 27th International Conference on Data Engineering (ICDE), pp. 291–302. IEEE (2011)
12. Lillibridge, M., Eshghi, K., Bhagwat, D., Deolalikar, V., Trezise, G., Camble, P.: Sparse indexing: large scale, inline deduplication using sampling and locality. In: Proceedings of the 7th Conference on File and Storage Technologies, pp. 111–123 (2009)
13. Wallace, G., Douglis, F., Qian, H., Shilane, P., Smaldone, S., Chamness, M., Hsu, W.: Characteristics of backup workloads in production systems. In: Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FAST12) (2012)

Personalized Recommendation via Relevance Propagation on Social Tagging Graph

Huiming Li, Hao Li, Zimu Zhang, and Hao Wu^(✉)

School of Information Science and Engineering, Yunnan University,
No.2, North Green Lake Road, Kunming 650091,
People's Republic of China
haowu@ynu.edu.cn

Abstract. This paper presents a novel random walk based relevance propagation model for personalized recommendation in social tagging systems. In the model, the tags are used to express the profiles of both users and resources, and then candidates of resources are recommended to the users based on the profile relevance between them. In particular, how the users to find the resources of interest is modeled as a random walk by which the relevance spreads in User-Resource-Tag relation graph. Experimental results on two real datasets collected from social media systems show the merits of the proposed approach.

1 Introduction

Collaborative tagging systems [1], such as Delicious, Flickr, Youtube, Lastfm, Connotea, CiteUlike and MovieLens, have become a kind of booming business on the Internet. These systems provide a wealth of information, where any persons can freely find, annotate, organize various resources of interest and share their findings (this practice is coined as Folksonomy by Thomas Vander Wal). As an information carrier, the tags play a key role in such systems. Since they cannot only express the main features of the resources, but also cover relationships of users-resources/items (we use them alternatively) and items-items.

The size and complexity of folksonomy-based systems can unfortunately lead to information overload and reduced utility for users. Too many resources can make users helpless in their process of finding useful contents. Consequentially, the increasing need for recommender services from users has arisen. For these reasons, researchers have sought to apply the techniques of recommender systems to deliver personalized views. The current researches of personalization in such systems can be classified into tag recommendation and item/resource recommendation. Given a user and a resource, the former predicts what and how tags will be adopted by the user to explain the resource, whereas the latter emphasizes suggesting unseen items of interest to the user. Compared to tag-oriented recommendation research, how to develop tag-aware personalized recommendation technologies to come forward with the application needs remains many issues [2]. To this end, this paper presents a novel random walk based relevance propagation model for personalized recommendation in social tagging systems. In the

model, the tags are used to express the profiles of both users and resources, and then candidates of resources are recommended to the users based on the profile relevance between them. In particular, how the users to find the resources of interest is modeled as a random walk by which the relevance spreads in User-Resource-Tag relation graph. Experimental results on two real datasets collected from social media systems, show that our model can improve the accuracy of resource discovery, and thus enhance the personalized recommendation in social tagging systems.

The rest of paper is organized as follows. Section 2 presents our models in detail. How to assess the relevance of user-user and user-resource, how to extract and build neighborhood-based social tagging graph and how the relevance spreads with a random walk on extracted graph are proposed. Next, in Sect. 3, the solid experiments are conducted to watch the effectiveness of our method. Then, we review some works most akin to us, and make some discussions. Finally, we conclude the works and point to future directions.

2 Methods

2.1 Neighborhood-Based Social Tagging Graph

One of the most commonly used algorithms in personalized recommendation is a neighborhood based approach [3], which works by first computing similarities between all pairs of users, and then to predict by integrating ratings of neighbors. Here, we follow this common idea to create a neighborhood-based tripartite graph G_{URT} for personal resource recommendation in social tagging systems.

Given a random user u_i , we first define a user profile as: $\mathbf{u}_i = (r_{i,1} : w_{i,1}, r_{i,k} : w_{i,k}, \dots, r_{i,n} : w_{i,|R|})$, where $r_{i,k}$ is the k -th resource collected by u_i , $|R|$ is the cardinality of resource collection, $w_{i,k}$ is the preference degree of u_i on resource $r_{i,k}$. Then, we estimate the pairwise relevances of users using the cosine similarity (Eq. 1),

$$R(u, u_i) = \text{cosine}(\mathbf{u}, \mathbf{u}_i) = \frac{\sum_{k=1}^{|R|} w_k \times w_{i,k}}{\sqrt{\sum_{k=1}^{|R|} w_k^2} \sqrt{\sum_{k=1}^{|R|} w_{i,k}^2}} \quad (1)$$

Both w_k and $w_{i,k}$ in user profile \mathbf{u} and \mathbf{u}_i can be obtained by TF-IUF(Term Frequency-Inverse User Frequency) as followings,

$$w_{i,k} = tf_{i,k} \cdot \log \frac{|U|}{uf_{i,k}} \quad (2)$$

where, $tf_{i,k}$ is the normalized occurrence frequency of the k -th resource in the user profile \mathbf{u}_i , $|U|$ is the cardinality of user collection, $uf_{i,k}$ is the total number of the user profiles in which the k -th resource occurred.

In the same way, we can represent a profile of the resource as a tag-aware vector, and estimate the relevance $R(u, r)$ between a user u and a resource r .

Given a target user u , the top- n users most similar to u are firstly found using metric (Eq. 1), these users together with u forms a virtual community $C(u)$. $C(u)$

is added to G_{URT} . Then, the resources and the tags used by $C(u)$ to annotate them are added to G_{URT} . Finally, the User-Resource-Tag relations are created according to social annotation traces. Built on the tripartite graph, if we want to automatically point a user to the most interested resources, we should imagine how the user searches resources of interest as he/she gradually surfs on a social tagging system. For simplicity, we present two independent and repeated surfing processes. In the first case, we assume that users preferably consult with community members as they search resources:

- At any time: (a) randomly visit a resource, or just pick a random community member;
- After consulting with a community member: (a) pick a resource tagged by this member, or (b) consult with another member recommended by current member;
- After visiting a resource: (a) consult with a community member who tagged this resource, or (b) visit another resource linked to this resource.

In the second case, we assume that users search resources by prefer to exploit tags:

- At any time: (a) randomly visit a resource, or just pick a random tag;
- After viewing a tag, pick a resource annotated by this tag;
- After visiting a resource, pick an interested tag to further view.

To model resource gathering process as well as to reduce the complexity of our proposed model, we separate the tripartite graph G_{URT} into two bipartite subgraphs as User-Resource graph G_{UR} and Resource-Tag graph G_{RT} (shown in Fig. 1), to respectively address the two surfing processes. And then, in our method described further, we try to overcome the limitations of the state-of-the-art works by modeling resources search as two infinite random walks on these two graphs. The results of these two processes are finally integrated to reach the original purpose of our model, as exploiting rich semantics within G_{URT} as far as possible to recommend resources to users.

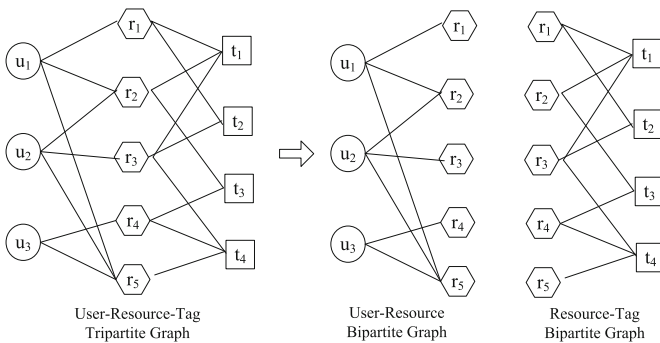


Fig. 1. An example of social tagging graph.

2.2 Relevance Propagation Based on Random Walk

We suppose that the walk in finding resources for a current user u is a non-stop process. That is, u visits the nodes in User-Resource graph or Resource-Tag graph over and over again. During this infinite walk (also, a discrete Markov process), the resources visited more often are considered more beneficial for u . However, the stationary distribution of such a random walk does not depend on the state of the initial probability distribution. To assure the existence of a stationary distribution, also retain the importance of a candidate resource to stay close to relevant tags or users, the jump transition need to be added to the graph nodes.

We first introduce the possibility to return regularly to the resource nodes from any node of the bipartite graph and to start the walk through mutual resource-user or tag-resource links again. The likelihood of jumping to the specific resource $P_J(r_j)$ (shown as Eq. 3) is considered to equal its normalized probability to be relevant to the current user u . This assumption makes candidates situated closer to u , and the more likely that the candidate is known to u , the more it can be selected for a random jump.

$$P_J(r_j) = \frac{R(u, r_j)}{\sum_{r_k \in C(u)} R(u, r_k)} \tag{3}$$

Then, the probability to jump to a user $P_J(u_i)$ (in G_{UR}) and a tag $P_J(t_i)$ (in G_{RT}) is added, respectively. We consider that the taste of a community member $u_j \in C(u)$ is more close to the current user, or a tag t_i is more popular in the community, it is visited more often by the current user during consecutive walk steps. So, we make $P_J(u_i)$ equal to the normalized similarity of u_i to u , and let $P_J(t_i)$ equal to the probability to find the tag t_i in the community. These two measures of jump transitions are shown as followings,

$$P_J(u_i) = \frac{R(u, u_i)}{\sum_{u_k \in C(u)} R(u, u_k)} P_J(t_i) = \frac{cf(t_i)}{\sum_{t_k \in C(u)} cf(t_k)} \tag{4}$$

where, $cf(t_i)$ is the occurrence frequency of the tag t_i in the community, $R(u, u_i)$ and $R(u, r_j)$ are same to the above-mentioned definition. For relevance propagation on User-Resource graph, the following HITS-like equations are used for iterations until convergence:

$$P^n(u_i) = dP_J(u_i) + (1 - d) \sum_{r_j} P(u_i|r_j)P^{n-1}(r_j) \tag{5}$$

$$P^n(r_j) = dP_J(r_j) + (1 - d) \sum_{u_i} P(r_j|u_i)P^{n-1}(u_i) \tag{6}$$

For relevance propagation on Resource-Tag graph, the following equations are used for iterations until convergence:

$$P^n(t_i) = dP_J(t_i) + (1 - d) \sum_{r_j} P(t_i|r_j)P^{n-1}(r_j) \tag{7}$$

$$P^n(r_j) = dP_J(r_j) + (1 - d) \sum_{t_i} P(r_j|t_i)P^{n-1}(t_i) \quad (8)$$

where d is the probability that at any step the user decides to make a jump and not to follow outgoing links anymore. According to our test, setting $d \in [0.1, 0.2]$ is a good choice for the most cases. The convergence condition of iteration is given by $|P^n(\cdot) - P^{n-1}(\cdot)| \leq \epsilon$. The described Markov process is aperiodic and irreducible, and hence has a stationary distribution. Consequently, we consider to integrate the two stationary probabilities $P_{UR}(r_j)$ and $P_{RT}(r_j)$ (shown as Eq. 9) as the final relevance of the resource r_j to the target user u .

$$R(u, r_j) = \lambda P_{UR}(r_j) + (1 - \lambda) P_{RT}(r_j) \quad (9)$$

Algorithm 1. *RPRW:Relevance Propagation with Random Walk*

Require. A target user u , three parameters k , d and λ , convergence threshold ϵ .

Ensure. A ranked list of resource set I .

for each $u_i \in U$ **do**

 estimate the relevance $R(u, u_i)$;

end for

retrieve the top- k similar neighbors of u ;

create $C(u)$, and poll all resources in $C(u)$ as the candidate set I ;

create bipartite graphs G_{UT} and G_{RT} based on $C(u)$ and I ;

for each $r_j \in I$ **do**

 estimate the relevance $R(u, r_j)$;

end for

normalize $R(u, u_i)$ and $R(u, r_j)$;

repeat

 update $P(u_i)$ and $P(r_j)$ with d on G_{UT} according to Eq. 5 and Eq. 6;

until converged

repeat

 update $P(t_i)$ and $P(r_j)$ with d on G_{RT} according to Eq. 7 and Eq. 8;

until converged

get the final relevance $R(u, r_j)$ between u and r_j , using Eq. 9;

return A ranked list of I ;

The whole process of our proposed method is explained as Algorithm 1. A main part of the algorithm is to calculate the relevance $R(u, u_i)$ and $R(u, r_j)$, however, such a computational overhead can be controlled in an acceptable range by pre-clustering users. Another main part of the algorithm is concerning the relevance propagation with random walk. In each iteration, the random walk probability is updated from the neighbor nodes of u , so the complexity of the algorithm is $O(f(k))$, where $f(k)$ marks the scale of nodes surrounding to u . Also, in experiments, the calculation converges fast (after 20–40 iterations) by assigning ideal setting to d and ϵ . Therefore, the whole time cost for each recommendation is acceptable in realtime scenario.

3 Experiments

3.1 Datasets

For experiments, we use the actual datasets collected from two well-known social media systems-Lastfm and MovieLens. Lastfm¹ is the world’s largest online music catalogue, and allows user tagging music tracks and artists. In this dataset, we take artists as resources. MovieLens² is a recommender system and virtual community website that recommends films for its users to watch, based on their film preferences and using collaborative filtering. The website is kept by the lab of GroupLens Research. The collaborative tagging function had been included in the website, thus researchers can gather tag-aware data for research purpose. For these three systems, we use their data collections released in the framework of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems [4] to make an evaluation. Statistics of datasets are listed in Table 1, and more detailed descriptions of these datasets can be found in [5].

Table 1. The basic statistics of the datasets.

Dataset	Users	Resources	Tags	Tas(UR)	Density(RT)	Density(u)	Training(u)	Test(u)
Lastfm	1,892	12,523	9,749	186,479	$3.0 * 10^{-3}$	$2.2 * 10^{-3}$	1,821	337
MovieLens	2,113	5,908	9,079	47,957	$9.0 * 10^{-4}$	$7.0 * 10^{-4}$	1,598	135

To test the algorithmic performance, the Lastfm dataset is divided into two parts according to the tag assignment(tas) timestamp: the training set contains 90% past entries and the remaining 10% future entries make up the testing set. Because test cases for the MovieLens dataset are relatively small, we separate this dataset by the ratio of 80%:20%. This policy follows the universal observation as known information used for recommending, while no information in the testing set is allowed to be used for recommending. Also, it meets the online operation principle of recommender systems, that is, the recommender periodically provides active users with resources of interest, at a certain point of time, using the historical data of the systems. Note that, since we do not concentrate on the cold-start problem in this paper, new users and new resources are eliminated from the testing dataset. The finally selected test cases are also presented in Table 1. Also, when generating the recommendation candidate list for a specified user, the resources already collected by the user are excluded from the list.

3.2 Evaluation Metrics and Baseline Methods

To give solid and comprehensive evaluation of the proposed algorithm, we employ three well-known metrics: Precision at top-K($P@K$), Recall at top-K($R@K$),

¹ <http://www.lastfm.com>

² <http://www.imdb.com>, <http://www.rottentomatoes.com>

and their harmonic mean-F1 metric at top-K($F1@K$), to characterize the accuracy of recommendations. In addition, Hamming Distance is selected to measure the diversity of recommendation. It examines the uniqueness of recommendation lists to separate users. Given two users i and j , the hamming distance between their recommendation lists can be calculated by Eq. 10.

$$HD_{ij}(k) = 1 - \frac{overlap_{ij}(k)}{k} \quad (10)$$

where $overlap_{ij}(k)$ is the number of shared items in the top- k places of the two recommendation lists. Averaging over all pairs of users, we can obtain the aggregate diversity of the system. Clearly, higher diversity means higher personalization of users' recommendation lists, $HD(k) = 1$ points to the fact that every user receives his/her own unique top- k items.

As far as the baseline method concerned, the approaches that recommend tags or use explicit ratings or other kinds of implicit information to make recommendations are not listed, considering that we focus on recommending items based on tag information.

ProbS [6]: similar to our work, a hybrid mass diffusion based algorithm using both User-Item graph and Item-Tag graph was proposed to fulfill personalized recommendation. Although mass diffusion can also work in multi-steps, we use the default two-steps diffusion in our experiments.

UserCF [7]: In this approach, the tag-based profiles were used to represent users' topic preferences as Eq. 1. The recommendation $rec(u, r_j)$ for a certain item r_j aggregates the votes of all neighbors of u using a similarity-weighting approach as Eq. 11,

$$rec(u, r_j) = \frac{\sum_{u_i \in C(u)} v_i(r_j) R(u, u_i)}{|C(u)|} \quad (11)$$

where, $v_i(r_j)$ is the normalized 'vote' of u_i to r_j . The neighborhood $C(u)$ for u and $R(u, u_i)$ are same as our above-mentioned definition.

Random Walk with Restart (RWR): RWR has recently attracted much attentions in various recommendation scenarios [8,9]. Here, we perform the RWR model on neighborhood-based tripartite graph, and set the personalized vector of the PageRank to bias the node representing the current user.

3.3 Experimental Results

We implemented our model and baseline methods using Java on a computer set to 4 GB memory and 3.1 GHz processors. We run tests extensively to find the optimal parameter settings for two datasets. The settings of the main parameters are shown in Table 2. Also, we set the convergence thresholds of iterative computation as a unified value $\epsilon < 0.001$, and perform experiments to investigate the computational efficiency of the proposed method (RPRW). Depending on Table 2, it takes only 20-40 iterations or several hundred milliseconds for our method to make a recommendation. This suggests our model can meet the

Table 2. The parameter settings for two datasets and the corresponding computational costs, where $|C(u)|$ is the community size to the user u , Iter4UR and Iter4RT are respectively the iteration times of RPRW on G_{UR} and G_{RT} .

Dataset	$ C(u) $	d	λ	Iter4UR	Iter4RT	TimeCost (ms)
Lastfm	20	0.15	[0.6,0.9]	19	33	150
MovieLens	20	0.15	[0.5,0.7]	21	39	30

demands of real-time application. In addition, we find that the best setting of λ in our model is consistent with the ProbS model, since they share the basic principle as combing the relevance score of r_j to u both on G_{UR} and G_{RT} .

We first compare our RPRW model with the baseline methods against the Lastfm dataset. According to Fig. 2, the ProbS performs best from the first position to the 15th position of recommendation list in all performance indicators, however, the RPRW model basically ranks in the first class after the 15th position in term of accuracy. In particular, the RPRW model outperforms all baseline methods in the F1 measure, where it improves the baseline methods by around 7% from F1@15 to F1@30. We next study the recommendation performance of selected methods based on the MovieLens dataset. Similar to the recommendation results achieved on the Lastfm dataset, our RPRW model does best after

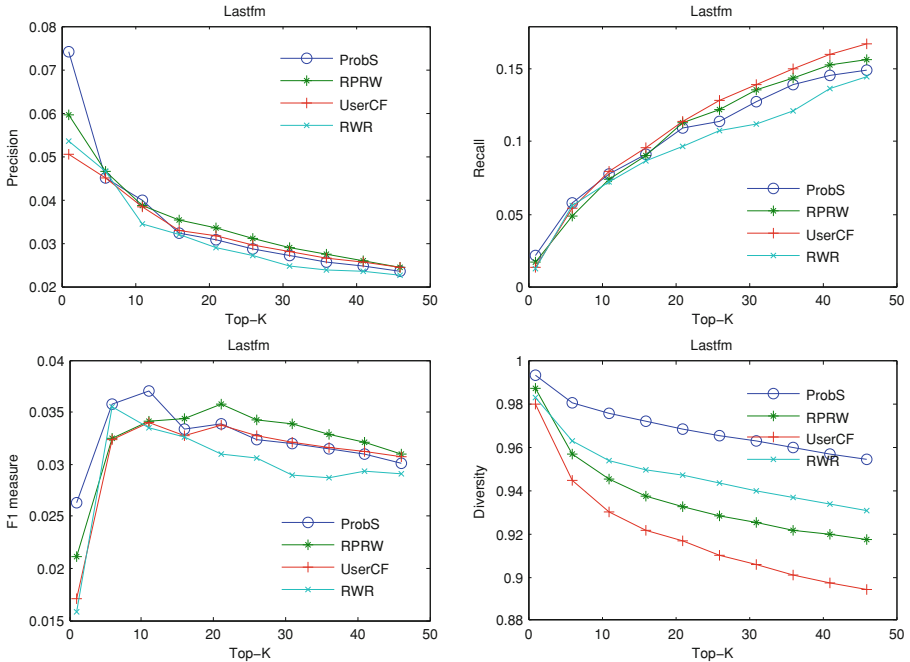


Fig. 2. Performance of recommendation based on the Lastfm dataset.

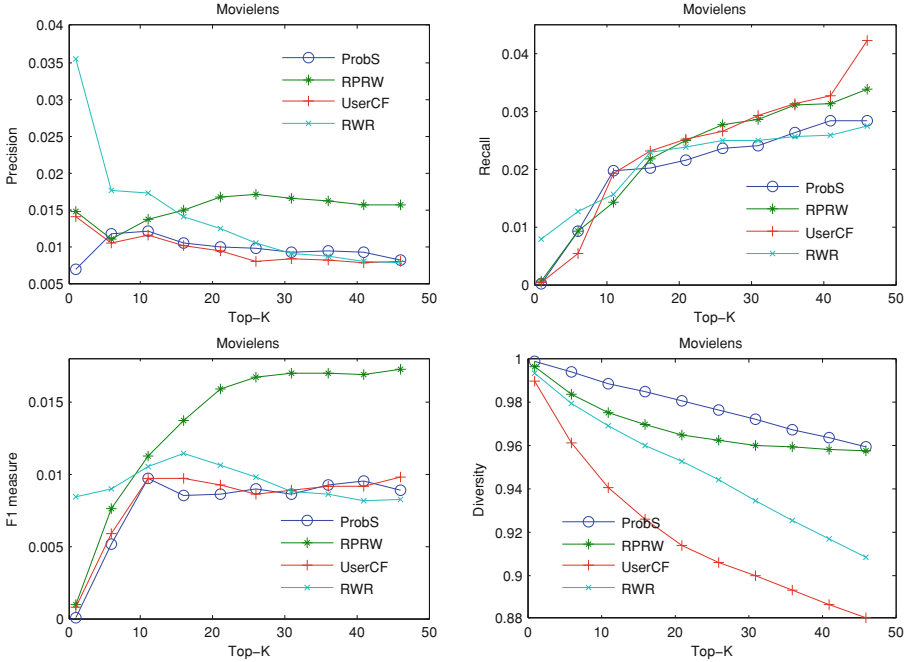


Fig. 3. Performance of recommendation based on the MovieLens dataset.

the 15th position in the recommendation list by accuracy metrics (see Fig. 3). Particularly, it significantly improves the F1 metric compared to all baseline methods. Different from the preceding experimental results where the ProbS achieves superior performance in P@10 and F1@10, instead, the RWR does best in P@10 and F1@10. By both Figs. 2 and 3, the UserCF performs best in the recall while slightly better than the RPRW model.

When examining the diversity of recommendation, the ProbS outperforms all other methods on both datasets. Both the RWR and the RPRW rank at the second place. A major reason caused this is that our model favors to popular items in the user community, i.e., the infinite random walk preferred to those nodes with higher degrees in social tagging graph. Such a situation is always observed in the classic random walk models, such as PageRank [10] and HITS [11]. Recommending commonly popular resources to users can obviously improve the accuracy while degrade the diversity. It is also a presentation of the well-known diversity-accuracy dilemma [9]. However, the RPRW model can still outperform some baseline methods by either the accuracy metric or the diversity metric, and achieve a better balance between the accuracy and the diversity of recommendation. This points to that the RPRW model has its own merits in recommendation situations. To further improve the diversification of recommendation results, the RPRW model can make use of a simple method to discount the popularity of resources [9].

4 Related Works and Discussion

There have been many technical advances in collaborative filtering models [3], topic-based models, and tensor-based models [6] for personalized recommendation. However, these models are distinctly different from our method, so we do not repeat them here. Instead, we concentrate some typical studies in applying random walks on personalized recommendation.

Hotho et al. proposed the FolkRank algorithm [12], an adaptation of the PageRank algorithm to the folksonomy structure. FolkRank performs a weight-spreading ranking scheme on folksonomies. It transforms the hypergraph between the sets of users, tags and resources into an undirected, weighted, tripartite graph. On this graph, it applies a version of PageRank that takes into account the obtained edge weights. Among applications, FolkRank provides a popularity measure of a document that seems to be better than PageRank, as it exploits the user produced folksonomy, rather than the Web links. FolkRank performs well in tag recommendation, however, it does not do well as other models on resource recommendation (for this, we omit the experimental results with respect to the FolkRank). ItemRank [13] proposed by Marco and Augusto, is used to rank products according to expected user preferences. It employs the naive PageRank on item-based graph to rank the item node. Then the PageRank and the preference to the expected user are integrated together to propose products. Similar to ItemRank, Yildirim and Krishnamoorthy [14] proposed a novel recommendation algorithm which performs random walks on a graph that stands for similarity measures between items. They evaluate their system using data from MovieLens. Although, the use of the random walk model performs well for recommendation, their use of an Item-Item similarity matrix raises some issues on the ability of the system to extend when other similarities are introduced based on social tagging. Konstas et al. [8] consider both the social annotation and the friendships inherent in the social graph established among users, items and tags. They adopt the generic framework of the RWR to provide with a more natural and efficient way to represent social networks. Their method is experimented with a self-collected Lastfm dataset and significantly outperforms the collaborative filtering method. However, their method utilizes all the training information to predict resources of interest, and seriously differs with our neighborhood-based training method. Hybrid ProbS [6] is recently introduced to item recommendation using tagging information. It applies respectively two mass diffusions in a user-resource and a resource-tag network to make recommendations. An item's preference is defined as a linear combination (similar to us) of its ranks in the two graphs. However, our approach can outperform this method by trustworthy experiments.

Regardless of the fact that these studies are close to our approach, we create a unique model, in which neighborhood-based method is first used to extract a dense social tagging subgraph, and then user-item preferences are propagated through infinite random walk. This strategy makes our model scalable even facing a huge amount of tagging data. Besides, our model has a better extendability:

- Advanced user/resource profiling methods (e.g. [15]) can be employed to strengthen relevance estimation;

- Explicit relations, such as friendships among users and inter-resources links, can be added to enrich the semantics;
- Tag-aware personalized search ([16]) can also be built on our model by making a neighborhood-based subgraph with an adhoc retrieval model.

5 Conclusion and Future Works

We have presented a relevance propagation model with random walk for a tag-aware personalized recommendation. According to solid experiments, our model performs effectively and efficiently in personalized recommendation, and achieves a better balance between accuracy and diversity metric. In future, we would consider developing advanced profiling methods to further strengthen relevance estimation in our model. Also, extending the model to cope with the cold start problem or tag-aware personalized search would also be interested.

Acknowledgments. This work is supported by the Applied Basic Research Project of Yunnan Province(2013FB009).

References

1. Golder, S.A., Huberman, B.A.: Usage patterns of collaborative tagging systems. *J. Inf. Sci.* **32**(2), 198–208 (2006)
2. Gupta, M., Li, R., Yin, Z., Han, J.: Survey on social tagging techniques. *ACM SIGKDD Explor.* **12**(1), 58–72 (2010)
3. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. *Adv. Artif. Intell.* **2009**(4), 1–1 (2009)
4. Cantador, I., Brusilovsky, P., Kufflik, T.: 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In: Proceedings of the 5th ACM Conference on Recommender Systems. RecSys 2011, New York, NY, USA, ACM (2011)
5. Bellogin, A., Cantador, I., Castells, P.: A comparative study of heterogeneous item recommendations in social systems. *Inf. Sci.* **221**(1), 142–169 (2013)
6. Zhang, Z.K., Zhou, T., Zhang, Y.C.: Tag-aware recommender systems: a state-of-the-art survey. *J. Comput. Sci. Technol.* **26**, 767–777 (2011)
7. Diederich, J., Iofciu, T.: Finding communities of practice from user profiles based on folksonomies. In: Tomadaki, E., Scott, P. (eds.) EC-TEL 2006 Workshops Proceedings Innovative Approaches for Learning and Knowledge Sharing, pp. 288–297 (2006)
8. Konstas, I., Stathopoulos, V., Jose, J.M.: On social networks and collaborative recommendation. In: Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 195–202. ACM (2009)
9. Wu, H., Cui, X., He, J., Li, B., Pei, Y.: On improving aggregate recommendation diversity and novelty in folksonomy-based social systems. Submission to Personal and Ubiquitous Computing (2014).
10. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.* **30**(1–7), 107–117 (1998)

11. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. *J. ACM* **46**(5), 604–632 (1999)
12. Hotho, A., Jäschke, R., Schmitz, Ch., Stumme, G.: Information retrieval in folksonomies: search and ranking. In: Sure, Y., Domingue, J. (eds.) *ESWC 2006*. LNCS, vol. 4011, pp. 411–426. Springer, Heidelberg (2006)
13. Gori, M., Pucci, A.: Itemrank: a random-walk based scoring algorithm for recommender engines. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 2766–2771 (2007)
14. Yildirim, H., Krishnamoorthy, M.S.: A random walk method for alleviating the sparsity problem in collaborative filtering. In: *Proceedings of the 2008 ACM Conference on Recommender Systems*, pp. 131–138. ACM (2008)
15. Liang, H., Xu, Y., Li, Y., Nayak, R.: Connecting users and items with weighted tags for personalized item recommendations. In: *Proceedings of 21st ACM Conference on HyperText and Hyper-Media*, Toronto, pp. 51–60. ACM (2010)
16. Xie, H.R., Li, Q., Cai, Y.: Community-aware resource profiling for personalized search in folksonomy. *J. Comput. Sci. Technol.* **27**(3), 599–610 (2012)

Optimizing Pipelined Execution for Distributed In-Memory OLAP System

Li Wang^(✉), Lei Zhang, Chengcheng Yu, and Aoying Zhou

Software Engineering Institute, East China Normal University, Shanghai, China
{wangli1426,zhangleicasa,yuchengcheng.ycc}@gmail.com,
ayzhou@sei.ecnu.edu.cn

Abstract. In the coming big data era, the demand for data analysis capability in real applications is growing at amazing pace. The memory's increasing capacity and decreasing price make it possible and attractive for the distributed OLAP system to load all the data into memory and thus significantly improve the data processing performance. In this paper, we model the performance of pipelined execution in distributed in-memory OLAP system and figure out that the data communication among the computation nodes, which is achieved by data exchange operator, is the performance bottleneck. Consequently, we explore the pipelined data exchange in depth and give a novel solution that is efficient, scalable, and skew-resilient. Experimental results show the effectiveness of our proposals by comparing with state-of-art techniques.

1 Introduction

The data volume that people collect is growing rapidly in the big data era. However, the real problem we are actually facing is not to collect and store the data with extremely large size as we have successfully done decades ago but to make valuable analytics on the huge amount of data within the time allowed [3]. Thanks to the development of hardware manufacturing, it is attractive for the distributed OLAP system to be equipped with large memory capacity (e.g., 100 GB per node) and benefit from efficient in-memory data processing.

The well known pipelined execution, which was proposed in Volcano [8], is very suitable for the distributed in-memory OLAP system, because it can effectively reduce the communication cost among operators, has good scalability, and avoids materializing intermediate result.

In this paper, we discuss the pipelined query execution in distributed in-memory OLAP system. More specifically, given a physical execution plan, our goal is to execute it as efficiently as possible. To do this, we introduce a general performance model for the pipelined execution on distributed in-memory OLAP system, which could figure out the performance bottleneck that lies on some part of the pipelined execution plan. The model also implies that in distributed in-memory OLAP system, the data exchange, which involves in network transmission, is very likely to be the bottleneck. Thus, we analyze the factors

accounting to the inefficiency of pipelined data exchange and propose a novel data exchange method that is efficient, scalable and skew-resilient. Finally, the experimental results show the effectiveness and efficiency of our proposals.

In the rest of the paper, Sect. 2 formally defines the pipelined data processing and shows the motivation of this paper. Section 3 gives theoretical analysis of the performance of pipelined data processing and proposes a general performance model. In Sect. 4, we discuss the challenges of pipelined data exchange and proposes an efficient, scalable, and skew-aware data exchange strategy. We evaluate our proposals in details and shows the results in Sect. 5. The related works are introduced in Sect. 6. Finally, we summarize in Sect. 7.

2 Preliminary

2.1 System Overall

This paper focus on the distributed in-memory OLAP system, which is running on a set of share-nothing servers (nodes) that are inter-connected through the network (e.g., 1 Gbps network router). The data is partitioned and distributed among these nodes and resides in the main-memory. The system accepts query in declare language (e.g., SQL), optimizes the received query into physical plan, and executes the plan in pipelined fashion, which will be described formally soon.

2.2 Pipelined Data Processing

Pipelined data processing was first proposed in Volcano [8]. In the pipelined execution, all the operators within a query are piped. Each operator consumes the data flow generated by its input operator(s) and produces processed data flow to its output operator(s) such that the data flow is continuously passing through the operators and the data processing are pipelined. The most promising advantage of such pipelined data processing is that the intermediate results produced by each operator is (if possible) immediately sent to the output operator(s).

A *Pipelined Execution plan* can be modeled as a direct acyclic graph $g = \langle \mathcal{O}, \mathcal{F} \rangle$, where $\mathcal{O} = \{O_1, O_2, \dots, O_n\}$ is a set of operators and $\mathcal{F} = \{f_1, f_2, \dots, f_m\} = \{\langle O_i, O_j \rangle \mid O_i, O_j \in \mathcal{O}\}$ is the data flows among these operators. The operators define how the data is processed and the data flows describe how the data is transmitted among these operators. Figure 1(a) is an example of pipelined execution with four operators, where O_c and O_d read data from the input source file and pass the data flow to the hash join operator O_b , which passes the join result to O_a .

2.3 Parallelism and Data Exchange

Parallelism can be easily achieved vertically and horizontally by running operators in different nodes. One possible parallelism of the pipelined execution plan shown in Fig. 1(a) is demonstrated in Fig. 1(b), where the operators running on a same parallel instance are in the same dark rectangle.

To transmit the data between the nodes in a desirable fashion, a control operator, namely *data exchange*, is introduced. An data exchange operator can be defined as $\mathbf{E} = \langle \mathcal{S}, \mathcal{R}, p(t) \rangle$, where $\mathcal{S} = \langle S_1, \dots, S_m \rangle$ is the list of senders for sending the data and $\mathcal{R} = \langle R_1, \dots, R_n \rangle$ is the list of receiver for receiving data. $P(t) : t \rightarrow R^{E'}$, $R^{E'} \subseteq R^E$ is a function defining the destination receiver(s) for each tuple t . In other words, $P(t)$ defines how the data are transmit (hash partition, range partition, or, broadcasting, etc.) . Each sender $s \in \mathcal{S}$ is running a node and sends the data flow to a subset of \mathcal{R} donated by $R(s) = \{R_1^s, \dots, R_m^s\}$ through network. Each receiver $r \in \mathcal{R}$ is running on different node and receives the data flow from a subset of \mathcal{S} donated by $S(r) = \{S_1^r, \dots, S_n^r\}$. All the senders and the receivers are logically viewed as an exchange operator.

In this paper, we explore the efficient execution for a given pipelined execution plan. For a given query, there may be a large number of possible execution plans, each of which results in different executing cost. There are extensive works on converting a query to the optimal execution plan, and it is out of the scope of this paper. We assume that the system has obtained the best execution plan and our goal is to execute the optimized execution plan as efficiently as possible.

3 Optimization Decomposition

3.1 Maximum Concurrently Executing Plan

There are two kinds of operator in a pipelined execution plan: *pipelined operators* and *synchronized operators*. An pipelined operator, such as *filter*, *predicate*, can immediately produces the output data flow to their successor(s) as soon as a piece of the input data flow arrives. In contrast, synchronized operators, such

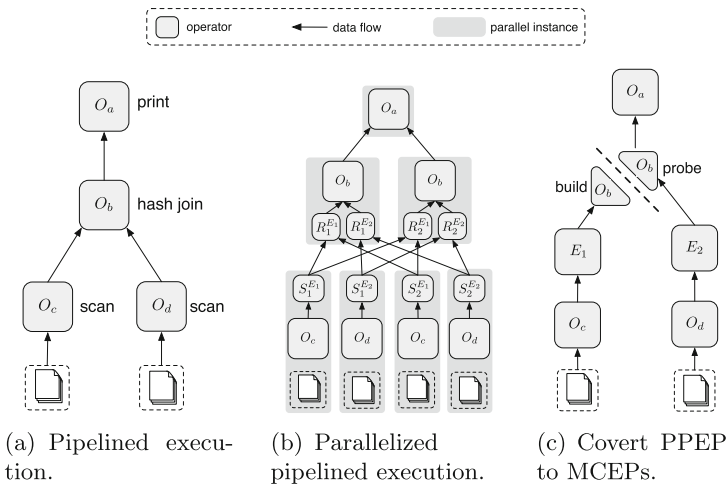


Fig. 1. The demonstration of pipelined execution.

as *hash join*, *aggregation*, *sort*, etc., refer to the operators which cannot produce any data to its successor(s) before the complete view of the input data flow.

Definition 1. An execution plan is a **Maximum Concurrently Executing Plan (MCEP)** if all the operators in the plan are pipelined operators.

Definition 2. A execution plan is a **Partially Pipelined Execution Plan (PPEP)** if it contains at least one synchronized operator.

In PPEP, the data flow will be blocked at the synchronized operators waiting for the complete view of the input data flow. Consequently, only part of operators in such case are processing the data at a given time.

Theorem 1. A PPEP can be converted into one or more MCEPs by cutting at the synchronized operators.

For instance, the execution plan in Fig.1(a) is divided into two MCEPs by cutting at hash join operator as shown in Fig.1(c).

3.2 MCEP Execution Optimization

Definition 3. For any MCEP \mathcal{M} where there are $k-1$ exchange operators E_1, E_2, \dots, E_{k-1} , \mathcal{M} can be divided into a list of stages denoted as $\text{stage}(\mathcal{M}) = \langle s_1, s_2, \dots, s_k \rangle$ by cutting any E_i between \mathcal{R} and \mathcal{S} .

All the operators within any stage s_i is running on the same node and the data flow is transmit through the efficient memory share. We denote $\theta_I(s_i)$ to be the throughput at which s_i receives data flow from its predecessor stage (through network) or input source file, denote $\theta_P(s_i)$ to be the throughput of all the operators in s_i , and denote $\theta_O(s_i)$ to be the throughput at which s_i could send the data flow to its successor. And let amplification rate, denoted as r_i , be the ratio of the data volume s_i produces when s_i consumes a piece of data flow of unit volume. For instance, if s_i could produce 1KB output data flow when consuming every 4KB, then $r_i = 0.25$. Amplification rate is very important in bridging $\theta_I(s_i)$ and $\theta_O(s_i)$. Now we have the following theorem.

Theorem 2. Let θ_M be the memory access throughput. For any MCEP \mathcal{M} , which has k stages s_1, \dots, s_k , the throughput at which \mathcal{M} produces data flow is $\theta(\mathcal{M}) = \min\{r_k\theta(s_{k-1}), \theta_P(s_k), \theta_M\}$, where

$$\theta(s_i) = \begin{cases} \min\{\theta_M, \theta_P(s_i), \theta_O(s_i)\} & \text{if } i = 1 \\ \min\{r_i\theta(s_{i-1}), \theta_P(s_i), \theta_O(s_i)\} & \text{if } 2 \leq i \leq k-1 \end{cases}$$

Proof. For s_k , it reads data from s_{k-1} , processes the data, and write the produced data into the memory buffer of the synchronized operator. As this process is pipelined and s_k could produce r_k unit data volume when consuming every 1 unit data volume, the throughput of data flow that s_k produces is limited by the minimum of $r_k\theta(s_{k-1})$, $\theta_P(s_k)$ and θ_M . For any s_i other than s_1 and s_k ,

the data flow it produces is transmit through the network at the throughput of $\theta_O(s_i)$, and hence the throughput of s_i is limited by the minimum of $r_i\theta(s_{i-1})$, $\theta_P(s_i)$, and $\theta_O(s_i)$. Finally, for s_1 , it consumes data flow from the input data source or the memory buffer of a synchronized operator, and consequently the throughput is limited by the minimum of θ_M , $\theta_P(s_i)$, and $\theta_O(s_i)$. \square

Theorem 2 implies that the performance bottleneck in a MCEP is either the memory access, or data processing, or the data exchange. The memory bandwidth and the in-memory data processing is several orders of magnitude faster than that of network. Besides, there are extensive in-memory database optimization techniques, such as the work in MonetDB/X100 [4], HANA [10], etc. The in-memory processing throughput can be dramatically improved by leveraging the multi-core data processing techniques, such as [2] for sort-merge join, [6] for aggregation, etc. Hence, in this paper, it is reasonable to assume that $\theta_P(s_i), \theta_M \gg \theta_O(s_i)$. Consequently, the efficiency of exchange operator is key to the throughput of a MCEP. In the next section, we will discuss the optimization to the exchange operator in depth.

4 Optimizing Data Exchange Operator

In this section, we first analyze the performance issues in pipelined exchange operator, then propose our novel method. For convenience, we hereby define the status of the senders and the receivers in an exchange operator.

Busy status. For any $r_i \in \mathcal{R}$ or $s_i \in \mathcal{S}$, it is under the busy status if the data transmission throughput reaches the capacity of currently available network bandwidth.

Starvation status. For any $r \in \mathcal{R}$, it is under the starvation problem if it cannot obtain data fast enough compared with the available network bandwidth.

Blocking status. For any $s \in \mathcal{S}$, it is under the blocking status if any receiving sub-operation $r \in R(s)$ is in starvation problem but s_i cannot sending any data to r due to some reasons.

Starvation status and block status mean that the network bandwidth is wasted. A typical case that introduces starvation and block status is that there are two receivers r_1 and r_2 and two sender s_1 and s_2 in an exchange operator. At a time, when s_1 is sending the data flow to r_1 at the maximum network bandwidth such that s_1 and r_1 are in busy status, s_2 will be in blocked status and r_2 will be in starvation status if the tuple that s_2 is sending is for r_1 .

4.1 Data Pushing Rather than Data Pulling

Data pulling is typically used in the materialized data exchange, such as the shuffling in Hadoop [11] and Spark [12]. However, data pulling fashion is impractical in the pipelined data exchange. Firstly, as the data sending is on the fly, the

receivers do not know whether the data is available to be fetched in their corresponding senders. As a result, the receivers have to blindly ask the senders for data and will fall in starvation status if the ask fails. Furthermore, on the sender side, whether to immediately send the data to the receiver which is asking for data is hard decided.

In the rest of this section, we will show how our novel data exchange extends data push fashion in terms of efficiency, scalability, and skew-resilience.

4.2 Sending Buffer Diversification Strategy

Definition 4. For any sender s and its buffer $B(s)$, the diversity of $B(s)$ is defined as $D(B(s)) = \frac{|R(B(s))|}{|R^s|}$, where R^s is the set of receivers of s , $B(s, R_i^s)$ is the set of blocks in $B(s)$ corresponding to R_i^s , and $R(B(S)) = \{r | r \in R^s, |B(s, r)| > 0\}$.

The range of diversity is $[0, 1]$ and is proportional to the size of $R(B(S))$. The larger $R(B(S))$ is, the more options we have to choose the receivers to send data and hence smaller probability of being blocked status. To keep the diversity, we propose a novel Sending Buffer Diversification (SBD) strategy which picks the blocks in the buffer according to the following probability:

$$p(R_i^s) = \frac{|B(s, R_i^s)|}{|B(s)|} \quad (1)$$

According to Eq. 1, the receivers corresponding to more blocks in the buffer have higher probability to be chosen as the receivers and vice versa. Consequently, Eq. 1 could not only keep the diversity of the sending buffer but also tend to keep a balanced number of blocks for each receiver.

4.3 Receiver Status Awareness

Now, we ameliorate SBD such that the information on the receiving sides (e.g., whether the receivers are busy or not) can be used. Being aware of the receiver situation is very important to avoid the blocked status and busy status. Consider that the buffer are skewed to a few receivers such that these receivers have much larger probability to be chosen as the sending target. If all of these receivers are busy, the sender will repeatedly try to send the data to these busy receivers and always fail, resulting in blocked status of the sender and the starvation status of other receivers. To solve this problem, we propose Sender Buffer Diversification with Weighted Penalty (SBD-WP) strategy as shown in Eq. 2, which adapts Eq. 1 by giving a penalty to the currently busy receivers.

$$p(R_i^s) = \alpha \frac{|B(s, R_i^s)|}{|B(s)|} - (1 - \alpha) \frac{fail(R_i^s, k)}{k} \quad (2)$$

In Eq. 2, $fail(R_i^s, k)$ denotes the number of sending failures related to R_i^s since the last k attempts. Compared with Eqs. 1 and 2 will reduce the probability

of a receiver for a while if the receiver fails to accept the data block from the senders, and the penalty is released after the next k attempts. We will show in the experimental section that this penalty has great contribution to the performance of data exchange, especially in the case of skewed data distribution.

4.4 Skew-Resilient Strategy

The imbalance workload of data exchange refers to that one or a few receivers (senders) have more data volume to be received (sent).

The motivation of our skew-aware strategy is as following. For a data exchange, the total execution time is decided by the sender/receiver which finished last. Hence, the total execution time can be effectively reduced if the senders/receivers which have large workload than others are assigned higher priority when sending/receiving.

Although such strategy sounds attractive and simple, it encounters with non-trivial challenges. For a pipelined MCEP, it is very difficult or impossible to predicate the accurate data volume and hence the exchange does not know how much data volume remains during the execution, especially for the MCEP containing deep stages. To avoid the expensive and error-prone prediction, we design an skew-resilient exchange strategy based on the watermark rather than the actual data volume.

When a MCEP is being executed, every operator maintains a *watermark*, which is a value between range from $[0, 1]$ indicating the ratio of the consumed data volume to the total data volume. For instance, if the watermark is 0.4, it means 40% of the input data flow has been consumed. For the operators at the bottom of an MCEP, they either consume data from the original data source file or from an buffer (e.g., hash table, etc.) in a synchronized operator. As the actual size of the data file or the buffer is known beforehand, the watermark is easy to obtain. Every time when the current watermark has increased by a given report threshold Δ (e.g., $\Delta = 0.02$), the operator will report the new watermark to its successor operator, which will continue to propagate the new watermark in the same fashion. For each receiver of the data exchange operator, it keeps track of the watermark received from each sender, and report the smallest among the received watermark to its successor. The value of Δ is the tradeoff between the report frequency and the accuracy.

The details of skew-resilient exchange strategy for any receiver R is shown in Algorithm 1. The watermark for sender s is denoted as $W(s)$. $S(R)^{ordered}$ is a list of senders which are currently trying to send data blocks to R, and is sorted on the watermark in increasing order. The watermark is transmit along with block. The algorithm gives higher priority to receive the data from the sender whose watermark is smaller than others. Every time when a block is received from a sender, it will check whether the block has the end-of-file (EOF) identifier. If it does, the algorithm will record that a sender is exhausted. Also, the algorithm will check whether the block containing a new watermark. If it does, the corresponding watermark will be updated and $S(R)^{ordered}$ will be

Algorithm 1. Skew-resilient Load balancing

```

1 exhausted_senders = 0;
2 while exhausted_senders < |S(r)| do
3   foreach s in S(R)ordered do
4     while there is any available block b in s do
5       read b from s;
6       if b is EOF then
7         remove s from S(R)ordered;
8         exhausted_senders ++;
9         GOTO 2;
10      if b contains new watermark then
11        update W(s);
12        if S(R)ordered is out of order then
13          reorder S(R)ordered;
14          GOTO 3;

```

reordered if it is out of order. The algorithm finishes when all the sending are exhausted.

5 Experiments

5.1 Setup

Our experiment is running on 12 nodes, each of which is a HP DL360 server with 192 GB RAM and two four-core CPUs. The nodes are connected by gigabyte switch. The operating system is Linux Redhat 6.3.

The evaluations are based on our distributed in-memory OLAP prototype system. We use both real dataset and synthetic dataset of TPC-H. The real dataset comes from transaction data from shanghai exchange. Without otherwise clarification, all the table are in column store and are resident in main memory. We use the following query¹.

```

SELECT T1.b, T2.c
FROM T1, T2
WHERE T1.a=T2.a AND T1.c>x

```

5.2 Pipelined Data Exchange

Buffer Size. The size of sending buffer is very important for the performance of data exchange, as it can effectively avoid the blocked status of the senders and

¹ Other queries are also evaluated in our experiment and result in similar results. Due to space limitation, these results are omitted.

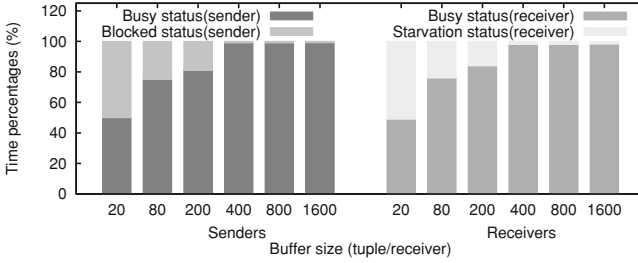


Fig. 2. The percentages of busy status to blocked/starvation status.

the starvation status of the receivers. To evaluate the benefit of sending buffer, we ran a 5-5 data exchange with different sending buffer size, and keep track of the throughput of the senders and the receivers during the execution. The execution time is divided into many short slices (e.g., 1 ms per slice). A time slice is in busy status, if the sending/receiving throughput is more than 90% of the theoretical maximal network bandwidth. Otherwise, a sender/receiver is in blocked/starvation status. The percentages of the time slices in busy status to the time slices in blocked/starvation status is shown in Fig. 2. When the sending buffer is small, the senders are very likely to be blocked as the buffer is easy to be full. When the buffer size increases, the percentage of blocked status is decreasing accordingly, resulting more time in busy status and more sending throughput. Similarly, the receivers suffer from starvation status a lot when the buffer size is small, as the sending in blocked status cannot produce new data to send. The benefit gain diminishes when the buffer size exceeds 400 tuples for each receiver, and hence we set the buffer size to be 400 tuples/receiver in the following experiments.

Sending Buffer Diversification and Weighted Penalty. To evaluate the effect of SBD strategy and SBD-WP strategy, we run 5-5 data exchange in the randomly sending strategy, SBD strategy, and SBD-WP strategy, respectively. The implementation of randomly sending strategy is according to the openly available codes of pipelined shuffling in [7]. To better evaluate the differences between these strategies, we generated the data in globally uniformly distributed but with local skewness. Such data can be viewed as a sequence of windows with equal size w (in number of tuples). The data within a window is in zipf ($\alpha = 0.5$) distribution, but the distribution of each window is independent and hence the data is roughly uniformly distributed in the global view. When $w = 1$, the data is in standard uniform distribution. The more w is, the more skewed that a piece of data will be, and consequently the more difficult it is for the senders to handle such local skewness. Figure 3 shows the sender throughput in the three strategies, when w varies. When $w = 0$, all the strategies could fully leverage network bandwidth and there is no obvious throughput difference among them. When w increases (more local skewness), the performance of randomly sending strategy is obviously reduced. That's because the randomly sending strategy fails

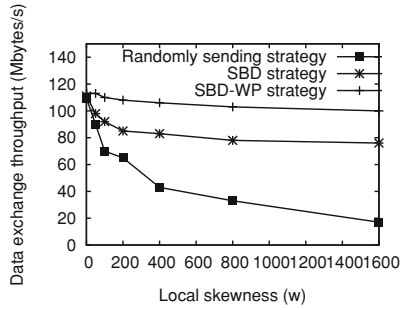


Fig. 3. The performance of the data exchange when local skewness varies.

to reserve the diversity of the sending buffer, which will make the sending more likely to suffer from the blocked status. In contrast, benefit from keeping the diversity of the sending buffer, SBD strategy is much better than the randomly sending strategy. By reducing the probability to send the data to the recently busy receiver, SBD-WP strategy can further avoid the time wasted on repeatedly sending data to the receiver that will reject the sending attempt immediately, and the performance of SBD-WP almost retains the same with more local skewness.

Skew-Resilience. To evaluate the performance of skew-resilient strategy, we generated 12 GB data in uniform distributed and partitioned the data among the nodes of the senders in an imbalanced way such that the data volume for a few senders is much more than others. Figure 4 shows how SBD-WP strategy without skew-awareness performs. Figure 5(a) demonstrates that at the beginning, each sender works at the same network bandwidth and the remaining data volume at each sender is degrading at the same pace. After 20s, all the sends except S1 and S2 finished the work, and hence S1 and S2 occupied all the sending network bandwidth.

As shown in Fig. 4, at the beginning phase, the remaining data volume at each sender is degrading at the same speed, because without skew-awareness

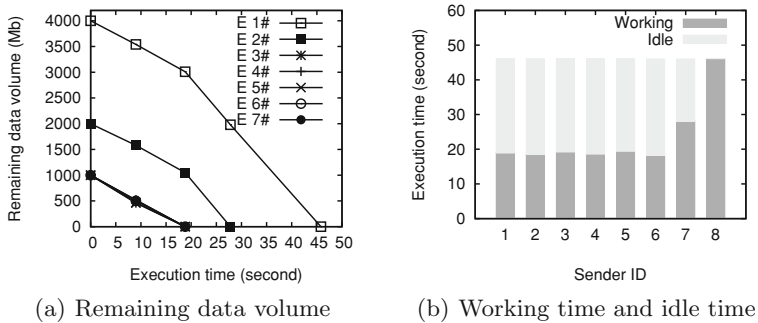


Fig. 4. Data exchange without skew-resilience.

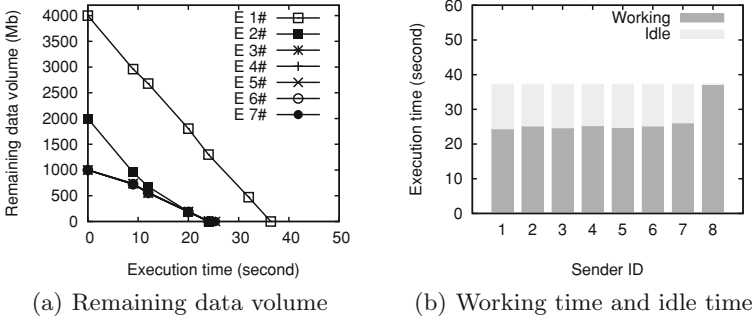


Fig. 5. Data exchange with skew-resilience

the aggregated network bandwidth of the receivers are equally divided into each sender. After 20s, all the senders except S1 and S2 finished the work, and hence S1 and S2 work at their maximal network bandwidth. S1 and S2 finished their work at 28s and 45s, respectively. Figure 5 shows the performance of skew-resilient strategy. Benefit from the watermark, each receiver could know the progress of each sender and is able to allocate more bandwidth to the senders that fall behind in progress. Consequently, S1 and S2, which have much data volume to send than other senders, have higher priority to occupy the network bandwidth. Thus, the total execution time is significantly reduced.

5.3 Overall Query Evaluation

In this subsection, we show the contributions of our data exchange method to the overall query performance. To do this, we compare the query performance running on our execution engine with naive data exchange implementation and our exchange method, respectively. The query is running on the synthetic datasets following in Zipf distribution with various skewness factor. The results are shown in Fig. 6. The performance of execution with naive exchange implementation is much lower compared with ours, as the naive exchange cannot fully leverage the network resource and sending/receiving information. Further, due to the absence of skewness handling capability, the performance reduces significantly with increasing the data skewness. In contrast, the execution engine adopting our data exchange method is much efficient. Moreover, the performance does not reduce greatly in case of data skewness. In other perspective, the results confirm our claims in Sect. 3 that the network transmission is very likely to be the performance bottleneck and efficient data exchange method is key to improve query performance.

6 Related Work

Volcano [8] is an early parallel execution engine, which follows the well-known pipelined execution that has good scalability and is widely employed by the

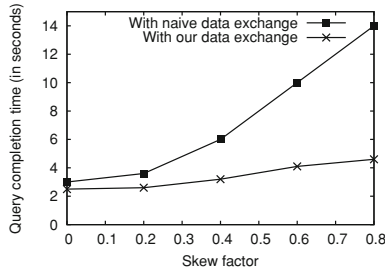


Fig. 6. Overall query time with various skewness factor.

following DBMS system. MonetDB/X100 [4] improve pipelined execution for in-memory data by modifying the pipelined fashion from tuple-at-a-time to block-at-a-time to facilitate the compiler’s automatic optimization, vector-wise computation, and column-wise data compression. However, MonetDB/X100 is a centralized database. Condie et al. [7] tries to pipeline the data shuffling in Hadoop [5] to improve the response time of the MapReduce job. However, our work is different from [7] in that our work focuses on the improving the query throughput by identifying and solving the performance bottleneck rather than applying the pipelined fashion to the MapReduce framework. Kumar et al. [9] tries to maximize the network usage for the data exchange of real applications by configuring the network parameters and scheduling the data exchange. However, it fails to handle the data skewness and leverage the information on the sending sides and the receiver sides.

There are extensive work in in-memory databases. When the data is shifted into main memory, the disk I/O bottleneck is removed accordingly and new bottleneck is the relatively slow memory access speed compared with fast CPU processing capability, which is called *the memory wall*. Boncz et al. propose radix partition algorithm to break the memory in hash partition. The light weight compression has been explored in [1, 10], which can increase the effective bandwidth by transmitting the compressed data.

7 Summary

In this paper, we discussed the pipelined execution in the distributed in-memory OLAP system. We first give a general performance model which could help to identify the bottleneck for a given pipelined execution plan. The model also implies that the efficiency of data exchange is key to the query performance. Thus, we discuss challenges of pipelined data exchange, and propose our novel data exchange operator. Finally, massive experiments prove that pipelined data exchange could dramatically improve the query throughput in the in-memory setting, and shows the efficiency and effectiveness of our proposals.

References

1. Abadi, D., Madden, S., Ferreira, M.: Integrating compression and execution in column-oriented database systems. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp. 671–682. ACM (2006)
2. Albutiu, M.-C., Kemper, A., Neumann, T.: Massively parallel sort-merge joins in main memory multi-core database systems. *Proc. VLDB Endow.* **5**(10), 1064–1075 (2012)
3. Barlow, M.: *Real-Time Big Data Analytics: Emerging Architecture*. O’Reilly Media Inc., Sebastopol (2013)
4. Boncz, P.A., Zukowski, M., Nes, N.: Monetdb/x100: hyper-pipelining query execution. *CIDR* **5**, 225–237 (2005)
5. Borthakur, D.: *The hadoop distributed file system: architecture and design* (2007)
6. Cieslewicz, J., Ross, K.A.: Adaptive aggregation on chip multiprocessors. In: Proceedings of the 33rd International Conference on Very Large Data Bases, pp. 339–350. VLDB Endowment (2007)
7. Condie, T., Conway, N., Alvaro, P., Hellerstein, J.M., Elmeleegy, K., Sears, R.: Mapreduce online. In: NSDI, vol. 10, pp. 20 (2010)
8. Graefe, G.: Volcano-an extensible and parallel query evaluation system. *IEEE Trans. Knowl. Data Eng.* **6**(1), 120–135 (1994)
9. Kumar, V.S., Tucek, J., Wylie, J.J., Krevat, E., Ganger, G.R.: Application-level flow scheduling for efficient collective data transfers (2012)
10. Plattner, H.: A common database approach for oltp and olap using an in-memory column database. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pp. 1–2. ACM (2009)
11. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Proceedings of the Mass Storage Systems and Technologies (MSST), pp. 1–10. IEEE (2010)
12. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M., Shenker, S., Stoica, I.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the NSDI, p. 2. USENIX Association (2012)

Hash^{ed}-Join: Approximate String Similarity Join with Hashing

Peisen Yuan¹(✉), Chaofeng Sha², and Yi Sun²

¹ College of Information Science and Technology, Nanjing Agricultural University, Nanjing 210095, China

² School of Computer Science, Fudan University, Shanghai 200433, China
{peiseny, cfsha, ysun}@fudan.edu.cn

Abstract. The string similarity join, which finds similar string pairs from string sets, has received extensive attention in database and information retrieval fields. To this problem, the *filter-and-refine* framework is usually adopted by the existing research work, and various filtering methods have been proposed. Recently, tree based index techniques with the edit distance constraint are effectively employed for evaluating the string similarity join. However, they do not scale well with large distance threshold. In this paper, we propose an approach for approximate string similarity join based on Min-Hashing locality sensitive hashing and *trie*-based index techniques. Our approach is flexible between trading the efficiency and performance. Empirical study using the real datasets demonstrates that our framework is more efficient and scales better.

1 Introduction

String is one of the most important data types in modern data processing systems. One of the important researches on string is the similarity join, i.e., finding all the similarity string pairs from the two string sets, which is a key operation in many real-world applications, such as data integration [1], data cleaning [2], duplicate detection [3] and so on. The similarity join has been received extensive attention from the database and information retrieval fields and there are extensive literatures for addressing this problem [3–5].

For measuring the string similarity, the string edit distance is used by most previous studies. However, the time and space complexity of evaluating string edit distance between two strings s_1 and s_2 is $O(|s_1||s_2|)$ [6], where $|s|$ denoted the length of the string. Therefore, most existing work mainly focus on the *filter-and-refine* framework. According to this framework, strings are firstly filtered by some filtering techniques in a heuristic way, and the edit distance evaluation is applied on remaining candidate string set subsequently.

In term of string similarity join with edit distance constraint, recent researches [4, 7] explore the tree index techniques. J. Wang et al. propose the Trie-Join [4] for string similarity join using a trie tree, which processes string similarity join efficiently and the space cost of the trie indexing structure is much smaller

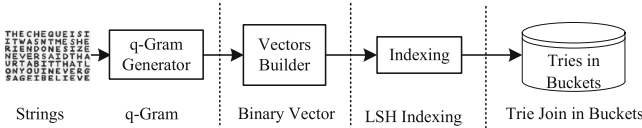


Fig. 1. Processing procedure

than existing work. However, the performance of the Trie-Join degrades for long strings and when the string edit distance threshold increases.

In this paper, we take the locality sensitive hashing (LSH for short) [8] and trie join techniques into account and propose a framework **Hash^{ed}-Join** for approximate string similarity join under edit distance constraint. The processing procedure is shown in Fig. 1, which can be divided into four steps.

In Fig. 1, the q -grams of each string in the string set are extracted in the first place. Then the binary vectors of the strings are built based on their corresponding q -grams set. In the third step, the Min-Hashing based LSH technique is employed, which hashes the similar strings into the same bucket, i.e., the string sharing q -grams. After that, a trie structure index is built in each collision bucket and the trie join technique is applied. The similar string pairs in each bucket are merged together and the final result for the string sets is returned finally.

To summarize, the main contributions of this paper are briefly outlined as follows:

1. The framework using hashing for the approximate string similarity join problem is proposed.
2. Relationship of approximate similarity join with the parameters and the string features are studied.
3. Extensive experiments on real datasets are conducted to demonstrate the effectiveness and efficiency of our approach.

2 Preliminaries

2.1 Problem Statement

Given two strings s_1 and s_2 and a proposition formula \mathcal{F} , which is defined in the form of $sim(s_1, s_2) \geq \theta$, where $sim(s_1, s_2)$ is the similarity metric for strings s_1 and s_2 . The task of string similarity join is retrieving the similar string pairs between two string sets that satisfying the proposition formula \mathcal{F} . The formal definition is presented as follows.

Definition 1. *String Similarity Join*

Given two string sets S_1, S_2 and proposition formula \mathcal{F} , the similarity join between S_1 and S_2 is denoted as $S_1 \bowtie_{\mathcal{F}} S_2$. The result of the join is denoted as $S_1 \bowtie_{\mathcal{F}} S_2 = \{ \langle s_1, s_2 \rangle \mid s_1 \in S_1 \text{ and } s_2 \in S_2, \mathcal{F}(\langle s_1, s_2 \rangle) \text{ is true} \}$, where $\langle s_1, s_2 \rangle$ is the similar pair that satisfying the proposition formula \mathcal{F} .

For example, given two string sets $S_1 = \{\text{"microsoft"}, \text{"applies"}, \text{"informix"}, \text{"tree"}\}$ and $S_2 = \{\text{"apple"}, \text{"infromix"}, \text{"google"}, \text{"trie"}, \text{"mcrosoft"}\}$, and the proposition formula \mathcal{F} is defined as the string edit distance defined in Sect. 2.3, $\text{dist}_{ed}(s_1, s_2) \leq 1$. The similarity join result on the two string sets is $S_1 \bowtie_{\mathcal{F}} S_2 = \{\langle \text{"microsoft"}, \text{"mcrosoft"} \rangle, \langle \text{"trie"}, \text{"tree"} \rangle\}$.

2.2 q -gram

Given a string s and an integer q , its q -grams can be obtained by a sliding window on s with length q , contiguously splitting the string into a group of substrings. For a given string s , its q -gram may occur multiple times, we treat the duplicate q -grams as new ones by inserting an integer representing the occurrence order [3]. In this way, a string s can generate $L^* = |s| - q + 1$ q -grams.

Example 1. Consider the string $s = \text{"mathematics"}$, let $q = 2$, the set of 2-grams of s , $\Phi_2(s) = \{\text{"ma"}, \text{"at"}, \text{"th"}, \text{"he"}, \text{"em"}, \text{"ma2"}, \text{"at2"}, \text{"ti"}, \text{"ic"}, \text{"cs"}\}$. The q -gram such as "ma" reoccurs later is appended with the order of the occurrence number to differentiate it. The length of $\Phi_2(s)$, i.e., $|\Phi_2(s)|$ is 10.

Given a string set S , for each string $s \in S$, we extract its q -grams and denoted as $\Phi_q(s)$. Let \mathcal{U} be the universal of the q -grams of S , i.e., $\mathcal{U} = \bigcup_{i=1}^{|S|} \Phi_q(s_i) = \{g_1, \dots, g_{|\mathcal{U}|}\}$, where $i = 1, \dots, |S|$. Then for a string s , it can be represented by a binary vector \mathbf{v}_b^s with the vector length $|\mathcal{U}|$, where $\mathbf{v}_b^s[j] = 1$, if $g_j \in \Phi_q(s)$; otherwise $\mathbf{v}_b^s[j] = 0$, for $j = 1, \dots, |\mathcal{U}|$. In this paper, the string is taken as the binary vector of its q -gram and they are used interchangeably if not confused.

2.3 Similarity Metrics

There are many similarity metrics that can be used for measuring string similarity, such as string edit distance, Jaccard similarity etc. In this paper, string edit distance and Jaccard similarity are used and introduced in the following section.

Jaccard Similarity. Given two sets A and B , their Jaccard similarity is defined as

$$\text{sim}_{jacc}(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

In many applications, the Jaccard distance is usually used as the distance measure for set similarity, which is defined as Eq. 2.

$$\text{dist}_{jacc}(A, B) = 1 - \text{sim}_{jacc}(A, B). \quad (2)$$

String Edit Distance. The string edit distance is also called as the *Levenshtein distance* [9], which is a metric used for measuring the differences between

two strings. The evaluation of string edit distance is based on three primitive operations: insertion, deletion and substitution, which are denoted as op_i , op_d and op_s . The definition is defined as follows.

Definition 2. *String Edit Distance*

Given two strings s_1 and s_2 , the edit distance between s_1 and s_2 is defined as the minimum number of the three primitive operations needed to transform s_1 to s_2 , which is denoted as $dist_{ed}(s_1, s_2) = \min \sum (c_i * op_i + c_d * op_d + c_s * op_s)$, where c_i, c_d and c_s are the cost of the operation respectively.

In this paper, the cost of each operation is set to 1. Considering two strings s_i and s_j , their q -gram sets are denoted as $\Phi_q(s_i)$ and $\Phi_q(s_j)$ respectively. If $dist_{ed}(s_i, s_j) \leq \tau$, then the common q -grams that s_i and s_j share should satisfy the formula $|\Phi_q(s_i) \cap \Phi_q(s_j)| \geq \max\{|s_i|, |s_j|\} - q + 1 - q \cdot \tau$ [10]. This property often used for filtering non-similar strings under the edit distance constraint.

2.4 Min-Hashing

The Min-Hashing is used for approximately set similarity evaluation [11]. It has the property that the probability of two sets have the same value of Min-Hashing is equal to their Jaccard similarity, and the formal definition is given as follows.

Given a random hash function $h : \mathcal{S} \rightarrow I$, where \mathcal{S} is the domain of the string set, and I is an integer set. For a string $s \in \mathcal{S}$, which is represented with the binary vector \mathbf{v} , the Min-Hashing function is defined as $m_h(\mathbf{v}) = \arg \min\{h(\mathbf{v}_i) \mid \mathbf{v} \in \mathbf{V}_b, \mathbf{V}_b \text{ is binary vectors for all the strings, } \mathbf{v}_i \text{ is the } i\text{-th index of } \mathbf{v} \text{ if } \mathbf{v}[i] = 1, \text{ for } 0 \leq i \leq |\mathbf{v}| - 1\}$.

According to the property of Min-Hashing, for two strings s_1 and s_2 and their q -gram sets $\Phi_q(s_1)$ and $\Phi_q(s_2)$, the binary vectors of them are represented by $\mathbf{v}_b^{s_1}$ and $\mathbf{v}_b^{s_2}$ respectively. Their Jaccard similarity can be approximately computed by Eq. 3.

$$sim_{jacc}(\Phi_q(s_1), \Phi_q(s_2)) = \Pr[m_h(\mathbf{v}_b^{s_1}) = m_h(\mathbf{v}_b^{s_2})]. \quad (3)$$

To reduce the probability of false positive retrieval, a random hash family $\mathcal{H} : \mathcal{D} \rightarrow I$ is usually used. Given a hash family \mathcal{H} , n Min-Hashing signatures are computed for each string. Thus the binary vector \mathbf{v}_b^s of the string s is transformed into $g(\mathbf{v}_b^s)$ and represented as Eq. 4.

$$g(\mathbf{v}_b^s) = \langle m_{h_1}(\mathbf{v}_b^s), m_{h_2}(\mathbf{v}_b^s), \dots, m_{h_n}(\mathbf{v}_b^s) \rangle, m_{h_i} \in \mathcal{H}, \quad (4)$$

for $i = 1, \dots, n$.

2.5 Locality Sensitive Hashing

The concept of locality sensitive hashing (LSH) is introduced in [8] and widely used for approximate nearest neighbor search of high dimensional data etc., the definition can be formalized as follow.

Definition 3. Let \mathcal{O} be the domain of the objects, $o_1, o_2 \in \mathcal{O}$, and $d_1 < d_2$ be two distances according to the distance metric $\text{dist}(o_1, o_2)$. A function family \mathcal{H} is said to be (d_1, d_2, p_1, p_2) -sensitive, if each $h \in \mathcal{H}$ satisfies the following two conditions:

- If $\text{dist}(o_1, o_2) \leq d_1$, then $\Pr_{\mathcal{H}}[h(o_1) = h(o_2)] \geq p_1$;
- If $\text{dist}(o_1, o_2) \geq d_2$, then $\Pr_{\mathcal{H}}[h(o_1) = h(o_2)] \leq p_2$, where $p_1 > p_2 \in [0, 1]$.

The LSH index is a data structure using a family of LSH functions \mathcal{H} , which is constructed in the following two steps [12]:

1. Given an integer r , define a function family $\mathcal{G} = \{g : \mathcal{O} \rightarrow I^r\}$, and for $g \in \mathcal{G}$, $g(o) = \langle h_1(o), \dots, h_r(o) \rangle$, where $h_i \in \mathcal{H}$ for $1 \leq i \leq r$.
2. For an integer b , randomly choose g_1, \dots, g_b from \mathcal{G} . Construct a hash table for each g_i , for $1 \leq i \leq b$.

In order to construct a Min-Hashing based LSH index, the *signature* matrix is divided into b bands with r Min-Hashing signatures in each band, i.e., $n = b * r$. If Jaccard distance is used as the distance metric defined in Eq. 2, then Min-Hashing LSH is $(d_1, d_2, 1 - (1 - p_1^r)^b, 1 - (1 - p_2^r)^b)$ -sensitive [13].

The parameter r control the filtering effectiveness and b controls the approximation factor. Given the Jaccard similarity of two objects is θ , the probability that they can be retrieved with Min-hashing LSH is equal to Eq. 5.

$$p = 1 - (1 - \theta^r)^b. \tag{5}$$

3 Trie Join and Observations

3.1 Trie Join

Trie Join [4] using a trie tree for string join, which is a tree structure and used for indexing the strings. Figure 3 demonstrates a trie tree of the running example

String ID	Sample Strings
s_1	<i>apple</i>
s_2	<i>applies</i>
s_3	<i>good</i>
s_4	<i>google</i>
s_5	<i>tree</i>
s_6	<i>trie</i>

Fig. 2. Strings of the running example

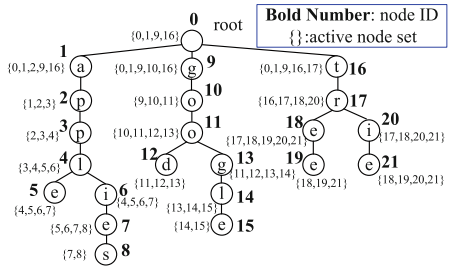


Fig. 3. Trie with $\tau = 1$ on running string set

strings in Fig. 2. The numbers in the tree indicate the node ID. For example, node 11 in Fig. 3 denotes substring “*goo*”.

Given a string s , node n in the trie tree is called an *active node* of s if $dist_{ed}(s, n) \leq \tau$ [4]. The active node set of s is denoted as $A(s)$ and $n \in A(s)$ if n is an active node of s . In Fig. 3, the node set enclosed by the $\{\}$ is the active node set of the corresponding node nearby with the string edit distance threshold $\tau = 1$. For example, the trie node 10 represents the substring “*go*” and the active node set of which is $\{9, 10, 11\}$. The active nodes 9, 10, 11 represent the substrings “*g*”, “*go*” and “*goo*” respectively. Their string edit distance with “*go*” is not bigger than 1.

To evaluate the active node set for each node in the trie, the active node of the root is computed firstly. Then for each internal node in the trie, its active node set can be evaluated using its parent’s one.

After obtaining the active node set for each node in the trie, the similar pairs can be evaluated with the active node sets. For each leaf node n_l in the trie, the active node $a' \in A(n_l)$ is verified whether a' is a leaf. If a' is a leaf node, then $\langle n_l, a' \rangle$ is a similar pair.

3.2 Problems of Trie Join

According to the evaluation procedure of the Trie-Join [4], we observe that dividing the string set into groups can reduce the computation overhead, especially with the increasing of the edit distance threshold τ . This can be illustrated by Fig. 4. According to Trie-Join, the trie nodes in the trie tree with the length τ in the first branch will be the active nodes of the other branches with the length no large than τ , and the nodes in other branches will be the active nodes of other branches as well. However, lots of the nodes in the active node set do not contribute to the finally result, thus it wastes much computation.

Take Fig. 3 as an example, the active nodes of the trie branches in Fig. 3 are shown in Fig. 5. The total active node number of Fig. 3 is 82, nevertheless, the total active node number of the three branches is $\sum(29, 24, 25) = 78$, which is less than the former. When τ is 2, the active nodes of the three branches is $\sum(45, 38, 37) = 120$, less than the total number of the trie tree 142 in Fig. 3,

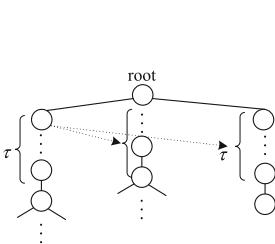


Fig. 4. Illustration of Trie-Join with edit distance

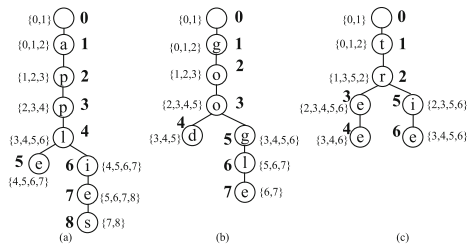


Fig. 5. Active node of the Trie tree branches with $\tau = 1$

which indicates that dividing the similar string set into groups previously can reduce the size of the active node set.

4 Processing Algorithms

4.1 String Binary Vector Building

In this algorithm, the q -grams of each string are extracted and the binary vector for each string is constructed.

Algorithm 1. String Binary Vectors Building

Input: String Set S ; Integer q .
Output: Compressed Binary Vectors.

```

1 Vector  $V = \emptyset$ ;
2 foreach  $s \in S$  do
3    $\Phi_q(s) = qgramGen(s)$ ;
4  $\mathcal{U} = \cup_{i=1}^{|S|} \Phi_q(s_i)$ ;
5 foreach  $\Phi_q(s)$  do
6   Vector  $v = \text{new Vector}()$ ;
7   foreach  $g \in \mathcal{U}$  do
8     if  $g \in \Phi_q(s)$  then
9        $v.add(1)$ ;
10    else
11       $v.add(0)$ ;
12    $v_c = \text{VectorCompressor}(v)$ ;
13    $V = \cup v_c$ ;
14 return  $V$ ;

```

In Algorithm 1, the q -grams of each string in the string set are extracted firstly (lines 2–3). After generating the q -grams, the binary vector for string s is generated based on the q -gram universal \mathcal{U} of the string set and the q -gram set $\Phi_q(s)$ of string s (lines 4–11).

Due to the high dimensionality of the binary vectors, for reducing the storage overhead, binary vector is divided with length L , where $L \leq 64$ and $L \leq |\mathcal{U}|$. In each group, the binary vector is transformed into an integer (line 12). Finally, the compressed binary vectors are returned (line 14).

4.2 Approximate Similarity Join Algorithm

The approximate join algorithm is outlined in Algorithm 2. The input of the algorithm includes compressed binary vectors, string set list SL and edit distance threshold τ . The output of the algorithm is the similar string pairs. In this paper,

Algorithm 2. Approximate Similarity Join Algorithm

Input: Compressed String Binary Vectors \mathbf{V} ; String Set List SL ; Edit Distance Threshold τ .

Output: Similar String Pairs within Threshold τ .

```

1 MinHashClass minHash = new MinHashClass();
2  $\mathbf{V}' = decompress(\mathbf{V})$ ; /* Decompress the binary vectors */;
3  $LSHIndex = minHash(\mathbf{V}')$ ; /*Generate the Min-Hashing LSH index for the
   string set */;
4 foreach band  $b \in LSHIndex$  do
5    $bandNo = getBandNo(b)$ ;
6   foreach bucket  $buc \in b$  do
7      $bucketNo = getBucketNo(bandNo)$ ;
8      $S' = getString(bandNo \oplus bucketNo, SL)$ ; /*Get the subset  $S'$  of the
       string set  $S$  from the collision bucket */;
9     if ( $|S'| == 1$ ) then
10       $sp = null$ ; /*similarity pair */;
11     else
12       $trie = buildTrieTree(S')$ ; /*Build a trie tree index for  $S'$  */;
13       $sp = trie.GenerateSimiPair(trie, \tau)$ ;
14       $SP = MergeSimilarPairs(sp)$ ; /*Similarity Pair Set */;
15 return  $SP$ ;

```

we only take self-join into account, i.e., $S \bowtie_{\mathcal{F}} S$. The join between two different string sets can be easily extended.

In this algorithm, the Min-Hashing values of each string are divided into b bands, each with r hash values. Within each band, the r Min-hashing values of a string are concatenated as the key, i.e., for string s , $key(s) = m_{h_1}(s) \oplus \dots \oplus m_{h_r}(s)$; and the string is hashed to one of the M different buckets based on the concatenated key. After hashing, a trie tree is built for the strings in the same bucket.

The algorithm can be divided into four steps. First, the Min-Hashing based LSH index is constructed for the string set (line 3), which decompresses the binary vectors from the integers.

Secondly, strings in the same bucket are obtained with the band number and the bucket number of the LSH index (lines 4–7). If there is only one string in the bucket, then this bucket can be skipped for it cannot make up pairs (lines 8–10). One key problem in this step is that the strings within the string edit distance threshold τ can be fall into the same bucket in at least one band with high probability by tuning the proper parameter of b and r .

Thirdly, A trie structure is built for the strings in each bucket in the third step (line 12), and the similar string pairs within the threshold τ are evaluated on the strings falling into the bucket by the Trie-Traversal algorithms proposed in [4] (line 13).

Table 1. Statistics of the sampled datasets

Dataset	$ S $	Max len	Min len	Avg len
DBLP Title	5885	163	3	58.14
DBLP TA	7040	284	8	81.89
UniProt	2578	1882	5	367.70

Finally, the similarity pairs in the buckets of each band are merged together and returned (lines 14–15).

5 Experiments

5.1 Experiment Setup

Algorithms are implemented in Java SDK1.6, and the computer is configured with Intel duo core E6550 2.33GHz CPU, 4G main memory and Ubuntu 10.04.

In our implementation, filtering and pruning techniques proposed in [4] are not take into consideration in our implementation. The default of q is set to 2 and b and r is set to 1 and 3 respectively, and the default bucket number M is 20. Datasets used for the experimental evaluation are described in the following.

DBLP¹ is widely used in computer science fields as the benchmark. We extract the dataset into two: one is consisted of the paper titles, denoted as DBLP Title; the other includes the title and the author names of the paper, denoted as DBLP TA.

UniProt² is a protein sequence database, which is widely used for sequence alignments, retrieval et al. We sampled the string sets randomly, the length distribution and the statistical information of the sampled dataset is shown in Table 1.

5.2 Efficiency

In this section, the Trie-Traversal algorithm [4] and All-Pair-Ed [14] are used for the performance comparison. Figure 6 illustrates the performance comparison with Trie-Join and All-Pair-Ed with different Min-Hashing parameters for the string edit distance threshold τ varying from 1 to 3.

The performance comparison in Fig. 6 shows that, with the increasing of the threshold τ , the time cost of Trie-Join increases much faster. When τ is 3, our approach just takes about 50%–70% time of Trie-Join. With the increase of the hash function number in each band, the performance increases a little, because with the increasing of r , the probability of the strings hashed into the same bucket decreases. From the comparison result, we can see that both Trie-Join and our approach outperform the All-Pair-Ed on the datasets except the UniProt dataset. This is because that Trie-Join excels at short strings and the length of the UniProt dataset is 367, which is much longer.

¹ <http://www.informatik.uni-trier.de/~ley/db/>

² <http://www.uniprot.org/>

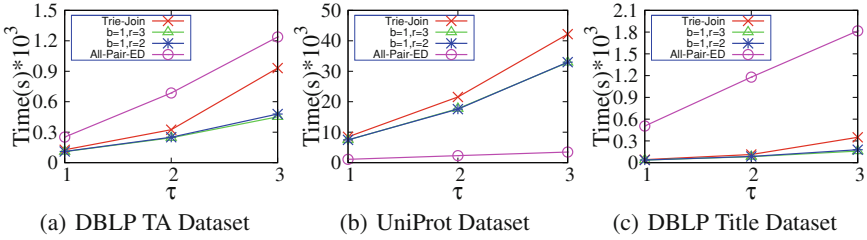


Fig. 6. Performance comparison

Table 2. Number of similar pairs with different τ

Dataset	$\tau = 1$	$\tau = 2$	$\tau = 3$
DBLP Title	1	3	7
DBLP TA	3	6	9
UniProt	1	252	402

From the above performance comparison experiment, conclusions can be drawn that: (1) performance improvement against Trie-Join with the increasing of the edit distance threshold; (2) As the Min-Hashing parameter r increases, the efficiency of our approach also increases. However, the join quality will decrease, because the bigger of r , the less of the candidates in the collision bucket; (3) All-Pair-Ed is more effective for long string set, which is illustrated by Fig. 6(b).

5.3 Quality

The similarity join quality results are reported in this section. Within each bucket, the *recall* measure is used and is defined as $recall = \frac{|retrieved \cap relevant|}{|retrieved|}$, where the *relevant* refer to the string pairs in the datasets that satisfying the string edit distance threshold, i.e., $relevant = \{ \langle s_i, s_j \rangle \mid s_i, s_j \in S, i < j, dist_{ed}(s_i, s_j) \leq \tau \}$; and the *retrieved* represents the retrieved result using our approach.

Table 2 summarizes the similar string pairs with different string edit distance τ varying from 1 to 3. The *recall* ratio is demonstrated in Fig. 7.

Figure 7 reveals the results of join quality with recall ratio metric. These results demonstrate that (1) with the increasing of the edit distance threshold τ , the recall ratio may reduce slightly. The reason is that strings within the edit distance threshold may be fallen into different buckets; (2) with the increase of the parameter r of Min-Hashing, the recall ratio may decrease, the Fig. 7(c) shows this obviously. The reason of recall ratio dropping is that with the increase of the parameters r , the false negative may increase, thus the similar string pair may be filtered with the LSH.

However, the join quality in Fig. 7(c) looks different with others when $r = 3$. The reason is that in DBLP Title dataset, there is only 1 similar string pair

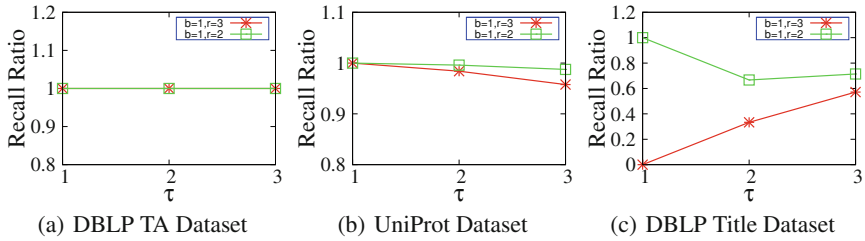


Fig. 7. Join quality with r on datasets

Table 3. Statics of the sampling datasets for scalability

Dataset	$ S $	Max len	Min len	Avg len
TA ₁	2321	238	10	66.233
TA ₂	4666	284	8	79.924
TA ₃	7035	284	8	81.892
TA ₄	14287	320	7	84.911
TA ₅	24314	432	7	84.076

in the dataset when threshold τ is 1, which is illustrate in Table 2. In addition, when $r = 3$, it may be missed with high probability, as a result, the recall ration is 0 if the similar pair missed. When the string edit distance threshold τ is equal to 2 and 3, there are 3 and 7 similar string pairs respectively in the dataset, and the recall can be increased in the experimental result. In consequence, the recall ratio looks different with Fig. 7(b).

Of course, the quality increases by reducing the parameter r . Nevertheless, reducing the parameter r can increase the probability of non-similar strings as well, i.e., increasing the false positive. When the string edit distance threshold τ increases, by reducing the parameter r and increasing b , the result quality can be tuned.

5.4 Scalability

For the scalability, the default parameters of Min-Hashing LSH b , r and bucket number M are set to 1, 3 and 100 respectively, and 5 groups of datasets are sampled randomly from DBLP with title and author, which are denoted as TA _{i} , $i = 1, \dots, 5$. Table 3 outlines the statistical information of the 5 sampled datasets.

Figure 8(a) demonstrates the scalability on the datasets with different string numbers. Experimental result show that, with the increasing of the size of the dataset, our approach scales well. Figure 8(b) indicates that our approach scales better, especially when the string edit distance threshold becomes larger. For example, when $\tau = 5$, it takes less than 20% time cost of Trie-Join.

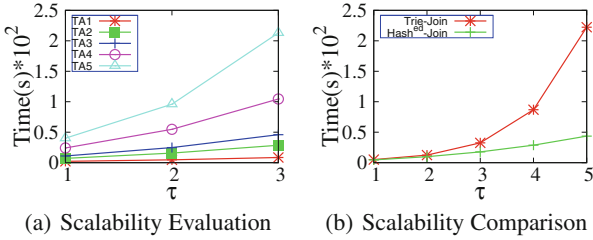


Fig. 8. Scalability evaluation

6 Related Work

String similarity join has been studied extensively in database fields [2, 10]. S. Chaudhuri et al. [2] introduce the similarity join for data cleaning and implement it as a primitive operator *SSJoin* in the relation database and the k -prefix filter based on the pigeon hole principle is proposed. Ed-Join [15] explores the *mismatching* q -gram for the edit distance constraint: content mismatching and location mismatching. All-Pair-Ed [14] is following filter-and-refine framework, which is based on the prefix filtering and follows an inverted list nested join style for the string similarity join. For each string, the first $q * \tau + 1$ q -grams are selected as the prefix, strings share q -gram with the prefix are verified by string edit distance evaluation. Hash-based filter-and-refine framework is employed in Spatio-Temporal indoor data tracking [16]. String similarity joins with synonyms is proposed [17], which is based on the term expansion framework.

Based the limitation of prefix filtering methods, [18] propose a cost model for selecting prefix. Top- k string similarity search with edit-distance constraints is proposed [19], which improves the pruning effective by using pivotal entries.

Recently, tree index based are proposed for string similarity join [4]. Z. Zhang et al. [20] propose the B^{ed} -tree with the edit distance constraint for string search. The Trie-Join [4] introduces the trie structure for prefix pruning, which benefits the string similarity join with edit distance constraint on short strings.

7 Conclusion and Future Work

In this paper, an approximate string similarity join using the Min-Hashing based LSH and the Trie-Join techniques is studied. Empirical study on real datasets indicates that our approach can effectively processing string join with high quality and better performance. As a future work, parallel evaluating will be considered.

Acknowledgments. This work was supported by the 973 project(No. 2010CB328106), NSFC grant (No. 61033007 and 61170085).

References

1. Wang, W., Xiao, C., Lin, X., Zhang, C.: Efficient approximate entity extraction with edit distance constraints. In: SIGMOD, pp. 759–770 (2009)
2. Chaudhuri, S., Ganti, V., Kaushik, R.: A primitive operator for similarity joins in data cleaning. In: ICDE, p. 5 (2006)
3. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: WWW, pp. 131–140 (2008)
4. Wang, J., Feng, J., Li, G.: Trie-join: efficient trie-based string similarity joins with edit distance constraints. VLDB **1**(1), 933–944 (2010)
5. Siragusa, E., Weese, D., Knut R.: Scalable string similarity search/join with approximate seeds and multiple backtracking. In: EDBT/ICDT, pp. 370–374. ACM (2013)
6. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. JACM **21**(1), 168–173 (1974)
7. Gouda, K., Rashad, M.: Prejoin: an efficient trie-based string similarity join algorithm. In: INFOS, pp. DE–37. IEEE (2012)
8. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: STOC, pp. 604–613 (1998)
9. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Sov. Phys. Dokl. **10**, 707–710 (1966)
10. Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S., Srivastava, D.: Approximate string joins in a database (almost) for free. In: VLDB, pp. 491–500 (2001)
11. Broder, A.Z.: On the resemblance and containment of documents. In: Proceedings of Compression and Complexity of Sequences, pp. 21–29 (1997)
12. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Multi-probe LSH: efficient indexing for high-dimensional similarity search. In: VLDB, pp. 950–961 (2007)
13. Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets. Cambridge University Press, New York (2013)
14. Bayardo, R.J., Ma, Y., Srikant, R.: Scaling up all pairs similarity search. In: WWW, pp. 131–140 (2007)
15. Xiao, C., Wang, W., Lin, X.: Ed-Join: an efficient algorithm for similarity joins with edit distance constraints. VLDB **1**(1), 933–944 (2008)
16. Lu, H., Yang, B., Jensen, C.S.: Spatio-temporal joins on symbolic indoor tracking data. In: ICDE, pp. 816–827 (2011)
17. Lu, J., Lin, C., Wang, W., Li, C., Wang, H.: String similarity measures and joins with synonyms. In: SIGMOD (2013)
18. Wang, J., Li, G., Feng, J.: Can we beat the prefix filtering? an adaptive framework for similarity join and search. In: SIGMOD, pp. 85–96 (2012)
19. Deng, D., Li, G., Feng, J., Li, W.-S.: Top-k string similarity search with edit-distance constraints. In: ICDE, pp. 925–936. ICDE (2013)
20. Zhang, Z., Hadjieleftheriou, M., Ooi, B.C., Srivastava, D.: B^{ed}-tree: an all-purpose index structure for string similarity search based on edit distance. In: SIGMOD, pp. 915–926 (2010)

Minimizing Explanations of Why-Not Questions

Chuanyu Zong^(✉), Bin Wang, Jing Sun, and Xiaochun Yang

College of Information Science and Engineering,
Northeastern University, Shenyang, China
{zongchuanyu,sunjing}@research.neu.edu.cn,
{yangxc,binwang}@mail.neu.edu.cn

Abstract. The problem of minimizing explanations of why-not questions exists in many scenarios such as query understanding and debugging. Several explaining techniques have been developed to minimize explanations of why-not questions, however, these techniques returned lots of bad explanations, including unreasonable explanations, incorrect explanations. Moreover, some of them returned all the possible explanations which waste time and space. To address this problem better, we propose a novel explaining approach to avoid such unexpected explanations and we guarantee that the generated explanations are correct and minimum. We propose one algorithm to address two situations: one for query statement contains a relation copy, and the other for query statement contains a query cycle. Experimental results demonstrate that our approach can efficiently get minimum explanations of why-not questions.

Keywords: Why-not questions · Minimum explanations · Data quality

1 Introduction

With the development of information extraction (IE) [2], why-not questions have received a growing attention in the hope of improving the usability of extracted database. Although IE can help users to find required information data quickly, data uncertainty is brought at the same time. Many users may ask *why the answer they expected is not in the query results* which called “why-not” [3] question, and we also call the why-not answer as missing answer. Therefore, it is becoming more and more important to answer why-not questions for users. Data provenance [1,4] records the origins and the evolution history of data. Many provenance technologies can explain why the answers are in the query results. But they cannot answer why the answers are not in the query results.

Motivation. To explain why-not questions, literature [10] has proposed a minimize explaining model based on value-modification. However, such technique has some disadvantages as illustrated by the example below.

The work is partially supported by the National Basic Research Program of China (973 Program) (No. 2012CB316201), the National Natural Science Foundation of China (Nos. 61322208, 61272178, 61129002), and the Doctoral Fund of Ministry of Education of China (No. 20110042110028).

EID	ENAME	AGE	GID
e_1	E1	Kehard	28
e_2	E2	James	29
e_3	E3	Adballah	30
e_4	E4	Gordana	27
e_5	E5	James	28

(a) Employee.

GID	GNAME	PID
g_1	G1	R*D
g_2	G2	Sales
g_3	G3	Project
g_4	G4	Program

(b) Group.

PID	PNAME	EID
p_1	P1	Provenance
p_2	P2	Similarity Join
p_3	P3	Data Quality
p_4	P4	Data Mining

(c) Project.

GID	GNUM
c_1	G1
c_2	G2
c_3	G3
c_4	G4

(d) Count.

Fig. 1. Extracted examples of different relations.

Example 1. Suppose that an IE application extracts partial IT company information for employees, work groups, and projects. And four extracted relations are shown in Fig. 1. Figure 1(a) shows a relation storing the names and ages of the employees and groups they belong to Fig. 1(b) shows a relation storing the names of the groups and projects they take part in Fig. 1(c) shows a relation storing the names and project managers of the projects. Figure 1(d) shows a relation that records the number of employees who belong to the same group. The value of `Count.GNUM` aggregately depends on attribute `Employee.GID`. And the attributes `Employee.EID`, `Group.GID`, `Group.GNAME`, `Group.PID`, `Project.PID`, `Project.PNAME`, and `Count.GID` are chosen as *trusted attributes*, whose values cannot be modified in the explanation.

By executing the following query Q , the IE application can extract the names of employees who are doing the project “similarity join.”

```
Select e.ENAME from Employee e, Group g, Project p
Where p.PNAME='Similarity Join' and g.PID=p.PID and e.GID=g.GID.
```

The answer to Q is **James**. The user may be confused that why answer **Gordana** does not appear in the query results, who belongs to the same work group with **James**.

Table 1 shows two explanations. Using explaining approach in [7], we can get explanation ex_1 , which modifies **James** in tuple e_2 to **Gordana**, then **Gordana** will appear in the query results. However, this explanation is unreasonable and incorrect, because the modification misses the existed answer **James**. Using explaining model in [10], we get explanation ex_2 , which modifies **G4** to **G2** in tuple e_4 , and does cascade modification from 1 to 0 in tuple c_4 , and from 1 to 2 in tuple c_2 . The explaining model in [10] can explain this why-not question based on the query Q , however, it does not work for the following two queries Q_1 and Q_2 .

Consider query Q_1 that a user wants to find the names of the employees who are doing the project managed by “James.”

```
Select e1.ENAME from Employee e, Employee e1, Group g, Project p
Where e.ENAME='James' and p.EID=e.EID and g.PID=p.PID and e1.GID=g.GID.
```

Table 1. Possible explanations for **Gordana**.

	EID	ENAME		AGE	GID		GID	GNAME	PID	PID	PNAME		EID	GID	GNUM	
ex_1	E2	James	→	Gordana	29	G2	G2	Sales	P2	P2	Similarity Join	E2	G4	1	→ 0	
ex_2	E4	Gordana			27	G4	→ G2	G2	Sales	P2	P2	Similarity Join	E2	G2	1	→ 2

We can get query results **James** and **Gordana** by executing Q_1 . The user may be confused that why **Adballah** does not appear in the query result. However, this query statement contains one relation copy of the relation **Employee**. We cannot use existed approaches explain why-not **Adballah**.

Moreover, by executing query Q_2 , one can know the age of “James” who is a project manager to lead a certain project, and the group he belongs to is also doing the same project.

```
Select distinct e.Age from Employee e, Group g, Project p
Where e.ENAME='James' and p.EID=e.EID and g.PID=p.PID and e.GID=g.GID.
```

Query Q_2 returns 29 but the user may want to know why 28 does not appear in the result. Checking Q_2 , the query predicates $p.EID=e.EID$, $g.PID=p.PID$ and $e.GID=g.GID$ constitute a cycle. Using existed explaining approaches cannot explain this why-not question with cycle join.

Challenges and Contributions. In this work, one challenge is to guarantee that the minimum explanations are correct. Moreover, the other challenge is to quickly get minimum explanations of why-not questions when query statement contains a relation copy or a query cycle.

We propose a novel technique for minimizing explanations of why-not questions. Our contributions can be summarized as follows. (i) We present a new approach to return correct and minimum explanations. (ii) We overcome those disadvantages of existed explaining models. (iii) We propose one algorithm to get minimum explanations for why-not questions when query statement contains a relation copy or a query cycle. (iv) We evaluate our algorithm using experiments in real world data set and synthetic data set.

The rest of the paper is organized as follows. We formally defines our problem and presents some related definitions in Sect. 2. In Sect. 3, we present our minimizing explanation algorithm when query statement contains a relation copy or a query cycle. We evaluate our explaining approach in Sect. 4. Section 5 concludes the paper.

Related Work. There have already existed some approaches for explaining why-not questions. For example, [7] explains why-not questions by telling users how to modify the original data, which related to our work. However, our work will return correct and minimum explanations for why-not questions. Literature [3] explains why-not questions by identifying the “culprit” operations which exclude the why-not answer in the query statement. Artemis [6] presents users what kinds of tuples should be inserted into the original database to explain why-not questions. This model and our work are both modifying the original database. But it bring up new data, while we only modify those values which already existed in the original database. Literature [5, 8, 9] explains why-not questions based on query refinement. They generates a refined query to make the new query results include both the original query answers and the why-not answers. Literature [5] addresses top- k queries, and [8] addresses reverse skyline queries. In [10] we propose a minimum explaining model to explain why-not questions

that also require to find the minimum explanations. However, we do not consider the issue that a query statement might contain a relation copy or a query cycle.

2 Preliminaries

We use the same explanation model in [10], if an attribute is a trusted attribute, the value of this attribute cannot be modified to explain why-not questions.

We continue to separate the original cell (we call a attribute value as a cell) from the modified cell by using an arrow (e.g. $v \rightarrow v'$). We only consider the equi-join between two relations T_1 and T_2 like $T_1.a_1 = T_2.a_2$, and the attributes a_1 and a_2 should satisfy referential integrity constraint.

Given an explanation ex , if some modifications in ex violate an aggregation constraint α or a functional dependency f , the other cells involved in α or f must be modified. We call those modifications *cascade modifications*. For example, consider explanation ex_2 in Table 1 and relations in Fig. 1, we modify **G4** to **G2** in tuple e_4 that would trigger two cascade modifications modifying the value of **Count.GNUM** to 0 in tuple c_4 and to 2 in tuple c_2 . Cascade modifications can help to improve the quality of minimizing explanations of why-not questions. We need to attach those cascade modified tuples to the end of the involved explanation to get minimum explanations.

Problem Definition. Given an extracted database, a query Q , and why-not w , the problem of minimizing explanations of why-not w is to find explanations such that the number of modified cells in the explanation is minimum among all the explanations for w .

For example, the number of modified cells in the explanation ex_2 is 3 as shown Table 1. The problem of minimizing explanations of why-not questions is to quickly find the minimum explanation ex_2 of why-not **Gordana**.

3 A Minimizing Explanations Algorithm

To get minimum explanations of why-not questions, we need to construct a why-not query template Q' such that based on Q' , its corresponding results only consists of answer to the user specified query Q and the why-not questions.

3.1 Why-Not Query Templates

According to the observation in [10], query predicates and why-not questions can be used as constraints to formalize the why-not query template.

Definition 1. Given a user specified query Q and a why-not question v , a query template is a virtual tuple $R(o_1, o_2, \dots, o_n)$, where R is the relation in Q and each o_i stands for a attribute value of the i -th attribute in R . Value o_i could be a predicate value in Q , the why-not question v , a variable, or a special variable satisfying a boolean expression.

Table 2. Query templates specified by both Q and why-not questions.

	Query template
qt_1	Employee (X_1 , Gordana , X_2 , J_1)
qt_2	Group (J_1 , X_3 , J_2)
qt_3	Project (J_2 , Similarity Join , X_4)

Consider query Q in Example 1 and the why-not question **Gordana**, its corresponding query templates are shown in Table 2. **Gordana** is the why-not question, **Similarity Join** is the predicate value in Q , variable J_1 and J_2 represents join attribute value in Q , and variables X_i represents other attribute values in the answers to the query template.

In order to get the answers to the query template, we need to modify tuples in underlying relations and map modified tuples to the query template. Then we get a minimum explanation by finding a mapping that the number of modified attribute values is minimum.

For example, consider query Q in Example 1 and why-not question **Gordana**. We can map the tuple **Project**(P2, **Similarity Join**, E2) to the query template **Project**(J_2 , **Similarity Join**, X_4) and map the tuple **Group**(G2, **Sales**, P2) to the query template **Group**(J_1 , X_3 , J_2). In order to get an explanation (an answer to the query templates in Table 2), we need to modify a tuple e_4 in **Employee** (see Fig. 1) and map the modified tuple to the query template **Employee**(X_1 , **Gordana**, X_2 , J_1). So the modified tuple is **Employee**(E2, **Gordana**, 27, G4→G2).

In our work, we define *tuple dissimilarity* as the number of different cells between one tuple in one relation and modified tuple matching to the query template of this tuple. For example, if J_1 equals to G2 and J_2 equals to P2, the tuple dissimilarity of e_4 is 3 including two cascade modifications, the tuple dissimilarity of g_2 is 0, and the tuple dissimilarity of p_2 is 0.

To avoid missing the existed query results and improve the quality of minimizing explanations, we have to confirm those tuples in the relations which can be used to generate the explanations of why-not questions.

Consider a tuple t contributes to the query result. If the tuple dissimilarity of t is not equal to 0, when we use t to explain the why-not questions, the cells in t which are different from their corresponding values in their query template must be modified to the same. However, modifying those different cells will result in missing existed answer. Therefore, t cannot be used to explain the why-not questions.

For example, consider query Q and why-not **Gordana**, tuple e_2 is contributes to generate query result **James**. The tuple dissimilarity of e_2 is greater than or equal to 1 no matter what join attribute value that J_1 can be determined. If we want to use e_2 to generate explanations for **Gordana**, we need to modify **James** in e_2 to **Gordana**. However, this modification would lead to miss existed query result **James**.

Consider a trusted attribute ta , if the corresponding value of ta in the query template is a constant $cons$, the value of ta which is not equivalent to $cons$

cannot be modified to *cons* to explain the why-not questions. Therefore we can only use the tuples whose value of *ta* are equivalent to *cons* to explain the why-not questions.

In our work, we call those tuples whose tuple dissimilarity is the smallest as the smTuples of the relation. Consider query Q and why-not **Gordana**, when confirm J_2 as P2 and J_1 as G2, the smTuples in **Employee** discarding a_2 (which is contribute to existed query result), in **Group** and in **Project** are e_4 , g_2 , and p_2 , respectively. Obviously the smTuples of one relation is constantly change with the changing of its query template when the query template contains variables.

3.2 Minimum Explanation for Queries with Relation Copy

In this section, we discuss how to get the minimum explanations of why-not questions when the query statement contains a relation copy.

Given two relations R_1 and R_2 , if one attributes in R_1 references to one attributes in R_2 , the reference link of R_1 and R_2 is defined as R_1 references to R_2 , and the reference order between R_1 and R_2 is defined as $R_1 \prec R_2$.

Considering query Q_1 in Example 1, there exist one relation copy of relation **Employee**. We need to rename this copy to distinguish between them because they have different query templates, and we call this copy as **CEmployee**. We can get the query templates for relations **Employee**, **Group**, **Project**, and **CEmployee** as $(J_1, \text{James}, X_1, X_2)$, (J_3, X_3, J_2) , (J_2, X_4, J_1) , and $(X_5, \text{Adballah}, X_6, J_3)$, respectively, according to Q_1 and why-not **Adballah**. These query templates contain variables, so we need to determine one relation's query template firstly, then we can determine the other relations' query templates in terms of equi-join.

Because the reference links between these relations constitute a cycle and Q_1 contains a relation copy, we cannot use the reference links to get an reference order for confirming those variables in the query templates. However, we can get an joining order according to the query statement.

We can find relation **Employee** in terms of the query predicate $e.EID = \text{'James'}$ firstly, then we can find **Project**, **Group**, and **CEmployee** orderly in terms of the query predicates $p.EID = e.EID$, $g.PID = p.PID$, and $e1.GID = g.GID$.

We can determine those query templates according to the order (**Employee**, **Project**, **Group**, **CEmployee**). If we determine the query template of **Employee** as $(E2, \text{James}, X_1, X_2)$, we can confirm the query templates of **Project**, **Group**, and **CEmployee** as $(P2, X_4, E2)$, $(G2, X_3, P2)$, and $(X_5, \text{Adballah}, X_6, G2)$, respectively. While if we confirm the query template of **Employee** as $(E5, \text{James}, X_1, X_2)$, we can confirm the query templates of **Project**, **Group**, and **CEmployee** as $(P4, X_4, E5)$, $(G4, X_3, P4)$, and $(X_5, \text{Adballah}, X_6, G4)$, respectively.

To get the correct explanations for **Adballah**, the tuples in **Employee** and **CEmployee** which are used to explain why-not **Adballah** should not be the same tuples. The reason is that we could modify **Adballah** to **James** in tuple e_3 and join with tuple e_3 in the process of explaining why-not **Adballah**, but this joining is unreasonable which would generate incorrect explanation like me_{x_1} as shown in Table 3.

Table 3. The minimum explanations for why-not question Adballah.

	EID	ENAME	AGE	GID	GID	GNAME	PID	PID	PNAME	EID	EID	ENAME	AGE	GID
mex_1	E3	Adballah	→ James	30	G3	G3	Project	P3	P3	Data Quality	E3	E3	Adballah	30 G3
mex_2	E2	James		29	G2	G3	Project	P3	P3	Data Quality	E3 → E2	E3	Adballah	30 G3
mex_3	E5	James		28	G2	G3	Project	P3	P3	Data Quality	E3 → E5	E3	Adballah	30 G3

After confirming those variables in query templates, we can get smTuples in relations Employee, Project, Group, and CEmployee to generate the minimum explanations for Adballah like mex_2 and mex_3 as shown in Table 3.

3.3 Minimum Explanation for Queries with Cycle Join

In this section, we will address the problem of minimizing explanations of why-not questions when the query statement contains a query cycle.

Wrong Cycle. For the same question that what are the names of employees who are doing the project managed by James. Maybe someone answering this question by executing the follow query Q_{1_w} :

```
Select e.ENAME From Employee e, Group g, Project p
where e.ENAME='James' And p.EID=e.EID And g.PID=p.PID And e.GID=g.GID.
```

Checking the query Q_{1_w} , we can see that query predicates $p.EID=e.EID$, $g.PID=p.PID$, and $e.GID=g.GID$ constitute a cycle. However, this cycle is a wrong cycle, because it makes the query only return the query answer James, but missing the query answer Adballah. We need to use a relation copy to replace this wrong cycle to meet the user’s query request, that is to say, we need to use query Q_1 to answer the user’s question mentioned in Example 1.

Necessary Cycle. According to the query Q_2 and why-not 28, we can get the query templates of Employee, Group, and Project as $(J_1, James, 28, J_3)$, (J_3, X_1, J_2) , and (J_2, X_2, J_1) , respectively. We cannot get an order to determine those variables in query templates by using traditional approaches. This is because their reference links constitute a cycle and the query statement also contains a cycle, we cannot get a start relation.

We can delete one reference link to get one reference order, which can also be used to as an joining order. Those variables can be confirmed based on the order, and partial confirming results are shown in Table 4. Then the minimum explanations for why-not question Adballah can be generated as shown in Table 5. However, we can see that the minimum explanations generated based on different orders are the same. The reason is that all the possible tuple joining combinations are need to be considered no matter based on which order. They are just different in the joining orders, but the overall joining operations are the same for the same why-not question. Therefore, we only need to explain the why-not questions based on one joining order.

Table 4. Confirm those variables in query templates for Q_2 .

Delete reference link	Reference order	Query templates		
		Employee	Group	Project
Group to Employee	Employee-Project-Group	(E5, James, 28, G3)	(G3, X ₁ , P4)	(P4, X ₂ , E5)
Employee to Project	Project-Group-Employee	(E1, James, 28, G1)	(G1, X ₁ , P1)	(P1, X ₂ , E1)
		(E5, James, 28, G4)	(G4, X ₁ , P4)	(P4, X ₂ , E5)
Project to Group	Group-Employee-Project	(E1, James, 28, G1)	(G1, X ₁ , P1)	(P1, X ₂ , E1)
		(E5, James, 28, G3)	(G3, X ₁ , P3)	(P3, X ₂ , E5)
		(E5, James, 28, G4)	(G4, X ₁ , P4)	(P4, X ₂ , E5)

Table 5. The minimizing explanations for why-not question 28.

Reference order	EID	ENAME	AGE	GID	GID	GNAME	PID	PID	PNAME	EID	
Employee-Project-Group	E1	Kehard	→ James	28	G1	G1	R*D	P1	P1	Provenance	E1
	E5	James		28	G3	G3	Project	P3	P3	Data Quality	E3 → E5
Project-Group-Employee	E1	Kehard	→ James	28	G1	G1	R*D	P1	P1	Provenance	E1
	E5	James		28	G3	G3	Project	P3	P3	Data Quality	E3 → E5
Group-Employee-Project	E1	Kehard	→ James	28	G1	G1	R*D	P1	P1	Provenance	E1
	E5	James		28	G3	G3	Project	P3	P3	Data Quality	E3 → E5

The biggest difference is that a certain relation need to be joined twice between this situation with the tradition query operation. For example, following the query statement and query semantic, the relation **Employee** need to be joined twice. We can get relation **Employee** firstly, then make a joining with **Project** and **Group** orderly. At last, make a joining with **Employee** again. One joining order can be got as **Employee, Project, Group, Employee**.

Algorithm 1: MinEx-CopyCycle

Input: The query Q , database D , why-not w and trusted attributes
Output: A minimizing explanation set E

- 1 Generate the query template for each query relation R ;
- 2 $R_{set} \leftarrow$ process contributing tuples and trusted attributes in R ;
- 3 $OSet \leftarrow$ get one joining order for the query relations;
- 4 **while** $OSet$ is not empty **do**
- 5 $Qt_{set} \leftarrow$ get a relation R and generate all the possible query templates of R ;
- 6 **if** R is in P_{set} **then**
- 7 $R_{set} \leftarrow$ get the smTuples of R from E_{set}
- 8 **while** Qt_{set} is not empty **do**
- 9 $Emp \leftarrow$ get one query template and generate the smTuples from R_{set} ;
- 10 $Emp' \leftarrow$ modify smTuples according to the query template;
- 11 $Emp'' \leftarrow$ attach cascade modifications tuples to Emp' ;
- 12 Add Emp'' to E_{set} and add R to P_{set} ;
- 13 $E_{x_{set}} \leftarrow$ make joins between smTuples in E_{set} ;
- 14 $E \leftarrow$ process $E_{x_{set}}$ to generate the minimal explanations ;
- 15 **return** E ;

In a word, Algorithm 1 can get minimum explanations when the query statement contains a relation copy or a query cycle.

Algorithm 1 generates the query template for each query relation R firstly (line 1). Then generates R_{set} by processing the contributing tuples to existed answers and trusted attributes in R (line 2). $OSet$ is generated by getting one joining order for relations in Q (line 3). We continuously get one relation in $OSet$

and generate Qt_{set} by getting all the possible query templates based on equi-join (line 5). We get all the smTuples in R_{set} and modify them based on the query template in Qt_{set} , if R has been manipulated, the R_{set} should be got from E_{set} , and we process cascade modifications and attach them to involved smTuples at the same time (lines 7-11). We add modified smTuples to E_{set} and manipulated relation to P_{set} (line 12). We do the join operations between smTuples of each relation in E_{set} (line 13). Finally, we compare the number of modified cells in each tuple in Ex_{set} to generate the minimum explanation set E (line 14).

In the process of getting the smTuples of relations in terms of the joining order, if R is the first relation, we need to consider all the tuples in R as its smTuples. If R is neither the first nor the last relation, R need to join with two relations. To improve efficiency, for the same left joining attribute value and right joining attribute value in one tuple of R , we only get the tuples whose tuple dissimilarity is the smallest as its smTuples. If R is the last relation, for the same right joining attributes value, we only need to get those tuples whose tuple dissimilarity is the smallest as its smTuples.

At the same time, we record the tuple dissimilarity of every smTuples of relations. When we finish a joining between relations to generate a explanation, we sum the accumulative tuple dissimilarity as the number of modified cells in this explanation. Finally, we can get the minimum explanations.

4 Experiments

Experiment Setup. We conducted the experiments on two data sets, including Synthetic data and DBLP data (see Table 6).

- Synthetic: We used an extension of the relations shown in Fig. 1, which included 2500 `Employee` tuples, 10 `Group` tuples, 10 `Project` tuples, and 10 `Count` tuples, respectively. We designed 3 queries $Q_{1_s} - Q_{3_s}$ and 3 corresponding why-not questions.
- DBLP: The DBLP data was extracted from DBLP Bibliography. It contains three relations `Author`, `Paper`, and `Cooperation`, including 2500 `Author`, `Paper` tuples, and 50 `Cooperation` tuples, respectively. We designed 3 queries $Q_{1_d} - Q_{3_d}$ and 3 corresponding why-not questions.

The algorithm was implemented using Java 1.7. Eclipse 4.3 was used as our Java IDE tool. The experiments were run on a PC with an Intel 3.10 GHz Quad Core CPU i5 and 8 GB memory with a 1 TB disk, running on a Windows 7 64-bit operating system. And we use MySQL 5.6 as our DBMS.

As shown in Table 6, queries Q_{1_s} and Q_{1_d} are common queries, queries Q_{2_s} and Q_{3_s} contain query cycles, and queries Q_{2_d} and Q_{3_d} contain relation copies. `Author2` is a copy of `Author`.

Comparison of explaining results with different data size. Table 7 shows the number of modified cells and the quantity of minimizing explanations for different queries when varying the number of tuples for `Employee` and `Author`

Table 6. Different queries on synthetic data set and DBLP data set.

Data sets	Qid	Queries	Why-not Questions
Synthetic	Q_{1_s}	$\Pi_{ENAME}(\sigma_{PNAME=Similarity\ Join}(Project) \bowtie Group \bowtie Employee)$	Fabrice Daumard
	Q_{3_s}	$\Pi_{Age}(\sigma_{ENAME=Samir\ Pokhrel}(Employee) \bowtie Group \bowtie Employee)$	28
	Q_{3_s}	$\Pi_{Age}(\sigma_{ENAME=E.F.Codd}(Employee) \bowtie Group \bowtie Employee)$	25
DBLP	Q_{1_d}	$\Pi_{ANAME}(\sigma_{YEAR=1974}(Paper) \bowtie Cooperation \bowtie Author)$	Markus Tresch
	Q_{2_d}	$\Pi_{Author2.ANAME}(\sigma_{ANAME=Rita\ Ley}(Author) \bowtie Cooperation \bowtie Author2)$	Linna Dong
	Q_{3_d}	$\Pi_{Author2.ANAME}(\sigma_{ANAME=Kristina\ Vuckovic}(Author) \bowtie Cooperation \bowtie Author2)$	Sara Librenjak

Table 7. Comparison of explanations for different queries.

Datasets	#tuples	Q_{1_p}		Q_{2_p}		Q_{3_p}	
		#modified cells	quantity	#modified cells	quantity	# modified cells	quantity
Synthetic data	500	3	1	2	53	2	54
	1000	3	1	2	105	2	91
	1500	3	1	1	1	1	1
	2000	3	1	1	1	1	1
	2500	3	1	1	1	1	1
DBLP data	500	2	90	2	48	2	25
	1000	2	90	2	48	2	25
	1500	2	90	2	48	2	25
	2000	2	90	2	48	2	50
	2500	2	90	2	48	2	50

from 500 to 2500. Taking queries Q_{1_s} as instance, the explaining results are the same with the increasing of the number of relation tuples. The reason is that the increased tuples does not contains corresponding values equal to why-not answers. For example, the tuple whose value of attribute `Employee.ENAME` equal to `Fabrice Daumard` has been existed and be used to generate a minimizing explanation of `Fabrice Daumard`. None of the values of attribute `Employee.ENAME` in increased tuples equal to `Fabrice Daumard`. Therefore, the minimizing explaining results have no changes. However, for the queries Q_{2_s} and Q_{3_d} , the quantity of minimizing explanations are growing with the increasing of the number of relation tuples. The reason is that the tuples which can be used to generate minimizing explanations is growing. The number of modified cells and quantity may also decrease when the increased tuples contain corresponding values equal to why-not answers like Q_{2_s} and Q_{3_s} , the details can be seen in Table 7.

Figure 2 shows the performance of our algorithm for different queries when increasing the number of tuples. Figure 2(a) shows the running time is very different for different queries. One reason is that the joining orders are different, which result in the number of tuple joining operations need to be finished is

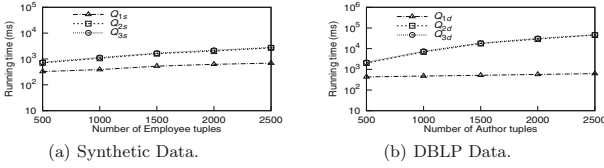


Fig. 2. Comparison of running time under different settings.

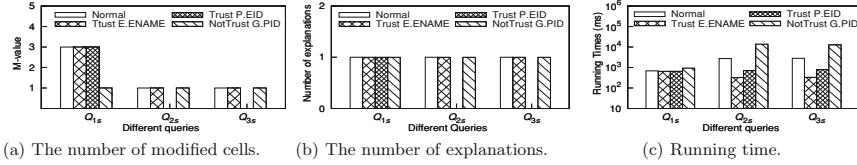


Fig. 3. Explanations under different trusted attributes for synthetic data.

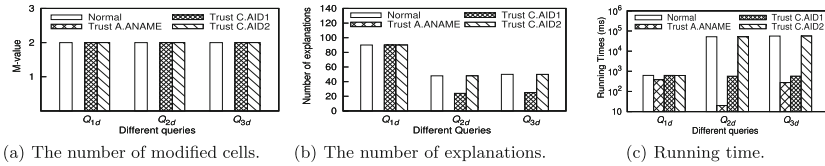


Fig. 4. Explanations under different trusted attributes for DBLP data.

different. However, for the similar queries, the changes of running time is similar such as queries Q_{2_s} and Q_{3_s} .

Figure 2(a) shows the running time for queries Q_{1_s} - Q_{3_s} is growing slowly with the increasing of relation tuples. As shown in Fig. 2(b), compare to the change of the running time for query Q_{1_d} , the running time for query Q_{2_d} and Q_{3_d} change significantly, the principle reason is that they contain a relation copy of the growing relation tuples, which equal to two relation tuples are growing.

Effect of different trust attributes. The minimizing explanations of why-not questions are also affected by choosing different trusted attributes.

For the Synthetic data, the trusted attributes are chosen as the same with Example 1. For the DBLP data, we chose the attributes `Author.AID` and `Paper.PID` as the trusted attributes.

In Figs. 3 and 4, we use P, E, G, C, and A to express `Project`, `Employee`, `Group`, `Cooperation`, and `Author`, respectively. In Fig. 3, the number of modified cells for Q_{1_s} is 3 when we no more trusting any attributes or trust `Employee.ENAME` or trust `Project.EID`. And we can get the number of modified cells is 1 when we no longer trusting `Group.PID`, which means the number of modified cells is 1 modify the values of `Group.PID`. As shown in Fig. 3(b), the number of explanations has no change for query Q_{1_s} . The running time is decreasing when we trust more

attributes, but increasing when we no longer trusting some attributes as shown Fig. 3(c).

For query Q_{2_s} which contains a query cycle, we cannot get any explanations for 28 when we trust `Project.EID` as shown in Fig. 3(a) and (b). They are also show that we can only modify the values of `Project.EID` to explain why-not 28. Figure 3(c) shows the running time changes obvious when we trusting different attributes for query Q_{2_s} and Q_{3_s} . The principle reason is that the more attributes we trust, the less the tuples can be used to explain why-not questions. Therefore, the higher the quality of extracted data, the faster the speed of getting minimizing explanations of why-not questions.

For query Q_{3_d} which contains a relation copy, we cannot get any explanations when we trusting `Author.ANAME` as shown in Fig. 4(a), and the number of explanations is decreasing when we trust `Cooperation.AID1` as shown in Fig. 4(b). This is because the minimizing explanations modify both the values of `Author.ANAME` and `Cooperation.AID1` which involved in query predicates. The explanation results changes nothing when we trust `Cooperation.AID2` because the minimizing explanations don't modify the values of `Cooperation.AID2`. As the same reason with query Q_{2_s} , the running time changes obvious when we choose to trust different quantity of attributes as shown in Fig. 4(c).

5 Conclusion

We propose an algorithm for explaining why-not questions when its related query contains relation copies or query cycles. Our experiments show that our approach can effectively return correct minimum explanations for the why-not questions. As parts of future work, we plan to study minimizing explanations of why-not questions, including a faster and effective explaining algorithms.

References

1. Buneman, P., Khanna, S., Tan, W.-C.: Why and where: a characterization of data provenance. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, p. 316. Springer, Heidelberg (2000)
2. Chang, C., Kayed, M., Girgis, M.R., Shaalan, K.F.: A survey of web information extraction systems. In: TKDE, pp. 1411–1428 (2006)
3. Chapman, A., Jagadish, H.: Why not '?'. In: SIGMOD, pp. 523–534 (2009)
4. Cheney, J., Chiticariu, L., Tan, W.C.: Provenance in databases: why, how and where. In: Foundations and Trends in Databases, pp. 379–474 (2009)
5. He, Z., Lo, E.: Answering why-not questions on top-k queris. In: ICDE, pp. 750–761 (2012)
6. Herschel, M., Hernández, M.A.: Explaining missing answers to spjua queries. In: PVLDB, pp. 185–196 (2010)
7. Huang, J., Chen, T., Doan, A., Naughton, J.F.: On the provenance of non-answers to queries over extracted data. In: PVLDB, pp. 736–747 (2008)
8. Islam, M.S., Zhou, R., Liu, C.: On answering why-not questions in reverse skyline queries. In: ICDE, pp. 973–984 (2013)

9. Tran, Q.T., Chan, C.Y.: How to conquer why-not questions. In: SIGMOD, pp. 15–26 (2010)
10. Zong, C., Yang, X., Wang, B., Zhang, J.: Minimizing explanations for missing answers to queries on databases. In: Meng, W., Feng, L., Bressan, S., Winiwarter, W., Song, W. (eds.) DASFAA 2013, Part I. LNCS, vol. 7825, pp. 254–268. Springer, Heidelberg (2013)

HadoopM: A Message-Enabled Data Processing System on Large Clusters

Wei Pan^{1,2}(✉), Zhanhuai Li^{1,2}, Bo Suo^{1,2}, and Zhuo Wang^{1,2}

¹ School of Computer Science and Technology,

Northwestern Polytechnical University, Xi'an 710072, China

² Guangdong Key Laboratory of Popular High Performance Computers,

Shenzhen Key Laboratory of Service Computing and Applications,

Shen'zhen 518060, China

panwei1002@nwpu.edu.cn

Abstract. MapReduce as a popular platform for solving embarrassingly parallel problems has been extensively used on large commodity clusters. However constrained by embarrassingly parallel assumption, some computation patterns are not easy to express in MapReduce, and in some cases performance and efficiency can not be achieved without communication between tasks, such as iteration and map phase filtration from a holistic perspective. This paper presents HadoopM, a message-enhanced version of Hadoop MapReduce architecture that it breaks the key embarrassingly parallel assumption and can execute the MR jobs in a more efficient and elegant way. HadoopM allows user-defined message to be passed between mappers or reducers by two message passing mechanisms: lightweight and heavyweight, and asynchronous and synchronous message passing are both supported by system. HadoopM retains the scalability and fault-tolerance of Hadoop and is binary compatible with Hadoop Mapreduce. Our experimental results demonstrate the superiority of modified version over original Hadoop MapReduce on a range of algorithms. In some cases, such as PageRank and Skyline, HadoopM significantly boosts the job performance up to 50 %.

1 Introduction

As one of the most representative distributed parallel computing paradigm initially proposed by Google, MapReduce [4] is powerful “hammer” for efficiently processing very large amounts of data, but would every problem looks like a nail? As a matter of fact, parallelism can be expressed in a variety of ways, each of which is applicable to only a subset of programs. Like any other parallel programming models, MapReduce can be applied for a specific range of issues which is known as embarrassingly parallel problem. Embarrassingly parallel problem

This work is sponsored by the National Basic Research Program (973 program) of China (No. 2012CB316203), the National Natural Science Foundation of China (Nos. 61033007, 61303037, 61332006), the National High Technology Research and Development Program (863 Program) of China (No. 2012AA011004).

is one for which little or no communication of results is required to separate the job into a number of parallel tasks, as map tasks or reduce tasks in MapReduce. As is well known, WordCount is one of such problem that can be broken down into smaller independent sub-problems whose results can be aggregated to produce the answer to the large or complex problem. The assumption of independent tasks allows MapReduce to flexibly partition the inputs, determine the execution order arbitrarily, and safely reschedule them in case of failures. View from another angle, this key assumption limits MapReduce's applicability and efficiency to handle more diverse set of parallel applications. Because some computation patterns are not easy to express in MapReduce without built-in support for communication between map tasks or reducer tasks. Here are some typical examples that aim to reveal the potential performance gain of job processing, if MapReduce could take advantage of communication mechanism in map or reduce inner-phase.

- Iteration is a common design pattern that can be seen in many data analysis applications, including PageRank, clustering, and community discovery in social network, in which data are processed repeated until the stopping criteria are satisfied. If there dose exist the channel for intermediate results exchange between the map tasks, the small quantities of loop-varying data between iterations can be synced among map tasks, and a large amount of data that may be unchanged during iterations will not have to be shuffled and re-loaded. As a result, an unnecessarily high costs of I/O, network bandwidth, and CPU resources can be avoided.
- Map-side pre-aggregation could help reduce the amount of intermediate results that have to be shuffled from map task to reduce task, as we know Combiner can fulfill this role in MapReduce. But without communication mechanism among map tasks, the current MapReduce can only carries out local filtering based on local view from individual map task. Thus, the data that can be decreased in global view but not in local view will have to be transfered to downstream. Obviously it is a time-wasting for reduce task to process these data. If there is existing communication mechanism between map tasks, a universal context of all map tasks could be gained, and then the data that obviously do not match the requirements could be eliminated for query processing, such as skyline, top-k and kNN.

The examples above indicates that inter-task message-passing is a new option for improving performance of MapReduce and extending the applicability of the model. Based on the insights from these examples, This paper presents a modified MapReduce architecture (Sect. 3.1) in which a small amount of meta-data (loop-varying data, aggregate results at local level, etc.) can be passed between map tasks, as well as reduce tasks, and while preserving the programming interfaces and fault tolerance models of native MapReduce frameworks. To validate this design, we developed the HadoopM prototype, a message-enabled version of Hadoop. Flexible message interface (Sect. 3.2) is provided for HadoopM to process arbitrary user-defined messages. Meanwhile, lightweight and heavy-weight message passing mechanisms (Sects. 3.3 and 3.4) are proposed to support

efficient implementation of algorithms. Furthermore, both synchronous and asynchronous message passing models (Sect. 4) are provided. We evaluated our system on two typical scenarios (Sect. 5) where performance benefit can be gained by using message. The experimental results demonstrate that communication mechanism can reduce job completion times by up to 50% in some scenarios.

2 Background

In this section, we review the MapReduce programming model based on the concept of embarrassingly parallel. The definition of embarrassingly parallel is given as follow:

Definition 1 (Embarrassingly parallel). *In parallel computing, suppose we are given a job \mathcal{J} which can be executed by a number of parallel tasks \mathcal{T} , input workload \mathcal{W} , and the communication cost \mathcal{C} between those parallel tasks. For any partition $\mathcal{P} = \{p_1, p_2, \dots, p_m\}$ of \mathcal{W} , where $p_i \cap p_j = \emptyset$ and $\cup_i p_i = \mathcal{W}$, $\mathcal{C}(\mathcal{T}(p_i), \mathcal{T}(p_j)) = 0, i \neq j$. The job \mathcal{J} can be expressed as: $\mathcal{J} = \cup_{i=1}^m \mathcal{T}(p_i)$.*

Here we use the above formalized definition of embarrassingly parallel to provide an in-depth analysis of existing MapReduce framework. From Fig. 1 we can see that the MapReduce execution schema consists mainly of two stages (map phase and reduce phase), each of which can be processed in pure parallel without any communications.

In the map phase the input dataset is divided into several disjoint equally-sized subsets called splits ($Split_0 \sim Split_m$), and then a corresponding number of map tasks ($Map_0 \sim Map_m$) should be created based on the amount of the splits with no interactive processing. Such an independent map task is hereinafter also referred to as mapper, which solely outputs new key-value pairs based on its own split. Thus map phase meets the definition of embarrassingly parallel.

Next take a look at reduce phase, the intermediate map-outputs are grouped to disjoint partitions ($Part_0 \sim Part_n$) by hashing, then the corresponding reduce tasks ($Reduce_0 \sim Reduce_n$) are launched to run independently and each takes one partition as input. Such an independent reduce task is hereinafter referred to as reducer. As map phase, reduce phase constituted by solely reducers also conforms to embarrassingly parallel definition.

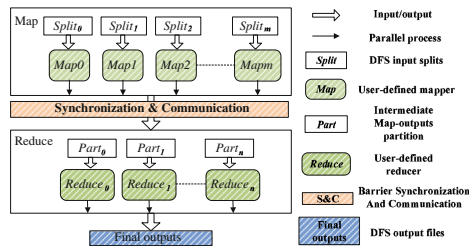


Fig. 1. Parallel programming model

Actually, as is illustrated in the picture, there is a implicit communication and synchronization which is transparent to user between map phase and reduce phase. And this is the only channel of data exchange during the execution of one MR job, the one-way data flow from upstream mapper to downstream reducer is known as shuffle, as each reducer is fed by many mappers.

By definition many applications can fit into this category, some prominent examples are counting words in documents, distributed search, inverted index and so on. And thus these problems are well-suited for solving by MapReduce. However, there are other categories of problems where there exist inevitable communication and dependency between parallel tasks. Without communication within mappers or reducers, we can only make use of shuffle to exchange data, which will incurred a lot of unnecessary overhead to write intermediate data to disk and transfer them on network. For example, the naive way to express PageRank algorithm on MapReduce is to chain multiple MapReduce jobs together for iterative processing. Besides loop-varying data, the unchanged data which is a significant fraction of the all data have to be shuffled and re-loaded at each iteration, and additional costs will be incurred to start multiple MapReduce jobs for one problem.

The above further analysis reveals the potential opportunity for improving the performance by using inter-mapper or inter-reducer communication and more flexible parallel model, as well be seen in Sect. 4.2, BSP model is adopted by HadoopM.

3 A New Message-Enabled Platform

Based on the insights from our analysis in before sections, we next propose a new data analysis platform-HadoopM that enable user-defined messages to be transferred either in map phase or reduce phase. In this section we introduce HadoopM's architecture, message format, lightweight and heavyweight message delivery modes.

3.1 Architecture

Our HadoopM prototype modifies the internals (TaskTracker, JobTracker, etc.) of Hadoop by adding several key message components to support message passing, which are shaded blocks in Fig. 2. Thereinto, **Message Transmitter & Receiver** is responsible for submitting, forwarding and receiving message among nodes in cluster, in addition, message serialization/de-serialization and message assembly are also managed by this new component. When the node receives the user-defined message, **Message Handler** component will be invoked to resolve this message. By separating the fixed format messages header (Sect. 3.2) from message, **Message Handler** will re-organize these messages from different nodes according different application requirements, then buffer and combine them into **Message Cache**. For communication-intensive application or large-size message, HadoopM could place message collections into a persistent storage, such

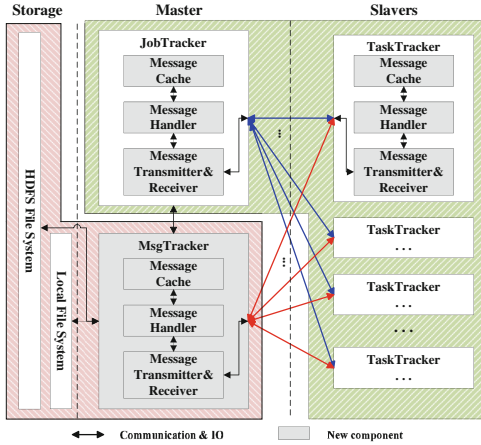


Fig. 2. The HadoopM framework

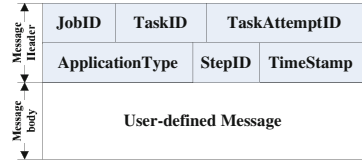


Fig. 3. Message format

as local file system or HDFS when available main memory can not afford too many messages. More details are discussed in later sections.

On the different message passing mechanisms the changes involved are differentiated, the lightweight message delivery mode in shade of green, and the heavyweight mode in shade of red as shown in Fig. 2. The lightweight message delivery mode trying to make full use of existing communication architecture of Hadoop for communication-sparse application or size-constrained message. And for communication-intensive application or large-size message, a new master node named **MsgTracker** will be in charge of all messages, which can work with JobTracker as well as independently. Before introducing the two message delivery modes, we first take a look at the new message interface.

3.2 Message Format

Figure 3 shows the new message format which consists of two basic parts: message header and message body. In HadoopM, message should be assigned to a unique identifier which refers to message header, composed of *JobID*, *TaskID*, *TaskAttemptID*, *ApplicationType*, *StepID* and *TimeStamp*. JobID, TaskID and TaskAttemptID were designed to uniquely identify job, task and speculative execution or re-execution of task respectively. ApplicationType can be used to indicate how to use message and what kind of transfer mode (synchronous or asynchronous mode) should be used in message-passing. Specifically in the case of synchronous mode, StepID is used to distinguish the super step that message attached to. Finally, added with message timestamp, the unique message header can be generated, which be referred to as message-identifier. Throughout the HadoopM, this unique message-identifier can be used for message identification and management.

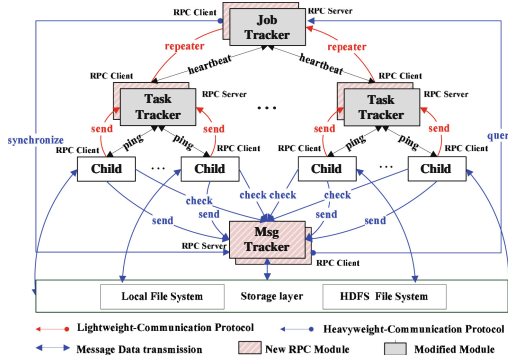


Fig. 4. Two message delivery modes

Message body is the content of the user-defined message represented by an implementation of `MPIMessage` - a new message interface, which is defined in HadoopM. And to support message passing and persistence, serializability of the user-defined message body is mandatory.

3.3 Lightweight Message Delivery Mode

The lightweight message delivery mode relies mainly on existing communication architecture, and several important changes should be made on original Hadoop framework. Implementation details are illustrated in Figs. 2 and 4, the main components are involved in lightweight mode including *JobTracker*, *TaskTracker* and *Child*, The message delivery procedure can be divided into two phases, *the push phase* and *the pull phase*.

In the push phase, each *Child* applies message-enabled map (or reduce) function to submit the user-defined message to *TaskTracker* when communication is needed. And message component is responsible for creating message header and assembling integrated message. *TaskTracker* uses *Message Transmitter & Receiver* to forward message that received from *Child* to *JobTracker*, accompanying with redundant message detection and cleaning. And then *Message Transmitter & Receiver* resident in *JobTracker* to be invoked upon incoming message. Based on the analysis of the message header carried out by message receiver, the message will be inserted at the right place into *Message Cache*, a efficient multi-branch tree used to maintenance entire cluster message topology. From the above several steps, *JobTracker* may aggregate user-defined messages from different execution nodes, and the aggregation of messages can form global view from *Message Cache*.

In the pull phase, we adopted receiver-pull model to let *Child* pull the combined message via *TaskTracker* from *JobTracker*. Receiver-pull model takes advantage of the fact that receivers (*TaskTracker* or *Child*) have more knowledge of what message they want to receive. Act as receiver, *TaskTracker* and

Child initiate the combined message transfer by exploiting periodic heartbeat and ping.

To support message-passing, Meanwhile, HadoopM modified and added some communication protocols. Ping and heartbeat are adjusted to allow for bring combined message back, moreover the new protocols, *send* and *repeater*, are responsible for submitting message to upper message acquisition node at any time, shown here in red line.

Lightweight message delivery mode is a straightforward and effective approach for less frequent communication and small-size message, But there are some potential problems that can not be ignored. In lightweight mode, JobTracker should need to coordinate jobs run on cluster as well as user-defined messages. As the sole master node, JobTracker could easily become the bottleneck, and message receiving delay inherent with periodical ping and heartbeat could become more longer with increases of the cluster size. So we need a dedicated mode which can separate message handling from JobTracker, here comes the Heavyweight message delivery mode.

3.4 Heavyweight Message Delivery Mode

The heavyweight message delivery mode designed for solving the above problem provides a new daemon named *MsgTracker* (Message Tracker) that can run on a separate node in cluster, as shown in Fig. 4. The new daemon can turn on and off along with Hadoop Cluster, as well as open and close separately.

MsgTracker can take charge of message management in complete coordination with core components in Hadoop. The way of collaboration between MsgTracker and existing components described as follows.

In MsgTracker, an RPC server is created in response to receive user-defined messages send by Childs and some initial status informations about jobs and TaskTrackers maintained in JobTracker. After RPC server resident in JobTracker starts, MsgTracker has the JobTracker's RPC client make an RPC call to a function on the JobTracker named *query*, where the JobTracker will gather information that MsgTracker needed into return value of query function. After completing initialization with return value, the standalone MsgTracker server can handle processing for all messages in whole cluster. During operation, JobTracker can use MsgTracker's RPC client to update the status of jobs to MsgTracker actively. As shown in the graph, new function of communication protocol named *synchronize* can be invoked in response to RPC call. In this way, HadoopM can keep status information in sync between JobTracker and MsgTracker. Child has the MsgTracker's RPC client make an RPC call to two functions on the MsgTracker named *send* and *check*. The *send* function guarantee real-time message transmission, and the message processing results can be got by using periodicity *check*. Either in MsgTracker or Child, each job or task has a memory buffer that stores messages. When the buffer reaches the cap, message will be spilled to disk, either local file system or HDFS. Especially for large-size message, Child is just sent metadata of user-define message to MsgTracker in order to reduce network overheads. In the implementation, the real message body will be spill to local

file system of Child, and message body will replace with output locations. The end-consumer for message could fetch message entity from location specified in message body.

Currently, as can be seen from Fig. 4, inter-Child and inter-TaskTracker have no direct communication. Since with Hadoop assumption-“hardware failure is the normal rather than the exception”, making direct communication between inter-TaskTracker and inter-Child will increase coupling degree between these components, excessive coupling and potential node failure will make cluster communication difficult to maintain. In the current design, all messages come from each node will converge to JobTracker or MsgTracker, which will in charge of coordinating and managing all message transfered over MapReduce cluster based on the entire jobs execution view.

4 Transmission Synchronization

4.1 Asynchronous Message Passing

Asynchronous communication mode is straightforward and free-lock, HadoopM provides the in-built support for this mode based on original MapReduce computing framework. Asynchronous message passing between tasks enables a task to post some metadata (such as local aggregation) about their current state and see state of all other tasks, and then they can get an aggregate view of multi-tasks from message server (TaskTracker or MsgTracker) to make globally coordinated optimization decisions without waiting.

Then we use top-k (i.e. $k=3$) query to illustrate how to use asynchronous mode to optimize performance. Pretend there are three parallel mappers, one of which ($mapper_1$) processes its own split (3, 7, 21, 28, 33, 62) and generates its local filter value (28), and then sends it to the message server. After receives the filter values (such as 28, 35, 19) of all mappers that submitted by messages, message server can generate a global filter value and chooses the most optimal one as the global filter value (35). Then mapper filter its intermediate data using the global filter value from the message server. For $mapper_1$, only 62 meets requirement, meanwhile, all output produced by mapper that send 19 as a local filter value can be filtered, So less reduce input than original MapReduce can improve performance.

The reasons for using asynchronous communication mode mainly in the following two areas. Firstly the correctness of one job processing result is not affected by message, the tasks only use the message to fulfill job optimization needs. Secondly MR job can have many waves of tasks in large cluster containing sufficient free slots, after each task completes, it sends its filter value to the message server and seek to receive a global filter value simultaneously. Even if the global filter value is null, the task can processes its split normally. Otherwise, the task can prune its intermediate results with the global filter value. So waiting is not required, the tasks that start on the successive rounds will surely get the more optimal global filter value to reduce the unpromising intermediate data.

4.2 Synchronous Message Passing

In some case, data carried by message is not just for optimization stated before, which have directly influence on the validity of execution result. Before moving forward to the next step each task needs to wait for all messages to be submitted by other tasks and accepted by message server. Because only when all messages have been collected the correctness of the result can be guaranteed, so synchronization techniques must be used.

To accommodate such demand, we modified the existing computing framework inspired by the BSP (Bulk Synchronous Parallel) [14], to support synchronous message passing in HadoopM. BSP computation consists of a sequence of supersteps where three ordered stages are involved: concurrent computation, communication and barrier synchronization. Thereinto, barrier is a mechanism for synchronizing tasks executing in parallel. We note that in the Fig. 5, The map and reduce phase were split into several supersteps by barriers, and in each superstep, mapper and reducer is further divided into two phases: *the active phase* and *the wait (inactive) phase*. Active mapper executes its code until it reaches a barrier, then it enters wait state until all other mappers have also reached the same barrier. Owing to various reasons, including input data distribution or hardware variations, mappers may execute the same code at different speeds and enter the barrier at different times. As shown in Fig. 5, *Map₂* is the last map task arrived at the barrier, until this, all other mappers has entered inactive state. When all mappers have arrived at the same barrier, all of them will be woken up and can continue to run into next superstep with the new data gathered from other mappers.

We also note that in the Fig. 5 where three types of barriers exist: *Mapper-Barrier*, *MR-Barrier* and *Reducer-Barrier*. MR-Barrier is original implicit barrier be used between the map and reduce stages in current MapReduce framework. In HadoopM, Mapper-Barrier and Reducer-Barrier are the newly added

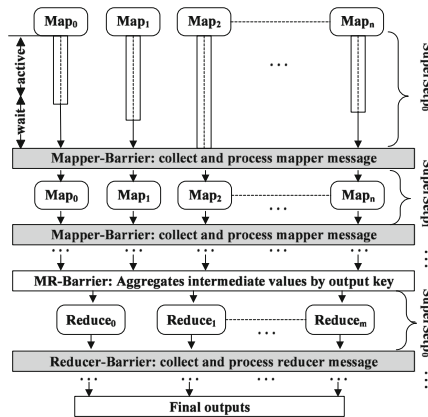


Fig. 5. Barriers in HadoopM

explicit barriers for providing the synchronization mechanisms based on message passing. They ensure that all relevant messages exchanged are ready to the next round of task.

This extended programming model used by HadoopM can cover more complex applications in one-round MR job to replace multi-round MR jobs where data can be exchanged between mappers or reducers with synchronous message passing. PageRank is a classic example that can benefit from this model, in which each superstep corresponds to one iteration. Within a superstep, task can evaluate PR (PageRank) value for each vertex contained in its split, and then at inter-superstep PR value can be passed by message in synchronous mode. After the Mapper-Barrier was released, all tasks can evaluate new PR value repeatedly with original split and loop-varying PR value until the stopping criterion or a convergence is satisfied. In such way, PageRank can achieve by using one-round map-only job and performance can be improved.

5 Experiments

In this section, two typical types of application, Skyline and PageRank are taken as examples to evaluate the job processing performance of our HadoopM framework.

5.1 Experimental Environment

For this evaluation, we used a cluster of 11 nodes with 4 Core 2.66 GHZ CPU, 4 GB memory and Ubuntu 9.10 each. We implemented HadoopM on Hadoop 0.20.2 and use the message-enabled version to run all benchmarks. We used the following two datasets and increased their sizes as needed. One is soc-LiveJournal 1 G from Stanford Large Network Dataset Collection for PageRank and increasing the data size from 5 G to 45 G without changing the graph topology (only added some description to vertex and edge). Another is a synthetically generated dataset for Skyline where the default number of data records is 1000 K, ranging from 200 K to 1000 K and the number of data dimensions of skyline query is 4. We compare HadoopM with Hadoop through running skyline query and PageRank. The main experimental benchmarks is the running time of job processing.

5.2 Experiments of PageRank

Figure 6 represents the overall running time of entire job, the result indicates that HadoopM performs significantly better than Hadoop, reducing the average running time to 50 % due to using inter-mappers communication to exchange loop-varied PR value. We also can see from Fig. 6(a), the more numerous iterations, the more reward from communication. As we described in Sect. 4.2, HadoopM adopting multi-round supersteps within one job to replace multi-round jobs, is able to reduce the expensive I/O cost for re-loading and shuffling unchanged data in subsequent iterations, the startup overhead for multi-round job and etc.,

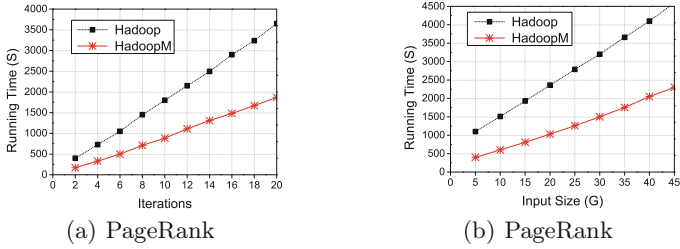


Fig. 6. Running time of PageRank

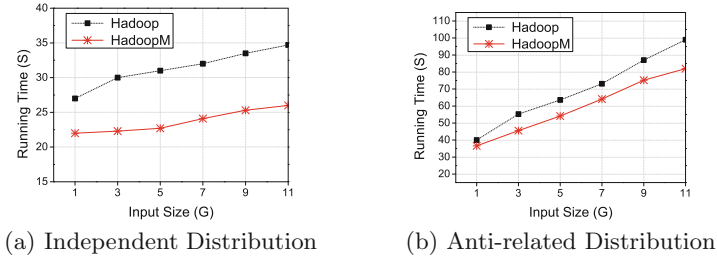


Fig. 7. Running time of Skyline

in other words, HadoopM use the cheaper network communication cost (including synchronous cost) to replace the expensive IO cost of static graph shuffling. Moreover Fig. 6(b) also shows that HadoopM has linear scalability on the number of iterations. To examine the scalability of the HadoopM’s synchronous behavior for workload size to PageRank, we perform exactly the same experiment on different data size from 15 G to 45 G. From Fig. 6(b), experimental results show that HadoopM also has linear scalability on the size of workload.

5.3 Experiments of Skyline Query

As it well known, the performance of Skyline query is closely related to the dataset distribution. During the experiments, the skyline query is evaluated in independent distribution and anti-related distribution. Figure 7 shows the performance of changing the data size of skyline query. As the data size increases, the running time increase dramatically. HadoopM is more optimal to Hadoop under different data distributions. In Fig. 7(b), although the data objects skew to the final results and the computation cost increases accordingly, the performance of HadoopM is still optimal to Hadoop. Because HadoopM can filter the unpromising data objects with synchronous message passing mechanisms, so the number of input records fetched by reducer of HadoopM is fewer than Hadoop. The detailed reason have been outlined in Sect. 4.1.

6 Related Work

Google's MapReduce, a popular framework for performing data intensive computation on a large cluster, has gained a lot of attention in academia [3, 6, 9–11] in recent years. Through these research efforts, the framework has been extended for diverse application requirements. But the above studies are still based on the assumption of embarrassingly parallel. There are some mapreduce-like system that have been developed by using different parallel model, such as Pregel [12] and HAMA [13]. Their studies about BSP model inspires our work on iterative processing data. Especially for iterative processing, a number of studies, such as HaLoop [1], have putted their efforts to improve the MapReduce framework for such computation pattern [2, 7, 8, 16]. Also, several skyline query algorithms based on MapReduce are also developed, which can be seen in the literature [5, 15]. Though non-adaptive, these techniques are complimentary to our approach, HadoopM adopts efficient communication mechanisms to solve problems which require data exchanging via inter-task in more elegant way.

7 Conclusions

This paper demonstrated that MapReduce framework with inter-task communication are feasible, and they can result in significant performance benefits. We present the architecture, implementation, and evaluation of HadoopM, a revised message-enabled Hadoop system, which is aimed to improve performance and expressive ability by building the communication channel either in inter-mapper or inter-reducer, build-in lightweight/heavyweight communication mechanism, message interface, and asynchronous/synchronous communication mode can support a variety of communication needs for different applications. We evaluated our HadoopM prototype on PageRank and Skyline query, The experimental results shows that HadoopM can significantly improve the progress of job.

References

1. Bu, Y., Howe, B., Balazinska, M., Ernst, M.D.: Haloop: efficient iterative data processing on large clusters. *PVLDB* **3**(1), 285–296 (2010)
2. Chu, C.T., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G.R., Ng, A.Y., Olukotun, K.: Map-reduce for machine learning on multicore. In: *NIPS'06*, pp. 281–288. MIT Press (2006)
3. Condie, T., Conway, N., Alvaro, P., Hellerstein, J.M.: Mapreduce online. In: *NSDI'10*, pp. 21–21 (2010)
4. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: *OSDI'04*, pp. 137–150 (2004)
5. Ding, L.-L., Xin, J., Wang, G., Huang, S.: Efficient skyline query processing of massive data based on map-reduce. *Chin. J. Comput.* **10**, 1785–1796 (2011)
6. Dittrich, J., Quian-Ruiz, J.-A., Jindal, A., Kargin, Y., Setty, V., Schad, J.: Hadoop++: making a yellow elephant run like a cheetah (without it even noticing). *PVLDB* **3**(1), 518–529 (2010)

7. Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J., Fox, G.: Twister: a runtime for iterative MapReduce. In: HPDC'10, pp. 810–818. ACM (2010)
8. Elnikety, E., Elsayed, T., Ramadan, H.E.: iHadoop: asynchronous iterations for MapReduce. In: CloudCom'11, pp. 81–90. IEEE (2011)
9. Floratou, A., Patel, J.M., Shekita, E.J., Tata, S.: Column-oriented storage techniques for MapReduce. PVLDB **4**(7), 419–429 (2011)
10. Jahani, E., Cafarella, M.J., Ré, C.: Automatic optimization for MapReduce programs. PVLDB **4**(6), 385–396 (2011)
11. Li, B., Mazur, E., Diao, Y., McGregor, A., Shenoy, P.J.: A platform for scalable one-pass analytics using MapReduce. In: SIGMOD'11, pp. 985–996 (2011)
12. Malewicz, G., Austern, M.H., Bik, A.J.C., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: SIGMOD'10, pp. 135–146 (2010)
13. Seo, S., Yoon, E.J., Kim, J., Jin, S., Kim, J.-S., Maeng, S.: Hama: an efficient matrix computation with the MapReduce framework. In: CloudCom'10, pp. 721–726 (2010)
14. Valiant, L.G.: A bridging model for parallel computation. Commun. ACM **33**, 103–111 (1990)
15. Zhang, B., Zhou, S., Guan, J.: Adapting skyline computation to the MapReduce framework: algorithms and experiments. In: Xu, J., Yu, G., Zhou, S., Unland, R. (eds.) DASFAA Workshops 2011. LNCS, vol. 6637, pp. 403–414. Springer, Heidelberg (2011)
16. Zhang, Y., Gao, Q., Gao, L., Wang, C.: iMapReduce: a distributed computing framework for iterative computation. In: IPDPS Workshops'11, pp. 1112–1121. IEEE (2011)

AntiqueData: A Proxy to Maintain Computational Transparency in Cloud

Himel Dev^(✉), Mohammed Eunos Ali, Tanmoy Sen, and Madhusudan Basak

Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
{himeldev,mohammed.eunos.ali,sen.buet,madhusudan.buet}@gmail.com

Abstract. Cloud computing offers computing and software services to users on an on-demand basis. It facilitates users to use computing resources as utility with pay-per-usage billing, which allows users to acquire computational resources with low or no initial cost. Due to this greater level of flexibility, the cloud has become the breeding ground of a new generation of products and services. Since more and more people rely on the cloud with their data and computing, ensuring the trustworthiness of cloud services has become a major issue for both the users and cloud providers. Due to the black box nature of cloud, there has been a lack of trust among providers and users, which has become a major barrier to the widespread growth of cloud computing. One of the trust concerns of cloud is lack of computational transparency. In particular, in current cloud architecture a provider controls all the logging and auditing records corresponding to computation and users do not have access to these records. This is a big concern for many clients of cloud. In this paper, we first identify the risks associated with lack of transparency in cloud and propose a middleware service that eliminates these risks.

1 Introduction

Cloud computing offers heterogenous services such as storage, computation, and applications to users on an on-demand basis. It is a form of utility computing that facilitates users with pay-per-usage billing, which requires low or no initial cost to acquire computational resources as resources are essentially rented from the cloud service providers. Tech giants such as Amazon, Google, Microsoft are providing cloud services of various forms. Amazon Elastic Compute Cloud (EC2), Simple Queue Service (SQS), Simple Storage Service (S3), Google App Engine (GAE), Windows Azure, SQL Azure, and Windows Intune are some of these services.

Since the emergence of cloud computing as a prominent medium of acquiring high performance e.g., mass storage, high processing power at a low cost, there have been numerous discussions on its trustworthiness [1]. There has been a lack of trust among users and providers of cloud. Eliminating this lack of trust and creating a trustworthy platform of resource management is one of the greatest challenges of cloud computing.

To discuss the issue, let us consider an example scenario where a company Titans is using cloud services provided by a company Hydra. Now, a malicious employee Hera from Titans has done something illegal (e.g., Denial-of-Service (DoS) attack on a site or mining sensitive unauthorized data) using the resources provided by Hydra. When the victim charges Hydra and Hydra in turn charges Titans, Hera denies the charges. Moreover, she accuses another member Hyercules of Titans. Hyercules has no way to prove his innocence as both Hera and Hyercules were using the resources at the same time and no one knows who was doing what. Moreover, if Hydra somehow collaborates with Hera, Hyercules will have no way to escape. This raises a serious trust issue within the users of the cloud.

The trust issue discussed above arises from the fact that users of cloud do not have access to the logging and auditing records of tasks performed by them in cloud. The associated mechanisms are fully controlled by the cloud providers. Hence, users cannot verify their tasks. Again, some applications such as eScience and healthcare needs to store records corresponding to computational tasks [2]. These records are used to analyze experimental results and other uses. The absence of computational records is a major barrier to the widespread growth of these applications in cloud computing platform. Another fact to consider is, in current cloud model, users can not be certain about different resource usages. In particular, a user cannot confirm that (i) the billing statements accurately reflect real use of resources, (ii) the deployed resources were up and running all the time, (iii) there is any security breaches or malfunctions affecting the outsourced resources [1]. As a consequence, a mistrust may build-up among users and providers.

Substantial work has been done to establish the trust among users and providers of cloud. But most of these works tend to establish trust in the storage context by maintaining the provenance of data objects [1]. However, ensuring trust in the context of processing (i.e., when users use virtual machines or applications provided by cloud) is still an open issue. Now-a-days many are proposing sophisticated cloud based applications such as software testing using cloud [3], cloud based malicious site detection [4,5], cloud based data mining [6], using cloud to conceal IP [7] etc. But these applications can be a massive security threat if not monitored properly. For example, some user may use multiple VMs to perform denial-of-service (DoS) attack on a site. Similarly, some may mine unauthorized data to misuse it. In a nutshell, a malicious user may perform illegal activities using cloud and accuse another user for it. The situation becomes more complicated if the cloud provider joins the malicious user. Again, a malicious provider may apply excessive charges on its clients. To eliminate such catastrophe and to establish the trust within users and also among users and providers, computational transparency is required.

In this paper, we present a middleware service, *AntiqueData proxy* that will establish trust among users and providers by introducing computational transparency. The system works by collecting information from users and providers after every session of data processing and storing these information as computational

provenance records. Session in this context refers to the lapse of time a user passes using a set of cloud resources. The records stored by *AntiqueData proxy* are session specific and are used to maintain transparency. These records are accessible to cloud providers and users in a hierarchically restricted way. The proxy also allows users to check their resource usage and thus ensure proper billing.

2 Background and Related Work

With the increase of popularity of cloud as a storage and computation medium, the demand for a trusted cloud structure increased simultaneously. So, the idea of provenance emerged in the field of cloud computing. Provenance generally refers to the information that helps to determine the derivation history of a data product, starting from its original sources [8]. Numerous techniques have been proposed for provenance in the cloud system. Among the proposed techniques, Provenance-Aware Storage System (PASS) [9] is considered a pioneer. Reddy et al. discussed the requirements of adding the provenance data to the cloud storage and four properties to make the provenance system truly useful. They proposed three protocols that monitor the client system call and stores both the provenance and data to AWS S3 storage. These three protocols use AWS S3, SimpleDB and SQS service hierarchically to ensure the properties like provenance data coupling, efficient query, and causal ordering, respectively [10, 11]. Zhang recently proposed an approach named dataPROVE [2] that maintains provenance data depending on the resource granularities [12]. Reilly and Naughton have proposed extending the Condor batch execution system [13] to capture data in execution environments, machine identities, log files, and file permissions. While there are significant new challenges on a cloud infrastructure, the Provenance-Aware Condor system certainly collects the right kind of provenance data. Abbadi et al. [14, 15] proposed use of middleware at different layers of cloud structure to maintain required provenance data.

Most of the approaches mentioned above deals with provenance associated with data storage in cloud. Again, almost all of the proposals involve storing object based provenance [1] data which are fully deployed and controlled by cloud providers and are not reasonably protected. This in turn questions the credibility of provenance data in the cloud. As a result, it affects the integrity of cloud built upon client and provider's mutual trust.

Recently Park et al. [16] introduced a new system RAMP for capturing and tracing provenance in MapReduce workflows. RAMP (Reduce And Map Provenance) is an extension to Hadoop that supports provenance capture and tracing for workflows of MapReduce jobs. Akoush et al. [17] introduced HadoopProv, a modified version of Hadoop that implements provenance capture and analysis in MapReduce jobs but with reduced provenance capture overhead. Whereas all these systems focus on debugging the Mapreduce workflows, they rely on data provenance to serve the purpose.

3 Threat Model

Numerous applications of heterogenous diversity are being developed on cloud computing platform. Butler et al. [7] proposed masking all network traffic via IP concealment with OpenVPN relaying to EC2 (MANTICORE). Such masking of network traffic using cloud may exploit malicious activities (e.g., cyber criminals may use such applications to remain anonymous and attempt to hide their IP address). Ferguson et al. [5] proposed using cloud infrastructure to obfuscate phishing scam analysis. Such applications allow blacklisted users to access sites without being detected and fetch contents. Zhang et al. [3] proposed design and implementation of cloud-based performance testing system for web services. Performance testing applications involve testing number of requests that can be served per second. A malicious user may use such applications to perform Denial-of-Service (DoS) attack on sites by saturating the target machine with external communications requests. Data mining using cloud computing platform is another popular concept with security issues [18–20]. According to the survey done by Rexer Analytics, 7% data miners use cloud to analyze data [21]. Malicious miners may use raw computing power provided by cloud to analyze sensitive unauthorized data and thus cause privacy violation. In a nutshell, the increasing growth of cloud computing is leading towards development of applications with higher complexity and credibility. If not monitored properly, these applications may exploit cyber crime using cloud computing platform.

Again, the cloud still remains a black box to its users. The providers control all records related to processing and users do not have access to these records. As a result, users cannot be certain about billing, malfunctioning of resources, etc. In a 2010 survey by Fujitsu Research Institute on potential cloud customers, it was found that 88% of potential cloud consumers are worried about who has access to their data, and demanded more awareness of what goes on in the back-end physical server (i.e., virtual and physical machines). This is an obstacle in the widespread growth of cloud computing [22].

Figure 1 shows an Ishikawa diagram representing some major issues (causes) that may build mistrust among users and cloud providers. These issues include

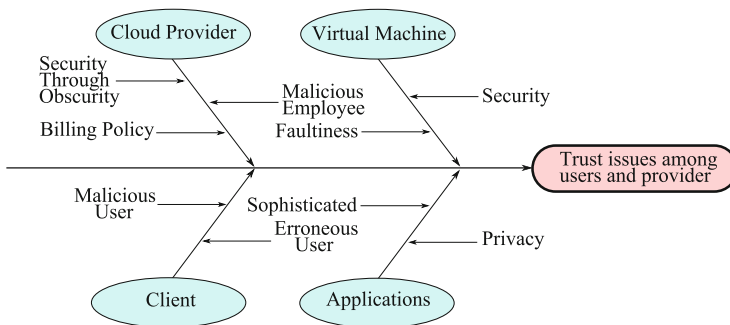


Fig. 1. Ishikawa diagram representing some trust issues in cloud

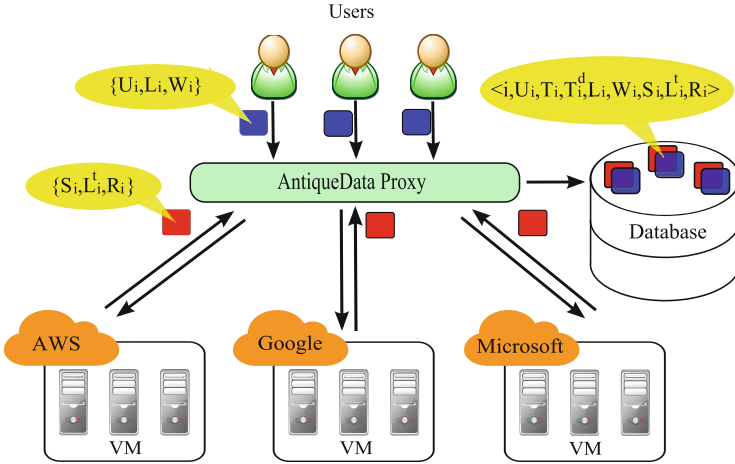


Fig. 2. System architecture

increasing use of sophisticated applications violating privacy, faultiness of virtual machines, *Security Through Obscurity* principle adopted by provider, malicious user etc.

4 AntiqueData Proxy

Our proposed system that introduces transparency among users and providers of cloud during computation or processing is shown at Fig. 2. The major component of our system is *AntiqueData proxy*, a proxy implemented at a third party server that acts as a middleware to ensure transparency. The proxy collects session-wise data from users and providers and stores it in a third party database. To use a particular resource for computational purposes, a user needs to be connected to the proxy and the proxy in turn connects to a cloud provider. Thus a user gets access to a cloud resource. That is, users can not attach themselves to cloud providers directly rather via *AntiqueData proxy*. To serve its purpose the proxy maintains information regarding providers and users. These information are maintained using three database tables dedicated to providers, clients and their relationship.

Cloud Provider Table: Each entry of the Cloud Provider Table contains information regarding a particular cloud provider. The information include the cloud provider’s name, its reliability expressed in terms of reliability levels, parameters and methods associated with pricing i.e., the cost model, its transparency expressed in terms of transparency levels etc. (Table 1). Reliability level ensures that a cloud provider meets the reliability demand of a client. Cost model is used to validate billing. Expression of transparency is given at Sect. 6.

Client Table: The entries of the Client Table correspond to information regarding all the users listed under specific clients. Client in this context refers to an

Table 1. Cloud Provider Table Sample (Partial)

Cloud provider	Reliability level	Transparency level
CP1	4	3
CP2	3	4

Table 2. Client Table Sample (Partial)

Client	Transparency level	(Virtual ID, Hierarchical level, Pass)
CL1	3	(vid1, 3, 98pX)
		(vid2, 0, m98r)
	
CL2	2	(vid1, 3, cv67)
		(vid2, 1, H7y5)
	

Table 3. Relationship Table Sample (Partial)

Cloud provider	Client	Transparency level	# of sessions	Session ID list
CP1	CL2	2	1627	{1234, ...}
CP2	CL1	3	2304	{2197, ...}

organization or a company and users refer to the members of the organization or employees of the company. The Client Table information include client’s identity, transparency level, list of triples combining a virtual ID, a hierarchy level and a password for each user belonging to the client etc. (Table 2). The hierarchy level is used for controlling access to the computational provenance records.

Relationship Table: Each entry of the Relationship Table maintains information regarding the relationship between a particular cloud provider and a particular client. The information include the cloud provider’s name, the client’s identity, the transparency level corresponding to the pair i.e., pairwise transparency level, number of sessions, list of session IDs etc. (Table 3).

The primary task of the *AntiqueData proxy* is to collect varieties of information from users and providers for each session. These information are required to perform continuous monitoring of tasks performed by users in the cloud and also to ensure proper billing by monitoring resource usage by user. The information include session id (i), virtual user id (U_i), login time (T_i), duration of the session (T_i^d), the location of the user (L_i), work description (W_i), system information (S_i), log of user task (L_i^t) and usage of resources by the user (R_i). A record is generated using the information collected from user and provider. This record is *AntiqueData record* which is represented by $\langle i, U_i, T_i, T_i^d, L_i, W_i, S_i, L_i^t, R_i \rangle$. The components of *AntiqueData record* are described below:

Session ID (i): i represents a unique session number associated with each session of computation.

Virtual User ID (U_i): U_i does not represent the real identity of a user, rather it is used to distinguish among different users within a client. This anonymization of user identity helps users to maintain their privacy. The mapping from real users within a client to their virtual identities are maintained by the client.

Login Time (T_i): T_i represents the beginning of a session. It is the unix timestamp at which a user logs into the *AntiqueData proxy* to perform processing tasks using cloud resources.

Duration of the Session (T_i^d): T_i^d represents the duration of a session. It is the lapse of time a user passes using a set of cloud resources for processing purpose.

Location of the User (L_i): L_i represents the location information of the user. This location information is collected using geolocation of IP [23] belonging to the user. The location information may include country, region/state, city, metro code/zip code, organization etc. Location information is required to solve dispute (e.g., a user to justify that his account has been compromised) among users in case of unauthorized access using user account.

Work Description (W_i): W_i represents information regarding the work done by the user using cloud resources. These information are provided by user. The information includes several fields such as type of the work (e.g., billing, mining, auditing), priority (represents the significance of the work), brief details etc.

System Information (S_i): S_i represents system information such as virtual resource status, kernel version, operating system, modules loaded, library configurations, the amount of main memory available, the memory allocated to the address space, file path of an object on the VM etc. The information about what (file operation), where (both PM and VM), and at what time a file is accessed, duplicated or transferred (which are captured within the cloud provider) are also maintained.

Task Log (L_i^t): L_i^t represents a log containing information regarding tasks (a work done by user is considered as a task sequence in this regard). The primary information is a complete workflow for the tasks done within the provider, such as complexity of tasks, whether network access is involved in a task, which blocks of a file have been modified or which records in a database table were changed or what processes and applications are run in a single machine for performing a particular task. Monitoring report of the movement of packets corresponding to a single file regarding a definite task in the network is also added to the log.

Resource Usage (R_i): R_i represents the usage of different resources by a user during a particular session. This information is required to ensure transparency regarding billing.

The *AntiqueData* records can be stored either in the third party server (where the proxy resides) or in the cloud. In both the cases, the records need to be encrypted before storing. This is done to ensure privacy of records. These records can be accessed by both cloud providers and users. But there are some hierarchical access control restrictions. A user can access any record corresponding to his own work or corresponding to work done by users of lower hierarchy. That is a user can not access works done by users of same (except his own) or higher hierarchy. Hierarchy here refers to the hierarchy level defined at the Client Table. Similarly, a cloud provider can only access his own records. The *AntiqueData* records are provided to users or providers using xml of following form.

```

<?xml version="1.0" encoding="utf-8"? >
  <SessionID>1234< /SessionID>
  <UserID>vid1< /UserID>
  <LogInTime>1376760597< /LogInTime>
  ...
  <TaskLog>
  <NoOfApplications>6< /NoOfApplications>
  ...
  < /TaskLog>
  ...
  
```

5 Communication Between Proxy and Cloud Provider

One of the biggest challenges of *AntiqueData* proxy is to collect information such as system information, task log, resource usage from providers. To do this, an entity (software, plugin, tool) needs to be present at all the virtual machines corresponding to the provider. The entity needs to serve two purposes: logging virtual machine to collect $\{S_i, L_i^t, R_i\}$ and communication with *AntiqueData* proxy.

The first purpose of the entity can be served either by implementing a logger with features such as process monitoring, resource monitoring etc. or extending existing data-centric logging mechanisms such as Flogger. Flogger [24] is a distributed file-centric VM/PM logger which monitors file operations and transfers

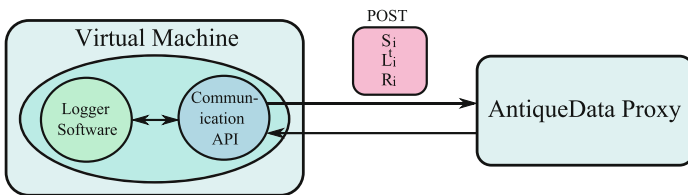


Fig. 3. Communication between proxy and provider

within the cloud. The second purpose of the entity can be served by using communication API provided by AntiqueData proxy. The API passes information to proxy using POST method (Fig. 3).

There are several issues to consider regarding the entity. The first issue to consider is the cloud provider authorization. The cloud provider may not be interested in using an API/software provided by third party. In that case, it can implement its own module that will serve the logging and communication purpose. The second issue to consider is all providers may not be interested in providing the required information. This issue is related to transparency and discussed in the next section.

6 Transparency

Transparency in cloud computing context refers to openness in communication between provider and client. It is a dual key lock that requires approval of both parties. Without cooperation of any of the parties (client or provider), the concept of transparency may fail in cloud context. We define two separate transparency parameters α and β for provider and client respectively.

The transparency parameter α for a cloud provider indicates the provider's consent in providing information. It can be defined as:

$$\alpha = \frac{\sum_{i=1}^n D_i * W_i}{\sum_{i=1}^n W_i}$$

Here, D_i is a boolean value which indicates the presence/absence of a particular information component (e.g., file path of object in VM, resource status etc.) according to the consent of cloud provider, W_i is a real value which represents the significance (weight) of the corresponding component in terms of trust establishment and n represents total number of information components. The transparency level TL of a cloud provider is defined using the value of α as: $TL = \lceil l\alpha \rceil$. The l transparency levels ($TL1, \dots, l$) represent l ranges of α 's value.

Similarly, the transparency parameter β for a client indicates the client's consent in allowing provider to monitor tasks in virtual machine. The formulation of β is similar to α except D_i here is determined by the consent of client. The transparency level of a client is determined using the value of β as: $TL = \lceil l\beta \rceil$.

The pairwise transparency parameter γ represents transparency for a pair combining a provider and a client. The formulation of γ is similar to α and β except D_i here is determined by the consent of both provider and client.

7 Evaluation

The goal of our evaluation is to (i) understand the storage and data transfer cost associated with provenance data, (ii) measure computational overhead introduced by the system and (iii) load test the system.

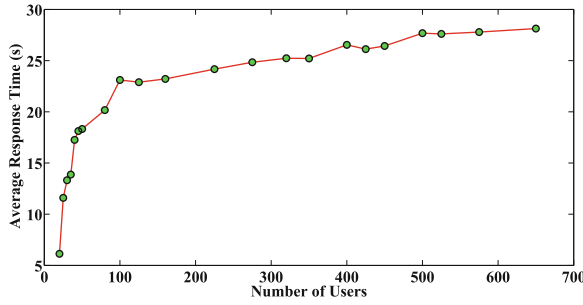


Fig. 4. Number of users vs user response time

A single session of computation generates 15 KB (maximum) provenance data. This is the upper bound for both storage and data transfer. So, the storage cost and data transfer cost introduced by the system is a modest one.

We have implemented a logger (similar to the one described in Sect. 5) using Java. In a PC having 2.5 GHz Intel Core i-5 processor with 2.88 GB usable memory running Windows 7 operating system, the logger on average uses 12458 KB memory and 5% of CPU.

We have implemented a prototype of proxy using PHP. We have tested the consistency of the proxy and have monitored its performance in terms of user response time. The results are shown in Fig. 4. We can see that with the increase in number of users, the average response time also increases drastically. To eliminate this bottleneck multiple proxies can be introduced.

8 Limitations

The proposed system establishes trust among users and providers by introducing transparency. This transparency is reflected by pairwise transparency parameter γ . If the value of γ is low for majority of the pairs (of provider and client) due to *Security Through Obscurity* principle adopted by provider or client, the availed transparency will not be as expected. As a result, the system will fail to serve its purpose of establishing trust. One of the elements of provenance record, W_i involves work description of the user. Hasty users may provide insufficient details in case of such information.

9 Future Work

In future, we would like to incorporate X.509 certificates to (i) prove a user is indeed bona fide, (ii) maintain authenticity of provenance records. A similar concept is found in grid computing. The Globus [25] security model uses X.509 certificates. We would also like to implement multiple proxy system that will eliminate the bottlenecks associated with single proxy (e.g., single point of failure).

10 Conclusion

Establishing trust among the users and cloud providers is a challenging task. Users now-a-days run varieties of complex applications on cloud computing platform. These applications are not only sophisticated in nature but they also exploit the vulnerabilities of cyber crime using cloud platform. Hence, proper monitoring of processing tasks in cloud has become a key concern. To ensure proper monitoring of computational tasks (to prevent malicious activities) and establish trust among users and providers (by introducing transparency), computational provenance records are required. In this paper, we present a middle-ware service AntiqueData proxy that serves the above purposes by maintaining computational provenance records.

References

1. Abbadi, I.M., Lyle, J.: Challenges for provenance in cloud computing. In: TaPP 2011: Proceedings of the Third USENIX Workshop on the Theory and Practice of Provenance (2011)
2. Zhang, O.Q., Kirchberg, M., Ko, R.K., Lee, B.S.: How to track your data: the case for cloud computing provenance. In: IEEE International Conference on Cloud Computing Technology and Science, pp. 446–453 (2011)
3. Zhang, L., Chen, Y., Tang, F., Ao, X.: Design and implementation of cloud-based performance testing system for web services. In: Proceedings of the 2011 6th International ICST Conference on Communications and Networking in China, CHINA-COM '11, pp. 875–880. IEEE Computer Society, Washington, DC (2011)
4. Lee, J., Cho, J., Seo, J., Shon, T., Won, D.: A novel approach to analyzing for detecting malicious network activity using a cloud computing testbed. *Mob. Networks Appl.* **18**(1), 122–128 (2013)
5. Ferguson, E., Weber, J., Hasan, R.: Cloud based content fetching: using cloud infrastructure to obfuscate phishing scam analysis. In: SERVICES, pp. 255–261 (2012)
6. Grossman, R., Gu, Y.: Data mining using high performance data clouds: experimental studies using sector and sphere. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, pp. 920–927. ACM, New York (2008)
7. Butler, P., Rhodes, A., Hasan, R.: Manticore: masking all network traffic via IP concealment with OpenVPN relaying to EC2. In: IEEE CLOUD, pp. 487–493 (2012)
8. Lyle, J., Martin, A.: Trusted computing and provenance: better together. In: TaPP '10: 2nd Workshop on the Theory and Practice of Provenance (2010)
9. Muniswamy-Reddy, K.K., Macko, P., Seltzer, M.: Making a cloud provenance-aware. In: First Workshop on Theory and Practice of Provenance, TAPP'09, pp. 12:1–12:10. USENIX Association, Berkeley (2009)
10. Muniswamy-Reddy, K.K., Macko, P., Seltzer, M.: Provenance for the cloud. In: Proceedings of the 8th USENIX Conference on File and Storage Technologies, FAST'10, pp. 15–14. USENIX Association, Berkeley (2010)
11. Muniswamy-Reddy, K.K., Seltzer, M.: Provenance as first class cloud data. *SIGOPS Oper. Syst. Rev.* **43**(4), 11–16 (2010)

12. Ko, R.K.L., Jagadpramana, P., Mowbray, M., Pearson, S., Kirchberg, M., Liang, Q., Lee, B.S.: Trustcloud: a framework for accountability and trust in cloud computing. In: SERVICES, pp. 584–588. IEEE Computer Society (2011)
13. Reilly, C.F., Naughton, J.F.: Transparently gathering provenance with provenance aware condor. In: First Workshop on Theory and Practice of Provenance, TAPP'09, pp. 13:1–13:10. USENIX Association, Berkeley (2009)
14. Abbadi, I.M.: Middleware services at cloud virtual layer. In: DSOC 2011: Proceedings of the 2nd International Workshop on Dependable Service-Oriented and Cloud computing, August 2011. IEEE Computer Society (2011)
15. Abbadi, I.M., Martin, A.: Trust in the cloud. *Inf. Secur. Tech. Rep.* **16**(3–4), 108–114 (2011)
16. Park, H., Ikeda, R., Widom, J.: RAMP: a system for capturing and tracing provenance in MapReduce workflows. *PVLDB* **4**(12), 1351–1354 (2011)
17. Akoush, S., Sohan, R., Hopper, A.: Hadoopprov: towards provenance as a first class citizen in MapReduce. In: Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance, TaPP '13, pp. 11:1–11:4. USENIX Association, Berkeley (2013)
18. Li, L., Zhang, M.: The strategy of mining association rule based on cloud computing. In: IEEE Computer Society, pp. 475–478 (2011)
19. Wang, J., Wan, J., Liu, Z., Wang, P.: Data mining of mass storage based on cloud computing. In: IEEE Computer Society, pp. 426–431 (2010)
20. Dev, H., Sen, T., Basak, M., Ali, M.E.: An approach to protect the privacy of cloud data from data mining based attacks. In: Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, SCC '12, pp. 1106–1115. IEEE Computer Society, Washington, DC (2012)
21. Karl Rexer, P.: 2010 data miner survey highlights the views of 735 data miners (2010)
22. Institute, F.R.: Personal data in the cloud: a global survey of consumer attitudes (2010)
23. Wang, Y., Burgener, D., Flores, M., Kuzmanovic, A., Huang, C.: Towards street-level client-independent IP geolocation. In: Proceeding NSDI'11 Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, NSDI'11, pp. 27–40. USENIX Association, Berkeley (2011)
24. Ryan, K.L., Ko, P.J., Lee, B.S.: Flogger: a file-centric logger for monitoring file access and transfers within cloud computing environments. Technical report, HP Cloud and Security Lab Singapore (2011)
25. Globus. <http://www.globus.org>

**Third International Workshop
on Spatial Information Modeling,
Management and Mining (SIM³)**

Monitoring Query Processing in Mobile Robot Databases

Kento Sugiura¹(✉), Arata Hayashi^{1,2,P}, Tingting Dong¹,
and Yoshiharu Ishikawa¹

¹ Graduate School of Information Science, Nagoya University, Nagoya, Japan
{sugiura,dongtt}@db.ss.is.nagoya-u.ac.jp,
ishikawa@is.nagoya-u.ac.jp
² Hitachi, Ltd., Chiyoda, Japan

Abstract. In this paper, we propose methods for monitoring query processing in mobile robot databases. We assume that a mobile robot can move based on the specified movement plan and perform sensing (e.g., temperature measurements) at the specified points. The purpose of our query processing is to reduce the total travel and measurement time while ensuring the given sensing quality requirements. We develop a framework based on an existing approach in sensor databases. Since features of mobile robots are different from those of sensor networks, we extend the former approach considering our context. We propose four algorithms for planning robot movements and compare these methods in simulation-based experiments.

1 Introduction

In recent years, research on mobile robots progressed greatly [1]. Such progress has brought about practical applications of mobile robots. For example, automatic vacuum cleaner are widely used at home and automatic driving of vehicles is advancing. Moreover, mobile robots will be widely used for indoor/outdoor environmental monitoring. Compared with sensor networks that are widely used for environmental monitoring, the use of mobile robots has some advantages:

- We can reduce the number of sensors for the monitoring. If we use a sensor network, it is necessary to deploy a sensor in each measurement point. On the other hand, a mobile robot can cover the given environment.
- We can easily change observation points. In contrast, some cost is required for restructuring a sensor network.
- We can dynamically change a monitoring plan. A mobile robot can update a plan anytime, anywhere.

Since environmental monitoring using mobile robots is promising due to the reasons above, we propose a framework to support environmental monitoring by mobile robots. Especially, we realize the framework as a *database* – it can receive a monitoring request as a query and perform robot-based monitoring as query processing. We call such a database a *mobile robot database*. There are two reasons to process a monitoring request using a mobile robot databases:

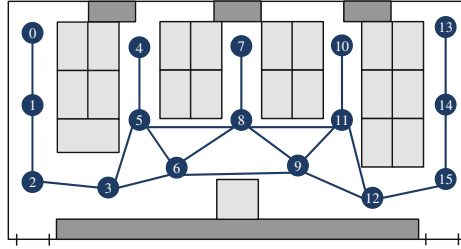


Fig. 1. Monitoring environment for a mobile robot in our laboratory

- A user can easily specify his or her monitoring request as a declarative query.
- We can optimize a monitoring plan considering the information about the sensing environment and the requirements specified in the query.

It is important for a mobile robot database to provide an effective monitoring plan because a monitoring plan affects the travel distance of the robot and time to answer the query. Figure 1 is an example of temperature monitoring in our laboratory room. Note that we use a graph to express a monitoring environment. A node indicates a point that the user is requesting the sensor value (e.g., temperature). An edge indicates that a robot can move between the nodes and we use their distance as the weight of the edge. Suppose a user wants to know the temperature of node 13 and a robot is now located at node 8. We can reduce the travel time and distance of the robot if we can estimate the temperature of node 13 using the measurement of temperature of node 10 because it is assumed that their temperatures have correlation. Such an estimating method was proposed in the context of sensor databases [2,3]. In this paper, we develop our framework by extending their proposals and propose methods for monitoring query processing in mobile robot databases.

The rest of the paper is organized as follows. First, we explain our framework, which is an extension of the existing research. Section 2 describes a summary of the existing research [2] and Sect. 3 explains the outline of query processing. In Sect. 4, we introduce four methods for selecting observation plans. Section 5 describes the settings and the results of the experiments and Sect. 6 concludes the paper.

2 Query Processing Based on Probabilistic Inference in Sensor Databases

In this section, we give an overview of the framework for processing monitoring queries, which is the basis of our proposals in this paper [2]. The approach is called *model-based sensor data acquisition* [3]. In a sensor network, a user gives a query that specifies a user's interest (e.g., temperatures of nodes), then the system answers the query by using sensors or predicting the sensor values. The main idea is to use correlations among the measurements of sensor nodes.

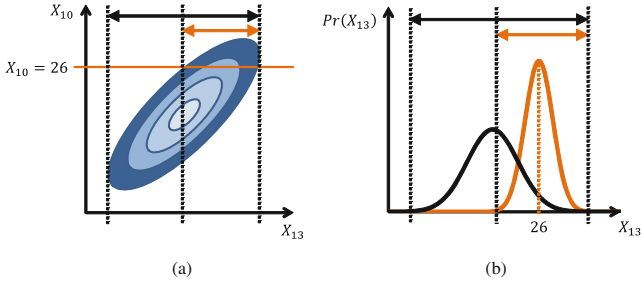


Fig. 2. Probabilistic inference on gaussian distributions

Such correlations are represented by a probabilistic model based on the *Gaussian distribution*. The number of sensor measurements is reduced using probabilistic inference on Gaussian distributions. For example, we construct a 16-dimensional Gaussian distribution for the case of Fig. 1 based on the measurements performed beforehand. Figure 2 illustrates an example of probabilistic inference between the temperature of node 10 and 13 in Fig. 1. X_{10} and X_{13} are random variables of node 10 and 13, respectively. Figure 2(a) shows an image of the two-dimensional Gaussian distribution obtained by projecting the 16-dimensional Gaussian distribution into two variables X_{10} and X_{13} , and (b) shows the one-dimensional Gaussian distribution of node 13. The black arrow in Fig. 2 illustrates the variance of the marginal distribution of X_{13} . Now suppose we get information that the temperature of node 10 is 26 degrees. The orange arrow corresponds to the variance of the posterior distribution of X_{13} after knowing the information. It means that we can predict the value of X_{13} is about 26 degrees by measuring the current value of X_{10} . In other words, we may not need to measure the temperature of node 13 if we know the temperature of node 10.

In [2], an *observation plan* P specifies the sensors that are the targets of actual measurements. Observation plan P describes how monitoring is executed in a sensor network, and consists of a set of tuples; each tuple is a pair of the target node ID and the type of the used sensor. For example, if we want to know the temperature of node 0 and the humidity of node 5 in Fig. 1, the corresponding plan is $[0, \text{temperature}], [5, \text{humidity}]$. To simplify the discussion, we assume that only one type of sensor is used for monitoring. In this case, an observation plan can be represented as a set of node IDs. The efficiency of an observation plan P is evaluated by two factors: cost and benefit. The cost $C(P)$ represents the energy for the monitoring using the target sensors. The benefit $B(P)$ is accuracy of prediction after executing the observation plan P .

The purpose of query processing is to select the best observation plan in terms of cost $C(P)$ and benefit $B(P)$. Such an efficient observation plan has the least cost under the constraint that the benefit is larger than the threshold θ defined by the user. The optimization problem can be formalized as follows:

$$\begin{aligned} & \min C(P) \\ & \text{such that } B(P) \geq \theta \end{aligned} \tag{1}$$

Since exhaustive search of the optimal plan is costly, [2] uses a greedy algorithm to select a suboptimal plan.

A summary of the query processing is as follows. Suppose a user is interested in a temperature of node 13 and a sink node is located at node 8 in Fig. 1. In order to reduce the cost, the system selects the best observation plan according to Eq. (1). Suppose a plan $\{10, 11\}$ is selected. It means that the system probably predicts the accurate temperature of node 13 by using measurements of nodes 10 and 11, and the cost of $\{10, 11\}$ is less than that of measuring the actual temperature of node 13. Therefore, the system can predict the temperature of node 13.

We develop a framework by extending this approach. Since features and requirements of mobile robots are different from those of sensor networks, we need to extend their approach for our context.

3 Query Processing Based on Probabilistic Inference in Mobile Robot Databases

In this section, we give an overview of our approach to processing monitoring queries based on probabilistic inference. Since our approach is based on [2], we explain only the extended part for mobile robot databases. First, we define two types of queries. Second, we define cost $C(P)$ in the context of mobile robot databases, and propose four algorithms for selecting efficient observation plans. Finally, we consider dynamic change of an observation plan to improve the accuracy of answers.

3.1 Query Definition

We define two types of queries in mobile robot databases following [2]. A *value query* is used to obtain the sensor value of the specified node.

Definition 1 (Value Query). *Given a node ID id , a threshold of confidence θ , and an acceptable error ϵ to a mobile robot database, a value query $VQ(id, \theta, \epsilon)$ returns a tuple $(id, temp, conf)$ which consists of the node ID, the sensor value, and the confidence score.*

A *range query* is used to confirm whether the sensor value of the specified node is within the given range.

Definition 2 (Range Query). *Given a node ID id , a threshold of confidence θ , and a sensor value range ρ to a mobile robot database, a range query $RQ(id, \theta, \rho)$ returns a tuple $(id, conf)$ which consists of the node ID and the confidence. The confidence $conf$ means the probability that temperature of node id is within range.*

3.2 Cost of an Observation Plan

In [2], energy consumption and the accuracy of prediction are used for planning. In mobile robot databases, energy consumption is not an important cost factor in general, whereas query processing time is large because a trip of a mobile robot takes much time. Therefore, we use the total execution time as the cost of an observation plan.

Total execution time of an observation plan is the time for answering the query, and consists of the travel time of a mobile robot and the observation time required to obtain sensor values in the sensing points. In this paper, we suppose that a robot waits at the initial point specified the user, so that the travel time $C_m(P)$ means the time of visiting all the nodes in the observation plan and finally returning at the initial point.

Since deciding the order of traveling between nodes in an observation plan corresponds to the traveling salesman problem, we use the suboptimal solution generated by the *nearest neighbor heuristics* and the *2-opt heuristics* [4].

The observation time T_o at each sensing point is obtained from the configuration of the sensing environment because it depends on the response time of the sensor used. Equation (2) shows the total cost $C(P)$ in our context, where $Num(P)$ is the number of nodes in the observation plan P .

$$C(P) = C_m(P) + T_o \times Num(P) \quad (2)$$

Reference [2] only uses a greedy algorithm to select an efficient observation plan. Since we consider that an observation plan is important to reduce the query execution time, we propose four algorithms for selecting an efficient observation plan: the apriori-based method, the bidirectional method, the greedy method, and the skyline-based method. In Sect. 4, we explain these methods in detail.

3.3 Dynamic Change of an Observation Plan

In [2], since an observation plan is generated by only the sink node in a sensor network, dynamic change of an observation plan is not considered. In contrast, a mobile robot can change the plan whenever it is required. We may be able to obtain a more accurate answer if we can change the plan dynamically. Therefore, we also propose a dynamic change scheme of an observation plan.

We take the simple approach that a robot selects a new observation plan when a sensor measurement is performed. Figure 3 illustrates an example. In the following, we denote the measured temperature value of node i by $temp_i$. Suppose a mobile robot is located at node 8 and a query $RQ(13, \theta, \rho)$ is given by a user, and suppose that an observation plan $\{10, 11\}$ is selected based on the algorithms described in Sect. 4. Now consider that the robot first visit node 11 based on the heuristics described in Sect. 3.2, and assume that the measured temperature $temp_{11}$ is an irregular value. In this case, it may be possible that we cannot estimate the temperature of node 13 accurately even if the temperature of node 10 is measured. In such a case, the mobile robot database changes the observation plan to improve reliability. Figure 3 illustrates the case that the old plan $\{10\}$ is updated to the new plan $\{13\}$.

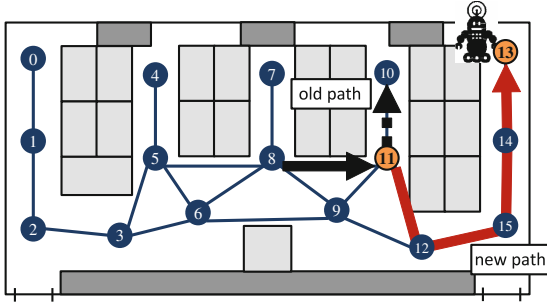


Fig. 3. Dynamic change of an observation plan

4 Algorithms for Selecting Observation Plan

In this section, we explain four algorithms for selecting an efficient observation plan: the apriori-based method (**apriori**), the bidirectional method (**bidirectional**), the greedy method (**greedy**), and the skyline-based method (**skyline**). **Apriori** and **bidirectional** can derive optimal observation plans. On the other hand, **greedy** and **skyline** select suboptimal plans.

The problem of selecting an observation plan is defined as Eq. (1). An exhaustive search over possible observation plans is impractical due to the exponential computation time. In order to select an observation plan effectively, we prune the candidate plans using the notion of monotonicity. In following subsections, we describe how each proposed method prunes the candidates and how to select a suboptimal solution in **greedy** and **skyline**.

4.1 Apriori-Based Method

The *apriori-based method* (**apriori** for short) is based on the apriori algorithm in association rule mining [5]. *Apriori* consists of the join step (generates all possible candidates) and the prune step (removes non-qualified candidates). As the join step is similar to that of the apriori algorithm, we describe how to prune the candidates in the prune step.

In order to prune the candidates, we use the *monotonicity relationships* of an observation plan. The monotonicity relationships of an observation plan P , which is a set of sensor IDs, is expressed as Eq. (3), where P_{sup} is a proper superset of P .

$$C(P_{sup}) > C(P) \tag{3}$$

We use observation plans $\{10\}$ and $\{14\}$ for Fig. 1 to explain how to prune candidates using Eq. (3). Suppose a robot is located at node 8 and a user gives a value query $VQ(13, \theta, \epsilon)$ to the mobile robot database. If the benefit of the observation plan $\{10\}$ is larger than θ , the plan $\{14\}$ is not an optimal plan because the cost of $\{14\}$ is larger than that of $\{10\}$. In addition to $\{14\}$, the cost

of supersets of $\{14\}$ is larger than that of $\{10\}$ according to Eq. (3). Therefore, we can prune the plan $\{14\}$ and the supersets of $\{14\}$ from the candidates immediately. **Apriori** can reduce a number of candidates because supersets of a pruned plan are not generated in the join step [5].

In apriori, we first initialize an optimal plan P_{opt} as the sensors specified in the query because such the plan gives an accurate answer obviously. The join step generates candidates of the optimal solution according to [5]. In the prune step, the candidate plan P is pruned when the cost of P is larger than that of P_{opt} . **Apriori** searches the remaining candidates to find an optimal solution after the prune step. When the benefit of P is larger than θ and the cost of P is less than that of P_{opt} , the plan P is the new optimal solution. For example, suppose a robot is located at node 8 and a user gives a value query $VQ(12, \theta, \epsilon)$ for Fig. 1. **Apriori** initialize an optimal plan P_{opt} as a plan $\{12\}$, and the join step generates candidates $\{\{0\}, \{1\}, \dots, \{15\}\}$. In the prune step, the candidates $\{\{0\}, \{1\}, \{2\}, \{13\}, \{14\}, \{15\}\}$ are pruned because their cost is larger than the cost of $\{12\}$ (i.e., the distances from node 8 to those nodes are larger than that of node 12). The join step generates new candidates (e.g., $\{3, 4\}, \{4, 5\}$, etc.) after **apriori** searches the remaining candidates to find an optimal solution. This process is repeated until a new candidate is not generated.

4.2 Bidirectional Method

When the threshold θ and the cost of P_{opt} is large, the computation time of **apriori** becomes larger because we cannot prune the candidates effectively. Therefore, we propose the *bidirectional method* (**bidirectional** for short) that uses Eq. (3) in addition to Eq. (4) to improve this problem.

$$B(P_{\text{sup}}) \geq B(P) \quad (4)$$

Equation (4) shows that we can prune the subsets P when the benefit of P_{sup} is less than the threshold θ . In contrast to the cost-based pruning used in **apriori**, this cost-and-benefit-based pruning effectively prunes candidates when θ is large.

Bidirectional searches the candidates combining top-down and bottom-up approaches. In this paper, we use the bottom-up search from the empty set and the top-down search from the set containing all node IDs. The bottom-up search prunes candidates using the cost of P_{opt} as in **apriori**, and the top-down search prunes candidates using θ . Note that we can calculate the number of the remaining candidates using the number of nodes in an observation plan and the information of pruned plans. Therefore, **bidirectional** can select the direction that has a smaller number of candidates, and efficiently search the candidates.

For example, suppose a graph has only four nodes. Figure 4 shows candidates of an observation plan, however \emptyset and $\{1, 2, 3, 4\}$ is omitted. **Bidirectional** selects the bottom-up search (i.e., initial targets are $\{1\}, \{2\}, \{3\}$, and $\{4\}$) because the numbers of candidates of the bottom-up and the top-down are the same. Assume that the cost of plan 1 is larger than the cost of P_{opt} (initially P_{opt} is the sensors specified in the query). In this case, $\{1\}$ and supersets are

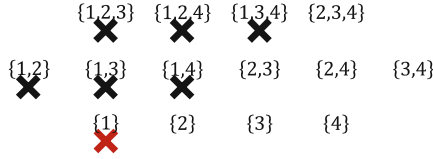


Fig. 4. An example of **bidirectional**

pruned according to Eq. (3). After the pruning, as the number of candidates of the top-down (i.e., $\{\{2, 3, 4\}\}$) is less than that of the bottom-up (i.e., $\{\{2, 3\}, \{2, 4\}, \{3, 4\}\}$), **bidirectional** next selects the top-down search. **Bidirectional** repeats this process until all the candidates are searched.

4.3 Greedy Method

The *greedy method* (**greedy** for short) finds a suboptimal observation plan by selecting a partial solution in a heuristic manner. Although the query processing method in [2] is based on a similar greedy-based idea, **greedy** in this paper is different due to the use of the monotonicity relationships. The way of pruning is similar to that of **apriori**, so that we omit the detail in this section.

In **greedy**, we first initialize a partial solution P as an empty set and a suboptimal solution P_{opt} as the sensors specified in the query. Then we add each node ID $i \notin P$ to the partial solution P and compute the benefit and the cost of the plan $P \cup i$ respectively. When $B(P \cup i)$ is larger than θ and $C(P \cup i)$ is less than $C(P_{opt})$, the plan $P \cup i$ is the new suboptimal solution. In each iteration step, we select a new partial solution from candidates $P \cup i$ based on the effectiveness of the prediction. In this paper, the effectiveness of the prediction is defined by $B(P)/C(P)$. We repeat this process until new candidates are not generated.

4.4 Skyline-Based Method

Skyline queries select a set of objects that are not dominated by the others in a database [6]. In this paper, we apply the notion of skyline query to our context. A plan belongs to the skyline if it is not dominated by any other plans in terms of cost and benefit.

The skyline-based method (**skyline** for short) constructs a skyline for partial solutions. We repeatedly add a node to the plan in the skyline to generate a new candidate from the skyline objects. For example, suppose the current skyline (a set of partial solutions) is $\{\emptyset, \{1\}\}$. If we add node 2 to the skyline, we get new candidates $\{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$. Suppose the benefit of $\{1\}$ is less than that of $\{2\}$ and the cost of $\{1\}$ is larger than that of $\{2\}$. In this case, **skyline** prunes the plan $\{1\}$ because $\{1\}$ is dominated by $\{2\}$. We repeat this process until all nodes are added to the skyline. As in **apriori**, **skyline** can also reduce computation time using the monotonicity relationships.

5 Experiments

5.1 Setup of Experiments

In this section, we perform experiments for evaluating the proposed methods based on simulations. We prepare two simulation environments. The first simulation environment is for measuring plan computation time. We utilize five graphs with $d = 10, 20, 30, 40,$ and 50 nodes, respectively. We arrange nodes in a grid form and link between the neighboring nodes. We set the travel time between each edge to three seconds. We construct a probabilistic model represented as a d -dimensional Gaussian distribution, however the parameters of the Gaussian distribution are set manually. The second simulation environment is for measuring the efficiency of observation plans. We use Fig. 1 as the underlying graph structure. Travel time between two seconds to three seconds is assigned to each edge depending on the real distance. We constructed a 16-dimensional Gaussian distribution from the actual temperature measurements in our laboratory room. In the first and second simulation environment, the observation time at each observation point is set to one second.

5.2 Evaluation of Algorithms

Computation Time. First, we evaluate the four algorithms using the first simulation environment. We only show the results of value queries because there is not much difference in the tendency between the results of value queries and those of range queries. Figure 5 shows the computation time of each algorithm. Note that we omitted some points in Fig. 5 if their computation times are larger than one second. In Fig. 5(a)–(c) correspond to the case of $(\theta = 0.9, \epsilon = 0.5)$, $(\theta = 0.95, \epsilon = 0.3)$, and $(\theta = 0.99, \epsilon = 0.1)$, respectively.

Apriori can reduce the computation time in Fig. 5(a) because the use of the monotonicity relationship can prune candidates efficiently. In Fig. 5(b) and (c), however, the computation time of **apriori** is large. The reason is that the use of the cost-based pruning cannot prune candidates effectively when θ is large or ϵ is small. On the other hand, **bidirectional** can reduce the computation time in

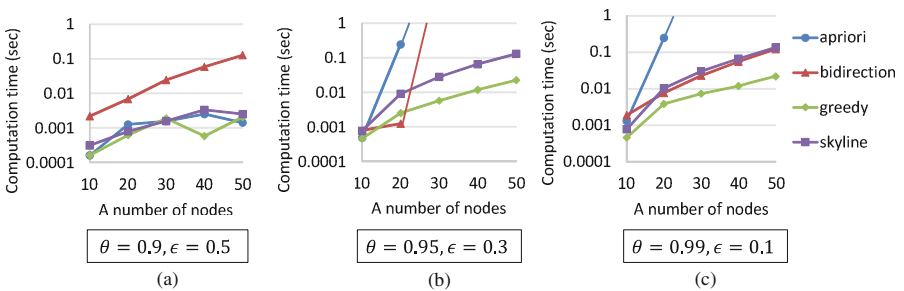


Fig. 5. Computation time of proposed algorithms

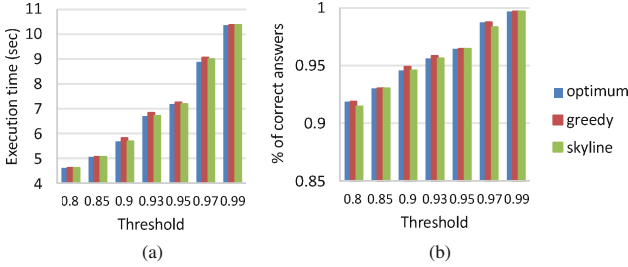


Fig. 6. Comparison of execution time and percentage of correct answers

Fig. 5(c) by using the cost-and-benefit-based pruning. However, its computation time is large in Fig. 5(b). This result means that pruning by the monotonicity relationship is not always effective. In contrast, since **greedy** and **skyline** can prune candidates by selecting partial solutions, they can reduce the computation time in all the cases. In addition, they can reduce the computation time further if the cost-based pruning is effective as shown in Fig. 5(a).

Quality of Observation Plans. Next, we evaluate the efficiency and the quality of the observation plan selected by each algorithm in the second simulation environment. Since both of **apriori** and **bidirectional** select the optimal solution, we unify them and represent as **optimum**. We only show the results of value queries $VQ(id, \theta, 0.2)$ because there is not much difference in other results including the range queries. Figure 6 shows (a) the execution time of each observation plan and (b) the percentage of correct answers. We treat a predicted value as a correct answer if the predicted value is within the acceptable error range ϵ from the real value. Note that the percentage of correct answers does not reach 100% because the algorithms consider the threshold θ given by the user — the algorithms terminate the search process when the user’s requirement is satisfied.

Figure 6 shows that the result of each algorithm is almost similar. There are two reasons. First, we used relatively small and simple graphs in the experiments. Since it is easy to derive an optimal solution when a graph is small and simple, **greedy** and **skyline** can easily generate effective observation plans. Second, we consider simple query types such as value queries and range queries. Since the queries considered in this paper are for predicting the result for only one node, it is not difficult to process them in contrast to complex queries.

5.3 Analysis of each Type of Queries

In this subsection, we show the experimental results only for **greedy** since it performs well among the four methods in the current experimental settings. Figure 7 shows the experimental result for value queries: (a) presents the execution time of observation plans, and (b) shows the percentages of correct answers. For a comparison, we show the execution time of the naive method as a black

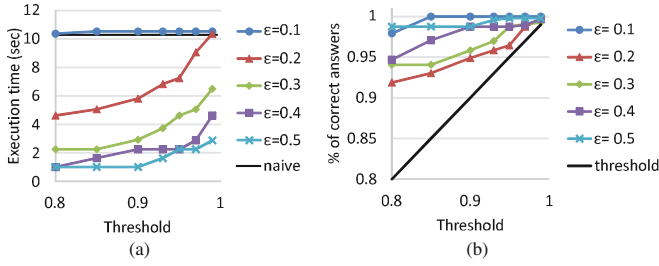


Fig. 7. Analysis of value queries

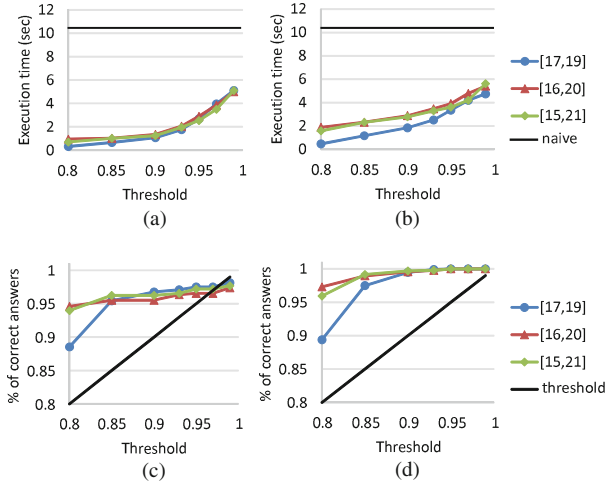


Fig. 8. Analysis of range queries without/with dynamic plan change

line in Fig. 7(a). In the naive method, a robot always moves to the specified observation point and measures a temperature. In Fig. 7(a), the cost is small when the threshold θ is small or the acceptable error ϵ is large. In other words, this result demonstrates that a mobile robot database can reduce the cost of query processing when the user does not need an accurate answer. Figure 7(b) shows that the user's accuracy requirement (denoted by the black line) is always satisfied in this experiment. Therefore, we can say that our approach can answer queries efficiently and accurately.

Next, we evaluate the effectiveness of dynamic change of an observation plan. Figure 8 shows the execution time and the percentages of correct answers in for range queries. Figure 8(a) and (c) are for the case without change, while (b) and (d) are for the case with dynamic change. In Fig 8(a) and (b), it is shown that dynamic change increases of the cost of observation plans. The reason is that a new observation plan requires additional observations. However, the

additional cost is not large. From Fig. 8(c) and (d), we can observe that dynamic change improves accuracy of answers. In particular, note that we can improve the accuracy when the threshold θ is large.

6 Conclusions

In this paper, we proposed query processing methods for mobile robot-based sensing. Extending the idea of an existing method in sensor databases, we developed adaptive methods in the context of mobile robot databases. We focused on the selection methods of observation plans, which were not elaborated in the former work, and we proposed four selection methods, **apriori**, **bidirectional**, **greedy**, and **skyline**. The proposed four selection methods are evaluated based on the experiments. Our future work includes refinement and re-evaluation of the selection methods of observation plans, support of various observation tasks, update of probabilistic models, coping with uncertainty of sensor values and robot movements, handling error and outliers, and implementation and evaluation using a moving robot in real environments.

Acknowledgment. This research is supported by the FIRST program, Japan, KAKENHI (25280039, 23650047), and MEXT COI STREAM Project.

References

1. Lenchner, J., Isci, C., Kephart, J.O., Mansley, C., Connell, J., McIntosh, S.: Towards data center self-diagnosis using a mobile robot. In: Proceedings of International Conference on Autonomic Computing (ICAC), pp. 81–90. ACM (2011)
2. Deshpande, A., Guestrin, C., Madden, S.R., Hellerstein, J.M., Hong, W.: Model-based approximate querying in sensor networks. VLDB J. **14**(4), 417–443 (2005)
3. Sathe, S., Papaioannou, T.G., Jeung, H., Aberer, K.: A survey of model-based sensor data acquisition and management. In: Aggarwal, C.C. (ed.) *Managing and Mining Sensor Data*, pp. 9–50. Springer, New York (2013)
4. Yamamoto, Y., Kubo, M.: *Invitation to the Traveling Salesman Problem* (in Japanese). Asakura Publishing, Tokyo (1997)
5. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of VLDB, pp. 487–499 (1994)
6. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of ICDE, pp. 421–430 (2001)

Efficiently Evaluating Range-Constrained Spatial Keyword Query on Road Networks

Wengen Li¹, Jihong Guan¹(✉), and Shuigeng Zhou²

¹ Department of Computer Science and Technology,
Tongji University, Shanghai, China
lwengen@gmail.com, jhguan@tongji.edu.cn

² School of Computer Science, Fudan University, Shanghai, China
sgzhou@fudan.edu.cn

Abstract. With the rapid development of geo-positioning technologies, spatial information retrieval plays an important role in a wide spectrum of applications, e.g., online maps and location-based services. Specifically, spatial keyword query (*SK* query), considering both spatial proximity to the query location and textual relevance to the query keywords, is now a hot research topic in database community. This paper addresses a specific type of *SK* query, termed *range constrained spatial keyword query* (*RC-SK* query), which searches for all the POIs (points of interest) whose textual description is relevant to the query keywords within a specified area. Though *RC-SK* query has received extensive studies in Euclidean space, little is done to deal with it on road networks. In this paper, alternative approaches with different indexing strategies are proposed to solve this problem. Extensive empirical studies on multiple real datasets demonstrate the efficiency of these proposed approaches.

Keywords: Range-constrained spatial keyword query · Road networks · Hierarchy indexing

1 Introduction

The rapid development of techniques for both geo-positioning and mobile communication has made location aware query a necessary part in many applications. In this paper, we consider a specific type of such query called *range constrained spatial keyword query*, *RC-SK* query for short, on road networks. Concretely, a *RC-SK* query, specified with a spatial location and a set of query keywords, is targeted for finding all the POIs whose textual relevance to the query keywords is larger than a specified threshold and location is within a specified distance to the query location. For instance, a visitor poses a query to search for all the banks offering exchange service within 2km from his or her current location. Here, a bank is a POI with a spatial location (e.g., longitude and latitude) and a piece of textual description about the services it offers.

Actually, there have been some works on *RC-SK* query in Euclidean space. In reality, however, people's trajectories are usually constrained by road networks.

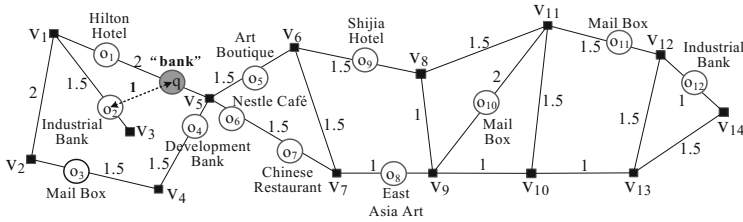


Fig. 1. An example of *RC-SK* query. Here, $dis(q, o_2) = 2.5$.

Figure 1 illustrates an example of *RC-SK* query on a small road network which has 14 vertices $v_i (i = 1, \dots, 14)$ and 12 POIs $o_j (j = 1, \dots, 12)$ denoted as black grids and circles, respectively. Each edge is labeled with its length. The query q , illustrated by a grey filled circle, searches for all the banks within 2 km. In Euclidean space, both o_2 and o_4 will be returned as the result. However, we cannot reach o_2 within 2 km along the road network. Hence, it is more practicable to conduct *RC-SK* query based on network distance than Euclidean distance.

However, conducting *RC-SK* query on road networks is much more challenging than that in Euclidean space because the shortest path between query location and any candidate POI should be computed. Especially for a larger query range, we need to enumerate many POIs and compute their shortest distances to the query location.

In this paper, three approaches are proposed to deal with *RC-SK* query on road networks. The first one is *expansion-based approach (EA)*, a baseline approach, which traverses the road network from the query location with the same flavor as Dijkstra’s algorithm. The second approach is *Euclidean heuristic approach (EHA)* which is an improvement of *EA* approach and employs Euclidean heuristic to accelerate query processing. As both *EA* and *EHA* have to traverse the road network vertex by vertex, they are inefficient for a large query range. To solve this problem, the third approach called *Rnet Hierarchy [1] based approach (RHA)* is proposed. *RHA* partitions the whole road network into a group of interconnected subnets and organizes them in a hierarchy structure, which greatly improves the query efficiency.

The remainder of this paper is organized as follows. Section 2 reviews the related work and Sect. 3 formally defines the problem. Sections 4–6 elaborate *EA*, *EHA* and *RHA*, respectively. Section 7 empirically evaluates the proposed approaches and Sect. 8 concludes the paper.

2 Related Work

Generally, there are two types of widely used spatial keyword queries [7], i.e., top- k spatial keyword query (top- k *SK* query) [13, 14], searching for the k best POIs based on both spatial proximity and textual relevance, and range-constrained spatial keyword query (*RC-SK* query) [8, 16], searching for all the POIs satisfying the required textual relevance within a specified area.

Table 1. Hybrid indices.

References	Hybrid index	Spatial index	Textual index
[15]	IR2-tree	R-tree	Signature file
[11]	IR-tree	R-tree	Inverted file
[8, 16]	KR*-tree	R*-tree	Inverted file
[18]	bR*-tree	R*-tree	Bitmap

During the past decade, *RC-SK* query has received extensive studies in Euclidean space. The original solution [8] to *RC-SK* query retrieves all the POIs within the query range area and conducts a detailed examination on these POIs based on their textual relevance, which is inefficient for large-size datasets. To solve this problem, previous works try to embed traditional textual indices, such as inverted file and signature file [12] into an R-tree [9], a widely-used index structure for multi-dimensional data, or its variants. Table 1 shows major hybrid schemes that merge text index and spatial index.

Almost all the proposed approaches for *RC-SK* query in Euclidean space are based on these hybrid indices. During query processing, spatial proximity and textual relevance are computed simultaneously, which make it efficient to prune irrelevant branches as soon as possible. However, all these index structures and processing algorithms are devised for spatial keyword queries in Euclidean space and cannot be directly used for *RC-SK* query on road networks.

In addition, Rocha-Junior et al. [10] proposed several efficient approaches to address top-*k* *SK* query on road networks, which is the most related work to ours. The framework of their overlay approach is similar to that of our *RHA*. However, both the partition strategy and index structure of *RHA* are different from those of the overlay approach. More importantly, we aim at evaluating *RC-SK* query instead of top-*k* *SK* query on road networks.

3 Problem Statement

Formally, a road network is represented as an undirected graph $G = (V, E)$, where V and E are the sets of vertices and edges, respectively. Each vertex $v \in V$ represents a road intersection or a road endpoint; each edge $e_{i,j} \in E (i \neq j)$ represents the road segment connecting v_i and v_j and its length is denoted as $|e_{i,j}|$. The distance between two vertices u and v , $dis(u, v)$, is the length of the shortest path between them.

A POI o is represented as $o = (l, e, d, K)$, where $o.l$ is the spatial location consisting of longitude and latitude, $o.e$ is the edge on which o resides, $o.d$ is the distance from $o.l$ to the beginning vertex of $o.e$, and $o.K$ is a set of keywords which describe the details of o .

A *RC-SK* query q over G is defined as $q = (l, K, \tau, r)$, where $q.l$ is the query location, $q.K$ is a set of query keywords, $q.\tau \in (0, 1]$ is a predefined textual

relevance threshold and $q.r$ specifies the query range. The answers to q are the set of POIs on G such that each of them satisfies

$$dis(o.l, q.l) \leq q.r \wedge \theta(o.K, q.K) \geq q.\tau$$

where $dis(o.l, q.l)$ is the distance between $o.l$ and $q.l$, $\theta(o.K, q.K)$ is the textual relevance between $o.K$ and $q.K$ and defined as follows [3].

$$\theta(o.K, q.K) = \frac{\sum_{k \in q.K} w_{k,o.K} \cdot w_{k,q.K}}{\sqrt{\sum_{k \in o.K} (w_{k,o.K})^2 \cdot \sum_{k \in q.K} (w_{k,q.K})^2}} \tag{1}$$

where $w_{k,o.K} = 1 + \ln(f_{k,o.K})$, $f_{k,o.K}$ is the occurrences of query keyword $k \in q.K$ in $o.K$; $w_{k,q.K} = \ln(1 + \frac{|P|}{df_k})$, where $|P|$ is the number of POIs on G , df_k is the number of POIs containing k .

Although the definition above and the following approaches are based on undirected road networks, they can be extended to directed road networks with only a little modification.

4 The Expansion-Based Approach

This baseline approach processes $RC-SK$ query in an expansion fashion like Dijkstra’s algorithm.

4.1 Index Structure

An R^* -tree [2] is employed to index all edges in E as illustrated in Fig. 2 where each edge is represented as a minimum bounding rectangle that totally encloses it. With the help of the R^* -tree, the edge on which $q.l$ resides can be quickly determined with a spatial point query.

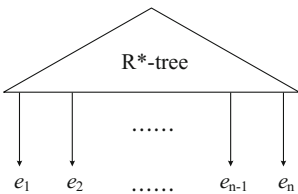


Fig. 2. R^* -tree for edges.

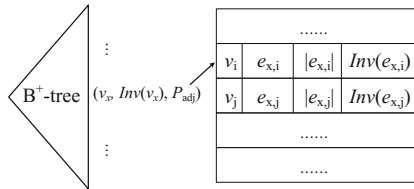


Fig. 3. Index for road networks.

Meanwhile, a B^+ -tree is used to index the modified adjacent lists as shown in Fig. 3 which keeps the connectivity of G . Each entry in leaf node is a triple

$(v_x, Inv(v_x), P_{adj})$, where v_x is a vertex, $Inv(v_x)$ is a pointer to the inverted file [3] (called *vertex inverted file*) which covers all the POIs on v_x 's adjacent edges. P_{adj} is another pointer pointing to the adjacent list of v_x . Each entry in the adjacent list is a quadruple $(v_i, e_{x,i}, |e_{x,i}|, Inv(e_{x,i}))$, where v_i is a neighbor vertex of v_x , $e_{x,i}$ is the edge between v_x and v_i with length $|e_{x,i}|$, and $Inv(e_{x,i})$ is a pointer to the inverted file (called *edge inverted file*) covering all POIs on $e_{x,i}$.

4.2 Query Processing

Initially, a priority queue U is created to store visited vertices during expansion based on their network distances to $q.l$. Meanwhile, a list L is created to store query results. First, the edge $e_{i,j}$ on which $q.l$ resides is located by using the R^* -tree built for edges. Then a verification is conducted on $e_{i,j}$ to check whether it contains any POI whose textual relevance to $q.K$ is larger than $q.\tau$. Next, both v_i and v_j are inserted into U with their distances to $q.l$. By obtaining vertices from U and checking their adjacent edges, we can traverse all edges within $q.r$ from $q.l$ and verify them in the same way as we do for $e_{i,j}$. During the verification, POIs satisfying the textual relevance threshold are added to L .

Consider the query q over the road network in Fig. 1, where $q.l$ is the filled circle, $q.K = \text{"bank"}$, $q.r = 2$. For presentation simplicity, we ignore $q.\tau$ and only require that each returned POI contains $q.K$. First, we find that $q.l$ is located on $e_{1,5}$ which has no desirable POIs. Then, v_1 and v_5 are inserted into U with $(v_5, 0.5)$ and $(v_1, 1.5)$, respectively. Here, we assume $|q.l, v_1| = 1.5$ and $|q.l, v_5| = 0.5$. Next, we get $(v_5, 0.5)$ from U . By checking the inverted file $Inv(v_5)$ for v_5 , we find that $e_{5,4}$ contains query keyword "bank". Then $Inv(e_{5,4})$ is checked and o_4 is inserted into L . As the distance from $q.l$ to v_4 is 2, we do not search beyond v_4 , which is same for v_6 and v_7 . Following that, we get $(v_1, 1.5)$ from U and its adjacent edges $e_{1,2}$ and $e_{1,3}$ are checked, and no POIs (assume $|o_2, v_1| = 1$, we have $dis(q.l, o_2) = 2.5 > 2$) are inserted into L . Now U is empty and the algorithm terminates. Finally, we get the query result $L = \langle o_4 \rangle$.

5 The Euclidean Heuristic Approach

EA is efficient enough for *RC-SK* queries with a small $q.r$. However, if $q.r$ is very large and numerous edges and POIs are covered, *EA* will incur a considerable overhead to obtain all desirable POIs because it has to verify the inverted files of all relevant edges (containing any query keyword). In general, however, a large portion of keywords cover only a small number of POIs. In such a situation, to check the inverted files for all relevant edges is unnecessary. To overcome this drawback, we propose the *Euclidean heuristic approach (EHA)*.

Intuitively, Euclidean distance between any two vertices on a road network is always smaller, if not equal to, than the network distance between them. Therefore, if a POI belongs to the query result based on network distance, then it must be in the query result based on Euclidean distance. Based on this observation, we propose the *EHA* to first retrieve all the candidate POIs according to any

state-of-the-art Euclidean distance based approach. Here we employ IR-tree [11] which augments each node of the R-tree with an inverted file. The set of edges having candidate POIs (satisfying textual relevance threshold) is recorded as E_q . During the expansion process, we avoid verifying the inverted file of a particular edge by checking whether it is contained in E_q . Thus, edges containing no desirable POIs are filtered. *EHA* is implemented based on *EA* by adding a *SK* query on the IR-tree at the beginning and an edge set check during expansion. We omit the detail here due to space limit.

6 The Rnet Hierarchy-Based Approach

Essentially, both *EA* and *EHA* expand from one vertex to another on G within $q.r$, which makes it very expensive to evaluate *RC-SK* queries with a large $q.r$. To solve this problem, we introduce the Rnet Hierarchy [1] to index road networks and further propose *Rnet Hierarchy-based approach*, *RHA* for short.

6.1 Indexing Structure

Rnet Hierarchy partitions a road network into connected subnets called Rnets (regional nets) and organizes them in a hierarchy structure as depicted in Fig. 4. An Rnet R is defined as (V_R, E_R, B_R) where V_R , E_R , and B_R are the sets of vertices, edges and border vertices of R , respectively. B_R are the vertices shared by two or more Rnets (e.g., v_5). In Fig. 4, there are four Rnets R_{11} , R_{12} , R_{21} and R_{22} at level 1 and two larger Rnets R_1 and R_2 at level 2. R_1 encloses R_{11} and R_{12} , and R_2 encloses R_{21} and R_{22} . Level 0 is the original road network. Therefore, a road network is organized as a group of connected Rnets at each level.

In order to skip over Rnets, *shortcut* is introduced. A *shortcut* is the shortest path between two border vertices of an Rnet, e.g., $SP(v_5, v_9)$. With the help of shortcuts on different levels, a search expands quickly with different step sizes. Here, a challenge is how to partition a road network into a group of Rnets with a minimum number of border vertices. In this paper, we first consider the *equal-size partition* [1], which adopts the geometric approach [4] and *KL* algorithm [5], to partition the whole road network into Rnets of the similar size. *Equal-size*

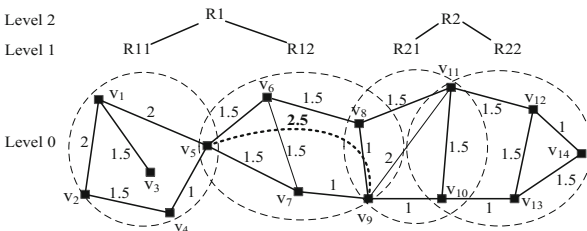


Fig. 4. An example of Rnet Hierarchy of a road network.

partition first partitions the whole road network into two parts with almost the same number of edges, and then tunes them by exchanging edges to reduce the border vertices. By doing this recursively, G is partitioned into a set of Rnets with almost the same size.

However, the partition method above ignores the road network’s semantics. In reality, POIs on road networks are often clustered [6] in some hot areas like commercial centers. For example, area around v_5 in Fig. 1 has more POIs than areas around other vertices (e.g., v_{13}). Figures 5 and 6 display the POI distribution on the road network of London. Obviously, most vertices have less than 5 POIs (POIs residing on the edges adjacent to the vertex). Accordingly, we consider partitioning a road network based on the distribution of POIs, i.e., *distribution-aware partition* and aim to partition as many as POIs into the same Rnet. To this end, we first collect all the vertices with more POIs than a specified parameter (e.g., 10) and then merge these vertices to form larger areas based on their spatial proximity until a Rnet is generated. Meanwhile, the other areas are partitioned by using the *equal-size partition*.

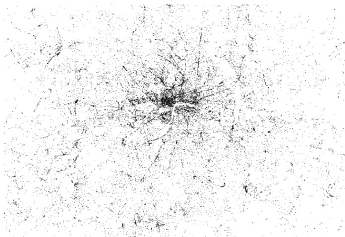


Fig. 5. POI distribution of London

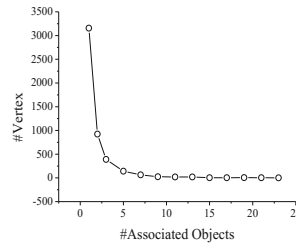


Fig. 6. POI distribution statistics

Rnet Hierarchy is organized using a B^+ -tree as illustrated in Fig. 7. The B^+ -tree indexes all vertices based on their identifiers and each entry in a leaf node points to an adjacent list or a *hierarchy tree*. Concretely, if a vertex v_i (e.g., v_2) is not a border vertex, the entry for v_i has a pointer pointing to an adjacent list just as EA . Otherwise, the entry for v_i has a pointer pointing to a *hierarchy tree* which records the organization of all the Rnets associated with v_i at different levels. For example, v_5 is a border vertex of R_{11} and R_{12} , and it has a *hierarchy tree* T_{v_5} . The root node of T_{v_5} contains two entries $E_{R_{11}}$ (for R_{11}) and $E_{R_{12}}$ (for R_{12}) and a pointer pointing to the inverted file for them. In a *hierarchy tree*, each entry in the intermediate nodes also stores all the shortcuts within the corresponding Rnet while each entry in the leaf nodes points to the adjacent list of the border vertex.

In addition, to utilize Euclidean heuristic to quickly find out those Rnets that contain desirable POIs, we also try to organize all Rnets at different levels into a variant IR-tree as illustrated in Fig. 8.

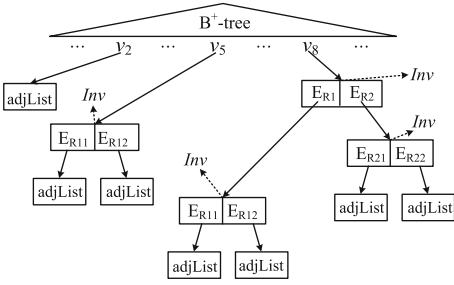


Fig. 7. Index for the Rnet Hierarchy

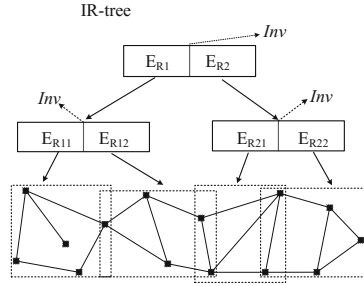


Fig. 8. IR-tree for Rnets

6.2 Query Processing

Given a query q , RHA searches for POIs in an expanding fashion as detailed in Algorithm 1. If a vertex v_i is not a border vertex, it proceeds in the same way as EA . Otherwise, the hierarchy tree T_{v_i} of v_i is checked. First, we examine the inverted file for the root node of T_{v_i} to check whether it contains desirable POIs. If not, we skip over the entire Rnet through the shortcuts without checking its inner edges. Otherwise, we check its child nodes to further retrieve desirable POIs.

Algorithm 1. The Rnet Hierarchy based Approach

Input: $G = (V, E)$, $q = (q.l, q.K, q.\tau, q.r)$

Output: Any POI o such that $dis(o.l, q.l) \leq q.r \wedge \theta(o.K, q.K) \geq q.\tau$

- 1: $U = \text{newPriorityQueue}()$; // used to store visited vertices during processing
 - 2: $L = \text{newList}()$; // used to store final result
 - 3: $e_q = \text{locateEdge}(q.l)$;
 - 4: $U.\text{enqueue}(e_q.v_1, |q.l, e_q.v_1|)$;
 - 5: $U.\text{enqueue}(e_q.v_2, |q.l, e_q.v_2|)$;
 - 6: $\text{checkEdge}(e_q)$; // check $Inv(e_q)$
 - 7: **while not** $U.\text{isEmpty}()$ **do**
 - 8: $v = U.\text{Dequeue}()$;
 - 9: **if not** $v.\text{isBorderVertex}()$ // v is not a border vertex
 - 10: **for** each adjacent edge e of v
 - 11: **if** e contains $q.K$
 - 12: $\text{checkEdge}(e)$;
 - 13: **if** $(v.\text{currentDistance} + e.\text{length} < q.r)$
 - 14: $U.\text{enqueue}(e.\text{anotherVertex})$;
 - 15: **else** // v is a border vertex
 - 16: $\text{check } v.\text{HierarchyTree}$;
 - 17: **return** L ;
-

Table 2. Statistics of datasets.

Attributes	Dublin	London	Australia	BritishIsles
#vertices	62,975	209,406	1,223,171	3,760,213
#edges	82,730	282,267	1,682,182	4,865,094
#POIs	5,297	34,341	70,064	300,891
#keywords	15,216	97,824	193,106	842,369
#distinct keywords	3,563	12,522	18,789	60,558

Table 3. Parameters in experiments.

Parameters	Values
$q.r$	1, 3, 5 , 7, 9 (km)
$q.\tau$	0.3, 0.5 , 0.7, 0.9, Boolean
$ q.K $	1, 2 , 3, 4, 5
Datasets	Dublin, London, Australia, British Isles
Partition strategy	Equal-size partition , <i>Distribution-aware partition</i>

For example, we consider the same query in Fig. 1 with a larger $q.r = 3$. First, we verify the edge $e_{1,5}$ on which $q.l$ resides. Then, $(v_5, 0.5)$ and $(v_1, 1.5)$ are inserted into U . Next, we obtain $(v_5, 0.5)$ from U . As v_5 is a border vertex, we evaluate R_{11} and R_{12} . R_{11} contains *bank* and a detailed examination is conducted. Then $e_{5,4}$ is checked and o_4 is added to L . Meanwhile, $(v_4, 2)$ is inserted into U . R_{12} contains no desirable POIs and its shortcuts from v_5 to v_8 and v_9 go beyond the query range. Then, we get $(v_1, 1.5)$ and $(v_4, 2)$ in order, and no more POIs are added to L . Finally, the result $L = \langle o_4 \rangle$ is returned.

As an expected improvement, we employ the IR-tree in Fig. 8 to accelerate query processing by getting all Rnets containing desirable POIs in advance, which avoids checking the inverted file for each Rnet.

7 Experimental Evaluation

7.1 Setup

The performance of proposed approaches is evaluated on four real datasets¹ which are road networks of Dublin, London, Australia, and British Isles, respectively. Table 2 displays some statistics of the four datasets. For each dataset, we randomly generate 500 locations within the road network area as query locations and 500 sets of keywords of size 1, 2, 3, 4, and 5, separately. Table 3 lists the parameters used in the experiments and marks their default values in bold. Experiments run on a PC with a 3.1 GHz Intel processor and a 4 GB RAM. The index structures of the three approaches are disk-resident and the buffer is set at 4 MB.

¹ <http://www.idi.ntnu.no/~joao/publications/EDBT2012/>

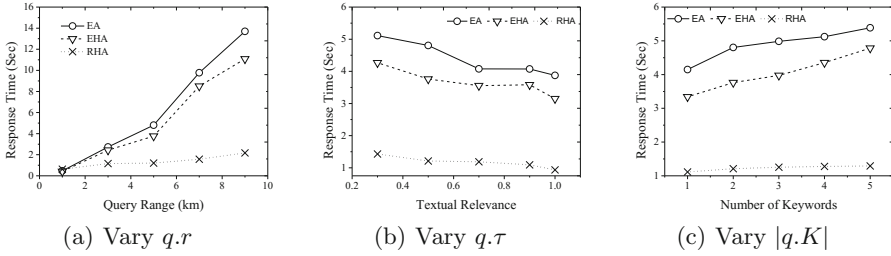


Fig. 9. Response time while varying $q.r$, $q.\tau$, and $|q.K|$

7.2 Experimental Results

Varying $q.r$. Figure 9(a) illustrates the response time while varying $q.r$. *EA* performs well for queries with a small $q.r$. With the increase of $q.r$, however, the response time increases rapidly because *EA* has to expand all the edges within $q.r$ from $q.l$. Compared to *EA*, *EHA* performs better because it avoids checking the inverted file for every edge that contains any query keyword. This differs from the conclusion in [17] that network expansion algorithms performs better than Euclidean distance heuristic based algorithms, because *EHA* just uses Euclidean heuristic to avoid unnecessary edge examination. However, both *EA* and *EHA* expand vertex by vertex, which inevitably incurs a high overhead. *RHA* performs much better than *EA* and *EHA* because it bypasses the Rnets containing no desirable POIs and avoids a detailed examination on their inner edges. Additionally, with the help of different layers and different size of Rnets, *RHA* works well with different $q.r$.

Varying $q.\tau$. Figure 9(b) shows the response time while varying the textual relevance $q.\tau$. A smaller $q.\tau$ covers more POIs and it consumes more time to evaluate these POIs. Both *EA* and *EHA* cost more than 3s to evaluate a *RC-SK* query while varying $q.\tau$ from 0.3 to 0.9. As for *RHA*, only about one second is required. Because no textual relevance is computed, Boolean query ($q.\tau = 1$)² takes less time than other queries.

Varying $|q.K|$. Figure 9(c) presents the response time while varying the number of query keywords $|q.K|$. With the increase of $|q.K|$, both *EA* and *EHA* have to verify more POIs relevant to $q.K$, which leads to an increase in query time. Although *RHA* also needs more time to evaluates *RC-SK* queries with more keywords, the increase of response time is very slow because *RHA* computes the textual relevance between an Rnet and $q.K$, and there is no detailed examination if the relevance is less than $q.\tau$. Therefore, increasing query keywords affects only a small portion of Rnets and the other Rnets are still skipped through shortcuts.

² This value is just a label for Boolean query instead of a textual relevance value.

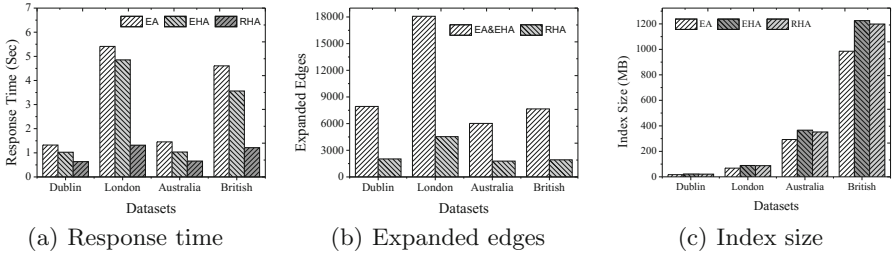


Fig. 10. Response time, expanded edges and index size for different datasets

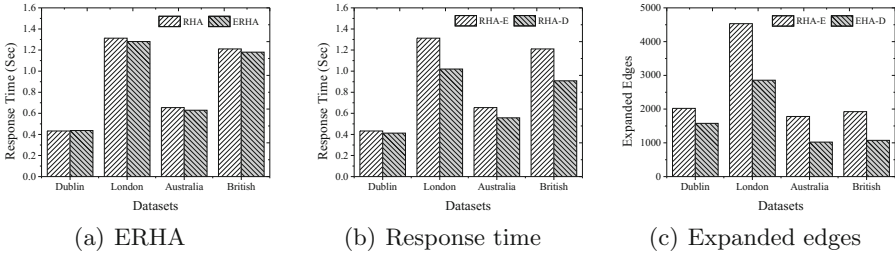


Fig. 11. Response time using ERHA, response time and expanded edges while varying partition strategies

Different Datasets. Figure 10(a) shows the response time for different datasets. *RHA* evaluates *RC-SK* queries on the four datasets of different size in about one second. We can find that the response time on Australia network is smaller than that on London network. This is because the London network is denser than Australia network. In general, given the same query range, London network usually has more POIs and edges than Australia network. This can also be seen from Fig. 10(b), which illustrates the number of edges expanded during query processing.

Index Size. Figure 10(c) shows the index size of the three approaches on different datasets. *EHA* has a larger index size than *EA* because it constructs an IR-tree for all POIs on the road network. *RHA* and *EHA* have all most the same index size.

Euclidean Heuristic Based RHA. Figure 11(a) presents the response time of the Euclidean heuristic based *RHA* (*ERHA*) that indexes all Rnets with an IR-tree as illustrated in Fig. 8. As *RHA* indexes a road network in a hierarchy and is already able to prune unrelated Rnets, *RHA* and *ERHA* have no much difference in response time.

Partition Strategy. Figure 11(b) and (c) illustrate the response time and number of expanded edges while *distribution-aware partition* (called *RHA-D*) is adopted. Compared to *RHA* using *equal-size partition* (*RHA-E*), *RHA-D* has a better performance. By partitioning a road network based on POI distribution, some Rnets have more POIs than others and accordingly have more keywords. Besides, areas containing few POIs can be partitioned into Rnets of larger granularity. Hence, this partition strategy makes *RHA-D* more advantageous in pruning unrelated Rnets than *RHA-E*.

8 Conclusion

In this paper, we define the *RC-SK* query on road networks and devise three approaches, i.e., *EA*, *EHA* and *RHA*, to deal with this problem. Both *EA* and *EHA* are suitable for *RC-SK* queries with a small query range while *RHA* also works excellently for *RC-SK* queries with a large query range.

In the future, we will consider some temporal spatial keyword queries over road networks because it is quite important for a city with heavy traffic.

Acknowledgement. This work was supported by National Natural Science Foundation (NSFC) under grant No. 61373036 and the Research Innovation Program of Shanghai Municipal Education Foundation under grant No. 13ZZ003.

References

1. Lee, K.C.K., Lee, W.-C., Zheng, B.: Fast object search on road network. In: Proceedings of EDBT, pp. 1018–1029 (2009)
2. Beckman, N., Kriegel, H.-P., Scheider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. In: Proceedings of SIGMOD, pp. 322–331 (1990)
3. Zobel, J., Moffat, A.: Inverted files for text search engines. *ACM Comput. Surv.* **38**(2), 1–56 (2006)
4. Huang, Y.-W., Jing, N., Rundensteiner, E.A.: Effective graph clustering for path queries in digital map. In: proceedings of CIKM, pp. 215–222 (1996)
5. Kernighan, B., Lin, S.: An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.* **49**(2), 291–308 (1970)
6. Yiu, M.L., Mamoulis, N.: Clustering objects on a spatial network. In: Proceedings of SIGMOD, pp. 443–454 (2004)
7. Cao, X., Chen, L., Cong, G., Jensen, C.S., Qu, Q., Skovsgaard, A., Wu, D., Yiu, M.L.: Spatial keyword querying. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012 Main Conference 2012. LNCS, vol. 7532, pp. 16–29. Springer, Heidelberg (2012)
8. Zhou, Y., Xie, X., Wang C., Gong, Y., Ma, W.: Hybrid index structures for location-based web search. In: Proceedings of CIKM, pp. 155–162 (2005)
9. Guttman, A.: R-trees: a dynamic index structures for spatial searching. In: Proceedings of SIGMOD, pp. 47–57 (1984)
10. Rocha-Junior, J.B., Norvag, K.: Top-k spatial keyword queries on road networks. In: Proceedings of EDBT, pp. 168–179 (2012)

11. Li, Z., Lee, K.C.K., Zheng, B., Lee, W.-C., Lee, D.L., Wang, X.: IR-tree: an efficient index for geographic document search. *IEEE TKDE* **23**(4), 585–599 (2011)
12. Faloutsos, C., Christodoulakis, S.: Signature files: an access method for documents and its analytical performance evaluation. *ACM TODS* **2**(4), 267–288 (1984)
13. Rocha-Junior, J.B., Gkorgkas, O., Jonassen, S., Nørvåg, K.: Efficient processing of top-k spatial keyword queries. In: Pfoser, D., Tao, Y., Mouratidis, K., Nascimento, M.A., Mokbel, M., Shekhar, S., Huang, Y. (eds.) *SSTD 2011*. LNCS, vol. 6849, pp. 205–222. Springer, Heidelberg (2011)
14. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. In: *Proceedings of SIGMOD*, pp. 337–348 (2009)
15. Felipe, I.D., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: *Proceedings of ICDE*, pp. 656–665 (2008)
16. Hariharan, R., Hore, B., Li, C., Mehrotra, S.: Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In: *Proceedings of SSDBM*, pp. 1–10 (2007)
17. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. In: *Proceedings of VLDB*, pp. 802–813 (2003)
18. Zhang, D., Chee, Y.M., Mondal, A., Tung, A.K.H., Kitsuregawa, M.: Keyword search in spatial databases: towards searching by document. In: *Proceedings of ICDE*, pp. 688–699, (2009)

A Spatial-Temporal Analysis of Users' Geographical Patterns in Social Media: A Case Study on Microblogs

Chao Li^{1,2}, Zhongying Zhao^{1,3(✉)}, Jun Luo^{1,4}, Ling Yin¹, and Qiming Zhou^{1,5}

¹ Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Beijing, China

² Graduate University of the Chinese Academy of Sciences, Beijing, China

³ College of Information Science and Engineering,

Shandong University of Science and Technology, Qingdao, China

⁴ Huawei Noah's Ark Laboratory, Hong Kong, China

⁵ Department of Geography, Hong Kong Baptist University, Hong Kong, China

{chao.li1,zy.zhao,jun.luo,yinling}@siat.ac.cn

qiming@hkbu.edu.hk

Abstract. With the development of information technologies, Social Media platforms have become popular and accumulated numerous data about individuals' behavior. It offers a promising opportunity of discovering usable knowledge about the individuals' movement behavior, which fosters novel applications and services. In this paper, in order to study the relations between communities and location clusters, we propose the index of location entropy to measure the degree of dispersion of the locations in each community, and the index of community entropy to measure the degree of dispersion of the communities in each location cluster. At last, we analyze users' trajectories and define four Trajectory Patterns. An algorithm is also proposed to extract those patterns from microblog data. We implement the algorithm and find some interesting and useful results for the intelligent recommender systems.

1 Introduction

Social Media is increasingly becoming one of the key platforms for people's interactions over the internet. The potential of these services lies in that they can gather large amount of data about individual's social behaviors as well as his/her personal information such as age, gender, home location, workplace, etc. At the same time, many users are now willing to share information about their locations, allowing for the study of the role of geographic distance in social ties.

This research was supported by International Science and Technology Cooperation Program of China (Grant No. 2010DFA92720-24), National Natural Science Foundation of China (NSFC) under Grant No. 61303167 and 11271351, and partially supported by Basic Research Program of Shenzhen (Grant No. JCYJ20130401170306838 and JC201105190934A).

Mining and analyzing those data has a great potential value for business. It offers an opportunity of discovering usable knowledge about the individual's behavior, which fosters novel applications and services.

The existing work on social media or social networks can be classified into community detection [1,2] information propagation [3], influence analysis [4], etc. However, few studies have taken individual's locations into consideration. One of the initial attempts to analyze how interactions on Social Media are affected by spatial distance is presented in [5], where the authors show that the probability of friendship decreases not only with distance but more precisely with the number of closer people. Li et al. [6] has proposed an information spreading model based on the location and community structure, but they did not give a definition about the main location of an individual as people might travel around and stay at several locations due to their business or entertainment. Some studies have investigated the structural properties of a location-based social network and how social and geographic distance influences the creation of new connections among its users [7]. Yet, they did not conduct a macroscopic statistical analysis about the recorded locations over the social media, neither did they study the relationships between online communities and geographical location clusters. In fact, users often change their geographical locations due to their work or travel, which results in different trajectories. However, scarce work has been devoted to identify the patterns behind those trajectories.

In this paper, we aim to study the spatial-temporal patterns by analyzing the Tencent microblogging users. Our contributions include:

1. We investigate the location clusters involved in each communities, and propose an index 'Location Entropy' to measure the degree of location dispersion in each community. It can potentially help measure the influential power for a topic community.
2. We also investigate the communities involved in each location cluster, and propose the index 'Community Entropy' to measure the mixing degree of the communities in each location cluster. It can potentially help monitor the active degree of people's online social behavior in a location cluster.
3. At last, we define four 'Trajectory Patterns' to describe individuals' moving behavior. And then propose an algorithm to detect Trajectory Patterns from large scale microblogging data. Some results gained from those patterns are useful to the intelligent recommender system.

The remainder of the paper is organized as follows. Section 2 introduces the related work. In Sect. 3, we analyze the relationship between users' locations and communities. In Sect. 4, we define individuals' Trajectory Patterns, identify and analyze those patterns based on Microblog data. Finally, we conclude this study in Sect. 5.

2 Related Work

In the recent years massive online social networks such as Facebook, MySpace, LinkedIn, Flickr and Twitter [8] have become increasingly popular, gathering millions of users and engaging them in the production, sharing and consumption of information over social links. Moreover, they are increasingly becoming location-aware: they offer an opportunity to share geographic locations in order to generate location-tagged information and to search for it. Therefore, a large number of studies have been done based on social interactions and locations.

The effect of geography over complex networks has been studied mainly in communication and transportation networks [2]. For instance, it has been found that the spatial properties of the Internet topology are mainly determined by both preferential attachment and linear distance dependence [8], whereas Internet traffic is spatially bound to a spanning network which connects the most important centers around the globe [9]. However, these studies do not investigate online social networks, which rarely have spatial constraints. One of the first attempts to analyze how interactions on online social networks are affected by spatial distance is presented in [5], where the authors show that the probability of friendship decreases not only with distance but more precisely with the number of closer people. Some studies have investigated the structural properties of a location-based online social network and how social and geographic distance influences the creation of new connections among its users.

The basic frequent sequential pattern (FSP) problem, originally introduced in [10], is defined over a database of sequences D , where each element of each sequence is a timestamped set of items, i.e., an itemset. Timestamps determine the order of elements in the sequence. Then, the FSP problem consists in finding all the sequences that are frequent in D , i.e., appear as subsequences of a large percentage of sequences of D . Therefore, many algorithms for sequential patterns have been proposed, from the earliest in [10], to the more recent PrefixSpan [11] and SPADE [12]. There are some studies about the trajectory patterns. The work in [13] considers patterns that are in the form of trajectory segments and searches approximate instances in the data; on the opposite, the work in [14] provides a clustering-based perspective, and considers patterns in the form of moving regions within time intervals, such as spatiotemporal cylinders or tubes and counts as occurrences all trajectory segments partially contained in the moving regions. Finally, a similar goal, but focused on cyclic patterns, is pursued in [15]: the authors propose an effective and fast mining algorithm for retrieving maximal periodic patterns, treating time as discrete, yet dealing with continuous spatial locations that are discredited dynamically through density based clustering.

3 Spatial Analysis: Location Clusters and Communities

In this section, we first give the description about the Microblog data. Then we analyze the degree of location dispersion in each community, and explore the mixing degree of the communities in each location cluster.

Table 1. The description of the three scales of networks.

Scale of the data sets	#nodes	#edges
Small	11221	108729
Medium	15912	171756
Large	36364	532444

3.1 Description About the Microblog Data

Tencent-Microblog offers an Application Programming Interface (API) for users to crawl and collect data, such as an user's ID, an user's microblogs' contents, and the geographical location of a microblog post at a city level. In order to study the geographical locations and social behaviors of the users, we crawled data of three months (2011.4–2011.6), which contains 41713 users, 3119942 microblog posts, and 331 different locations of posts. We choose three scale levels of networks: small, medium, and large network. And we then identify a social network at each scale. The details of the three scales of social networks data are described in Table 1. We only built one edge between two nodes which follow each other.

In order to study the interplay of online communities and location clusters, we need to identify online communities and discover location clusters. We first use the Main Location for each user based on [17], and then find the location clusters based on the province level in China. Meanwhile, we use the modularity method [16] to detect communities based on users' interactions: follow and be followed. As a result, we identify some online communities and geographical location clusters in the whole social network.

3.2 Analyzing Location Components for Each Community

To evaluate the degree of the location dispersion in community, we define the index of Location Entropy as follows.

Definition 1. We define the location entropy for the community C_i as $E(C_i)$:

$$E(C_i) = - \sum_{j=1}^n p_{l_{ij}} * \log_2 p_{l_{ij}} \quad (1)$$

Where $p_{l_{ij}}$ represents the percentage of the users of the Main Location l_j in community C_i

Figure 1 illustrates the distributions of the location entropy ($E(C_i)$) for each scale of networks. From Fig. 1, we can see that, for the smaller community which contains fewer users, the entropy is differs from 1.5 to 4. While for the larger community, the value of location entropy is very stable, this keeps in the neighborhoods of 4. It can potentially help measure the influential power for a topic community.

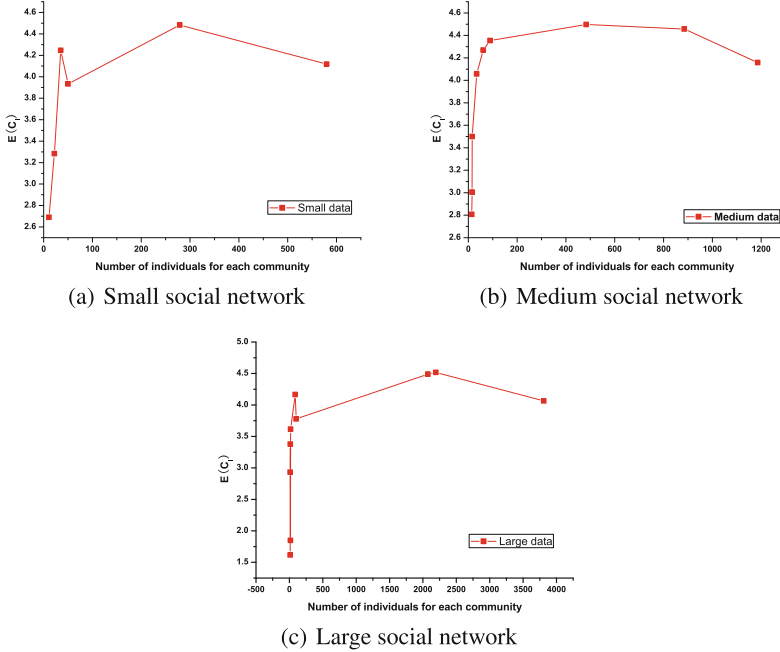


Fig. 1. The distributions of the location entropies in three scales of social networks

3.3 Analyzing Community Components for Each Location Cluster

In this section, we aim to analyze the community components for each location cluster. Similar to Sect. 3.2, we propose the definition of community entropy to describe the disordering degree of online communities within each location cluster. The definition of community entropy is given as follows.

Definition 2. We define the community entropy for location cluster L_m as $E(L_m)$:

$$E(L_m) = - \sum_{i=1}^n p_{c_i} * \log_2 p_{c_i} \tag{2}$$

Where p_{c_i} represents the percent of the users who belong to community c_i at location cluster L_m

Figure 2 shows the distributions of community entropy of three scales of networks. It indicates a weak relationship between the number of individuals at a location and the community entropy. But it can potentially help monitor the active degree of people’s online social behavior in a location cluster.

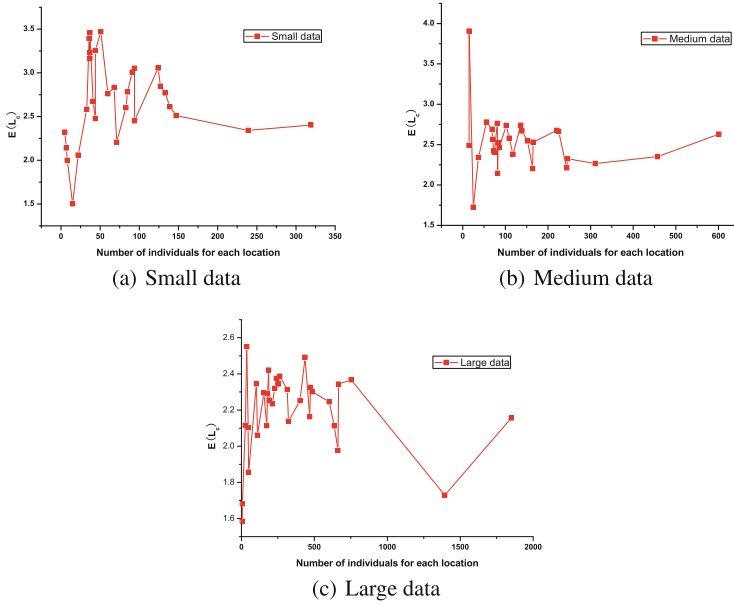


Fig. 2. The distributions of the community entropy in three kinds of networks

4 Spatial-Temporal Analysis: Trajectory Pattern Analysis

In this section, we define four typical Trajectory Patterns, and propose a Trajectory Pattern detection algorithm. At last we show some interesting results from the implemented algorithm on the Mircoblog data.

4.1 Trajectory Pattern

There are often some regular Trajectory Patterns (T-Patterns) behind users moving behavior. In this section, we define four T-Patterns to describe some typical individuals' moving behavior.

Definition 3. Stable Trajectory (STr)

$$STr(u) = \{A_{t_1}, A_{t_2}, \dots, A_{t_{n-1}}, A_{t_n}\} \tag{3}$$

where:

- $STr(u)$ represents the individual u 's Stable Trajectory Pattern
- A denotes the Main Location of user u
- A_{t_i} denotes the location of user u at the time of t_i
- $\{t_1 < t_2 < \dots < t_{n-1} < t_n\}$

According to the above definition, the Stable Trajectory reflects the situation that a user often stays in a certain location and rarely go to other places. Figure 3 illustrates an example of this pattern.

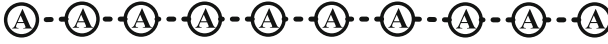


Fig. 3. Stable Trajectory (*STr*)

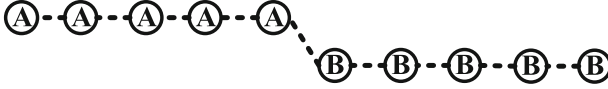


Fig. 4. Migrating Trajectory (*MTr*)

Definition 4. Migrating Trajectory (*MTr*)

$$MTr(u) = \{A_{t_1}, A_{t_2}, \dots, A_{t_i}, B_{t_{i+1}}, B_{t_{i+2}}, \dots, B_{t_{n-1}}, B_{t_n}\} \tag{4}$$

where:

- *MTr*(*u*) represents the user *u*'s Migrating Trajectory pattern
- *A* and *B* denote two locations of user *u*
- A_{t_i} denotes the location of user *u* at the time of t_i
- B_{t_i} denotes the location of user *u* at the time of t_i and $B \neq A$
- $\{t_1 < t_2 < \dots < t_i < t_{i+1} < \dots < t_{n-1} < t_n\}$

According to the above definition, the Migrating Trajectory reflects the situation that a user moves from the location *A* to another new location *B* and does not return. Figure 4 gives an example of this pattern.

Definition 5. Visiting Trajectory (*VTr*)

$$VTr(u) = \{A_{t_1}, A_{t_2}, \dots, A_{t_i}, B_{t_{i+1}}, B_{t_{i+2}}, \dots, B_{t_{i+m}}, A_{t_{i+m+1}}, \dots, A_{t_j}, C_{t_{j+1}}, \dots, C_{t_{j+k}}, A_{t_{j+k+1}} \dots A_{t_l}, D_{t_{l+1}}, \dots, D_{t_{l+d+1}}, \dots, A_{t_{n-1}}, A_{t_n}\} \tag{5}$$

where:

- *VTr*(*u*) represents the user *u*'s Visiting Trajectory Pattern.
- *A* denotes the Main Location of user *u*
- *B, C, D, ...* denote the visiting locations of user *u*
- $B_{t_i}, C_{t_i}, D_{t_i}, \dots$ denote the location of user *u* at the time of t_i
- $\{t_1 < t_2 < \dots < t_i < t_{i+1} < \dots < t_{n-1} < t_n\}$

According to the above definition, the Visiting Trajectory reflects the situation that a user often visits different places but still has a Main Location. Figure 5 illustrates two examples of this pattern. Figure 5(a) means that the user visits one place and then returns to his/her Main Location. While Fig. 5(b) means that the user visits more than one place and then returns to his/her Main Location.

Definition 6. Flickering Trajectory (*FTr*)

$$FTr(u) = \{A_{t_1}, A_{t_2}, \dots, A_{t_i}, B_{t_{i+1}}, B_{t_{i+2}}, \dots, B_{t_{i+m}}, A_{t_{i+m+1}}, \dots, A_{t_j}, B_{t_{j+1}}, \dots, B_{t_{j+k}}, A_{t_{j+k+1}} \dots A_{t_l}, B_{t_{l+1}}, \dots, B_{t_{l+d+1}}, \dots, A_{t_{n-1}}, A_{t_n}\} \tag{6}$$

where:

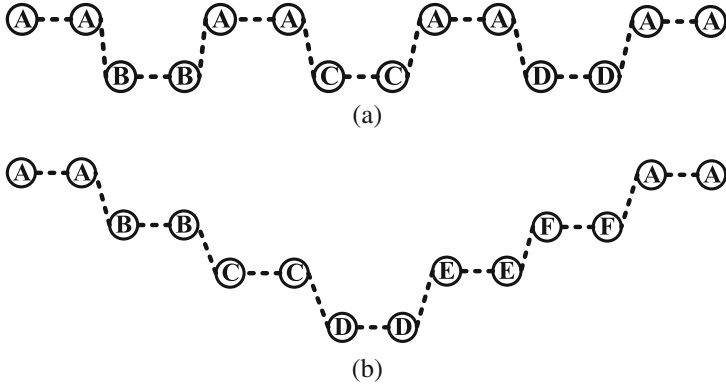


Fig. 5. Examples of Visiting Trajectory (*VTr*)

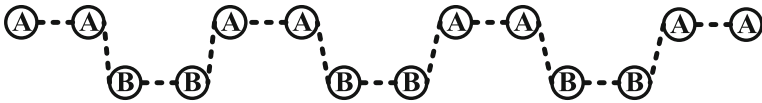


Fig. 6. Flickering Trajectory (*FTr*)

- $FTr(u)$ represents user u 's Flickering Trajectory Pattern.
- A, B denote different locations of user u .
- A_{t_i}, B_{t_i} denote the locations of user u at the time of t_i
- $\{t_1 < t_2 < \dots < t_i < \dots < t_{i+m} < \dots < t_j < \dots < t_l < \dots < t_{l+d}\}$

According to the above definition, the Flickering Trajectory reflects the situation that a user changes his location between A and B. Figure 6 is such an example of this pattern.

An algorithm for extracting the above four Trajectory Patterns is presented in Algorithm 1. The algorithm iteratively considers all the spatial-temporal data and calculates the probability of different trajectory for each individual. At last, the individual is assigned to the Trajectory Pattern with the highest probability value.

4.2 Trajectory Analysis on Tencent Microblogs

In order to study the trajectories and find some meaningful results, we have developed software to crawl data from Tencent microblogs. We implement algorithm 1 to detect four different Trajectory Patterns for each individual and then analyze those trajectories and cities involved. We also find some hot cities, hot pairs of cities and hot star cities.

From the our results, we find that more than half of people's moving behavior are characterized as Visiting Trajectory and few of them belong to the pattern of Migrating Trajectory. We also consider the cities that appear in trajectories

Algorithm 1. Trajectory Pattern discovery algorithm

Require: The list of user’s locations with timestamps

Data{*User*1, *Location*, *Timestamp*}

Ensure: The pattern label for each user and his/her trajectories

Formulate user’s trajectory(*UT*(*user*, *trajectory*)) based on locations and timestamps.

Create some rules to detect trajectory patterns according to Eq. 3,4,5,6

Get the Trajectory Pattern list {*Patternname*, *Patternlabel*, *Patternrule*}.

for User *u* in the list *UT*(*user*, *trajectory*) **do**

 Match the *trajectory*(*u*) with the rule of detecting trajectory pattern.

 Calculate the probability for each Trajectory Pattern.

 Assign the pattern label with the highest probability value to each individual.

end for

return {*user*, *patternlabel*, *trajectory*}

Table 2. Hot City Top 20.

City	#Users	City	#Users	City	#Users	City	#Users
BeiJing	1256	GuangZhou	365	ZhengZhou	228	HangZhou	172
ShangHai	842	TianJin	297	Xi’An	220	QuanZhou	166
ShenZhen	683	ChongQing	284	ChangChun	194	TaiYuan	159
ChengDu	611	ShenYang	274	ShiJiaZhuang	189	NanNing	157
WuHan	557	Ha’ErBin	230	JiNan	174	ChangSha	150

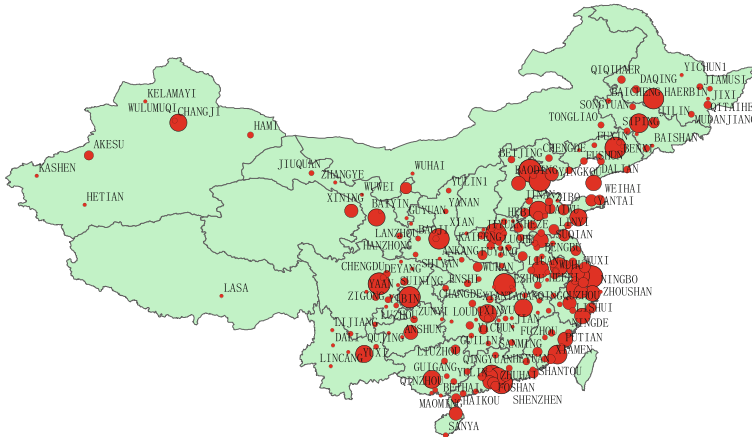


Fig. 7. The distribution of hot cities

frequently. Here we call them ‘hot cities’. Table 2 shows the top 20 hot cities which are ranked in order of the number of users involved. We find that not all of the cities are capitals (e.g. Shenzhen). The distribution of those hot cities are illustrated in Fig. 7.

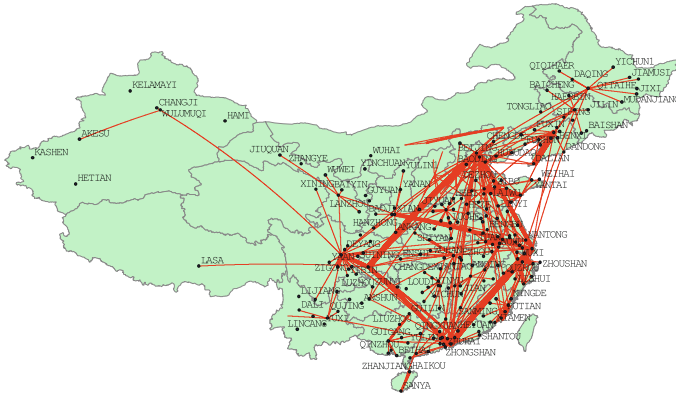


Fig. 8. The distribution of hot city-pairs

Table 3. Hot City-pairs Top 20.

City pair	#Users	City pair	#Users	City pair	#Users
GuangZhou–ShenZhen	1131	ChengDu–ChongQing	242	BeiJing–ShangHai	168
BeiJing–ChongQing	569	Ma’AnShan–HeFei	233	ShenZhen–DongGuan	154
JiNan–QingDao	497	LeShan–ChengDu	206	ShiJiaZhuang–BaoDing	124
ShangHai–ShenZhen	348	ChongQing–GuangZhou	198	ShiJiaZhuang–ChengDe	106
ShenYang–DaLian	294	HangZhou–NingBo	194	BeiJing–BaoDing	103
BeiJing–TianJin	255	Xi’An–NanJing	188	AnYang–ZhengZhou	100
QuanZhou–FuZhou	255	HangZhou–WenZhou	169	–	–

The last but not least, hot city-pairs are also detected from users' trajectories. We calculate the number of the co-occurrence of two cities in all the users' trajectories and then select those pairs with higher frequency. Here, we call them hot city-pairs. Figure 8 illustrates the distribution of the hot city-pairs which are detected from from users' microblogging trajectories. Table 3 shows the top 20 hot city-pairs.

Table 3 shows that lots of people like to travel between Shenzhen and Guangzhou, Qingdao and Jinan etc. On the one hand, the hot city-pairs mined from users' trajectories are useful for the traffic or flight planning. For example, with the knowledge of hot city-pair of 'Beijing-Chongqing', the corresponding department may consider to arrange more flights or trains between them. Therefore, the results can offer some supporting knowledge for the decision of governments. On the other hand, the hot city-pairs that we detected also reflect the current traffic situation. For example, there are 135 trains from between Shenzhen and Guangzhou. More than 46 trains are planned between Qingdao and Jinan. More than 94 flights are scheduled between Beijing and Chongqing. Therefore, our results are indeed consistent with the real transports. It will play a very important role in analyzing and planning transporting system between different cities.

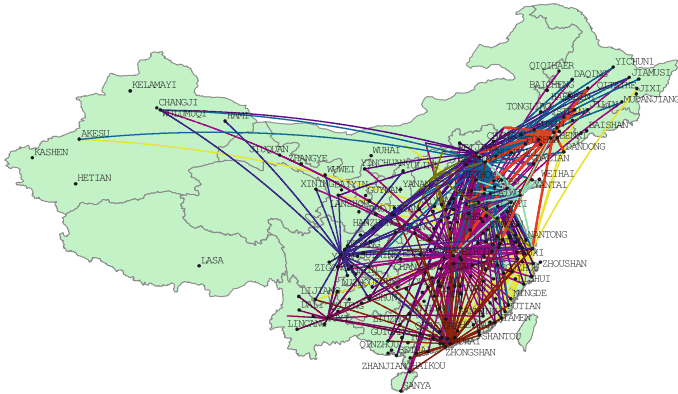


Fig. 9. The distribution of hot city-star

With users' microblogging trajectories, we also find some star structures. That is, many cities are connected closely with a certain city, which results in a star-structure. Some examples are shown in Fig. 9. According to Fig. 9, we can see that Beijing co-appears with many other cities and they form a star structure. And Beijing is in the center of a huge star-structure. Here, we consider it as a Beijing-centered star. Such examples include Shanghai-centered star, Wuhan-centered star, and Shenzhen-centered star. Comparatively, the city like Dalian, Xi'an, and Qingdao is connected with less cities. But they also form the star structure.

Furthermore, we find some hot routes, such as Shenzhen-Guangzhou-Chongqing, or Qingdao-Jinan-Beijing etc. from the above results. Those results can be referred by the tourism to make intelligent recommendations for people who are planning to travel.

5 Conclusion

In this paper, we make a spatial-temporal analysis on people's geographical patterns in a case study of Tencent-Microblog. We first study the relations between communities and location clusters, we propose the index of location entropy to measure the degree of dispersion of the locations in each community, and the index of community entropy to measure the degree of dispersion of the communities in each location cluster. More importantly those two indexes can potentially help measure the influential power for the topic community and monitor the active degree of people's online social behavior in a location cluster. At last, we analyze user's trajectory and define four Trajectory Patterns. An algorithm is also proposed to extract those patterns from microblog data. Finally, we demonstrate the effectiveness of the algorithm to find some interesting and useful results for intelligent recommender systems.

References

1. Zhao, Z., Feng, S., Wang, Q., Huang, J.Z., Williams, G.J., Fan, J.: Topic oriented community detection through social objects and link analysis in social networks. *Knowl. Based Syst.* **26**, 164–173 (2012)
2. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* **70**(6), 066111 (2004)
3. Cha, M., Mislove, A., Gummadi, K.P.: A measurement-driven analysis of information propagation in the flickr social network. In: *Proceedings of the 18th International Conference on World Wide Web*, pp. 721–730. ACM (2009)
4. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 199–208. ACM (2009)
5. Liben-Nowell, D., Novak, J., Kumar, R., Raghavan, P., Tomkins, A.: Geographic routing in social networks. *Proc. Natl. Acad. Sci. U.S.A.* **102**(33), 11623–11628 (2005)
6. Li, C., Zhao, Z., Luo, J., Fan, J.: Info-cluster based regional influence analysis in social networks. In: *Advances in Knowledge Discovery and Data Mining*, pp. 87–98 (2011)
7. Humphreys, L.: Mobile social networks and social practice: a case study of dodgeball. *J. Comput. Mediated Commun.* **13**(1), 341–360 (2007)
8. Yook, S.-H., Jeong, H., Barabási, A.-L.: Modeling the internet's large-scale topology. *Proc. Natl. Acad. Sci.* **99**(21), 13382–13386 (2002)
9. Barthelemy, M., Gondran, B., Guichard, E.: Spatial structure of the internet traffic. *Physica A: Stat. Mech. Appl.* **319**, 633–642 (2003)
10. Agrawal, R., Srikant, R.: Mining sequential patterns. In: *Proceedings of the Eleventh International Conference on Data Engineering*, pp. 3–14. IEEE (1995)
11. Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In: *ICDE*, pp. 215–224, April 2001
12. Zaki, M.J.: Spade: an efficient algorithm for mining frequent sequences. *Mach. Learn.* **42**(1), 31–60 (2001)
13. Cao, H., Mamoulis, N., Cheung, D.W.: Mining frequent spatio-temporal sequential patterns. In: *Fifth IEEE International Conference on Data Mining*, pp. 82–89. IEEE (2005)
14. Kalnis, P., Mamoulis, N., Bakiras, S.: On discovering moving clusters in spatio-temporal data. In: *Advances in Spatial and Temporal Databases*, pp. 923–923 (2005)
15. Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., Cheung, D.W.: Mining, indexing, and querying historical spatiotemporal data. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 236–245. ACM (2004)
16. Gastner, M.T., Newman, M.E.: The spatial structure of networks. *The Eur. Phys. J. B-Condens. Matter Complex Syst.* **49**(2), 247–252 (2006)
17. Li, C., Zhao, Z., Liu, S., Yin, L., Luo, J.: Relationships between geographical cluster and cyberspace community: a case study on microblog. In: *2012 20th International Conference on Geoinformatics (GEOINFORMATICS)*, pp. 1–5. IEEE (2012)

Solving Multiple Bichromatic Mutual Nearest Neighbor Queries with the GPU

Marta Fort and J. Antoni Sellarès^(✉)

Dept. Informàtica, Matemàtica Aplicada i Estadística,
Universitat de Girona, Girona, Spain
{mfort,sellares}@imae.udg.edu

Abstract. In this paper we propose and solve multiple bichromatic mutual nearest neighbor queries in the plane considering multiplicative weighted Euclidean distances. These multiple queries are related to the mutual influence of two sets of facilities of different type, in which facilities of the first type cooperates with facilities of the second type in order to obtain reciprocal benefits. The studied problems find applications, for example, in collaborative marketing. We present a parallel algorithm, to be run on a Graphics Processing Unit, for solving multiple bichromatic mutual nearest neighbor queries. We also present the complexity analysis of the algorithm, and provide and discuss experimental results that show the scalability of our approach.

1 Introduction

In this paper we study a problem related to the mutual influence of two sets of facilities of different type, in which facilities of the first type cooperate with facilities of the second type in order to obtain reciprocal benefits. The two main elements of the problem are facilities and locations. Facilities provide goods or services (hotels, warehouses, hospitals) and locations are spatial positions where facilities are located. Throughout the paper we represent locations by points on two-dimensional space and use interchangeably the terms facility and location of the facility. To take into account the different importance of the facilities, we assign a weight to each facility. In practice, experts take available information of the facilities (prestige, magnitude, services, etc.) and then aggregate these factors to obtain weights [8, 9]. To reflect that influence between facilities depends on distance and importance, we assign to each facility a multiplicative weighted distance that is the product of the Euclidean distance and the inverse of the facility weight. From now on, we assume that proximity and neighborhood influence between facilities is measured by using multiplicative weighted Euclidean distances.

Although the proximity, represented by distance measures, of a facility to other facilities is an indicator of its potential impact, we also need to know the

Work partially supported by the Spanish Ministerio de Economía y Competitividad under grant TIN2010-20590-C02-02.

capacity that a facility has to influence and to be influenced for other facilities with which it interfaces. Mutual nearest neighbor queries model, in a natural way, how a facility influences and it is influenced for its neighboring facilities. Given two data sets of facilities of different type and a query point of the first set, a bichromatic (k, k') -mutual nearest-neighbor query finds the points in the second set whose k -nearest neighbors on the first set include the given query point and meanwhile have the query point as one of the k' -nearest neighbors of the resulting points in the second set. Mutual nearest-neighbor queries are more suitable, compared with nearest-neighbor queries, for those applications involving symmetric nearest-neighbor relationships, including among others: data clustering [2, 13], outlier detection [2, 18], decision making [11, 16, 22], and pattern recognition [14]. Given two sets of facilities of different type, a multiple bichromatic (k, k') -mutual nearest-neighbor query associates each query point of the first set with its bichromatic (k, k') -mutual nearest-neighbors from the second set. To the best of our knowledge, multiple bichromatic mutual nearest neighbor queries considering multiplicative weighted Euclidean distances have not received attention so far, although they have practical applications as we will see in the example that follows.

Nowadays, it has become difficult for firms to stay competitive without allying themselves with other firms. Thus, there is an increasing interest in inter-firm relationships with the objective of building successful collaborative strategies that can provide competitive advantages to the firms involved. Collaborative marketing is accomplished by companies working together in cooperative activities, such as product promotion or marketing communication, in order to create synergies and achieve a superior market position for their products and services.

Example. Consider the set of hotels of a given region together a subset of these hotels, for example the hotels of a chain or having some characteristics, and the set of cultural, recreational and sport centers (museum, theater, zoo, thematic park, football or basketball stadium) of the same given region. An advertisement agency can use the results of a multiple bichromatic mutual nearest neighbor query for the subset of hotels and the cultural, recreational and sport centers, to make directly collaborative advertisements; e.g., hotels offer 5% discount for people going to one of the cultural, recreational or sport centers and these centers offer 10% discount to hotel customers who visit them.

Since the multiplicative weighted Euclidean distance does not obey the triangle inequality, we cannot use an index structure, that usually is associated with a partition of the plane heavily relying on the triangle inequality, to filter out irrelevant facilities during the search of multiple bichromatic mutual nearest neighbors. This has motivated us, in working towards practical solutions, to design a GPU-parallel approach, under CUDA architecture, for solving multiple bichromatic mutual nearest neighbor queries. We provide the complexity analysis of our algorithm and discuss some preliminary experimental results that show the scalability of our approach.

1.1 Related Work

In recent years, in the context of spatial datasets, k -nearest neighbor queries, bichromatic reverse k -nearest neighbor queries, and bichromatic mutual nearest neighbor queries have attracted considerable attention.

Most of the existing k -nearest neighbor query algorithms use index structures and are based on either depth-first [6] or best-first [17] traversal paradigm. Algorithms to solve all k -nearest neighbor queries are provided in [7, 24, 25]. The concept of reverse k -nearest neighbor was first introduced in [19]. There exist many papers on the bichromatic reverse k -nearest neighbor query [1, 21, 23]. Almost all the proposed solutions basically focus on using the Voronoi diagram, exact or approximated, and then find the region that corresponds to the query point. Wong et al. [22] introduced the concept of bichromatic mutual nearest neighbor query and employed it to deal with spatial matchings. Gao et al. [11, 15, 16] explored (monochromatic) mutual nearest neighbor queries involving spatial and trajectory databases.

Basic brute force k -nearest neighbor search on the GPU is much faster than it is on the CPU, and even compares favorably to CPU implementations that uses an index structure [5]. There exist many works about using GPUs to accelerate the brute force k -nearest neighbor search. Most of them are based on: index structures [3–5], sort algorithms with some modification or customization [12], or maintaining only the k -nearest neighbors during the search [20]. In [20] it is also solved the all k -nearest neighbors problem. A GPU-parallel approach for approximately solving reverse k -influential location problems is provided in [10].

2 Multiple Bichromatic Mutual Nearest Neighbor Queries

Let P be a set of n points within a bounded domain D of the Euclidean plane. Each $p \in P$ is associated with a positive real weight $w_p > 0$. The multiplicative weighted distance $d_p(q)$ from the location $p \in P$ to an arbitrary point $q \in D$ is defined as $d_p(q) = (1/w_p) d(p, q)$, where $d(p, q)$ denotes the Euclidean distance among p and q . Note that, the multiplicative weighted Euclidean distances is non-metric, it is not symmetric and the triangle inequality does not hold because the weights change for each point.

k-nearest neighbor queries

For any point $q \in D \setminus P$, denote by $d_k(P, q)$ the weighted distance from q to its k -th nearest point in P , i.e. the weighted distance $d_{\bar{p}}(q)$ to the point of $\bar{p} \in P$ that ranks number k in the ordering of the points by increasing weighted distance from q . Observe that several points of P can share the same k -th weighted distance value.

A *k*-nearest neighbor query for a point $q \in D \setminus P$, finds the subset $NN_k(P, q)$ of k -nearest points of P to the query point q :

$$NN_k(P, q) = \{p \in P \mid 0 < d_p(q) \leq d_k(P, q)\}.$$

For a given q , the k -nearest neighbors of q may not have q as one of their own k -nearest neighbors. The points of $NN_k(P, q)$ are the points that have the highest influence on q . The nearest neighbor relation is not symmetric.

Bichromatic reverse k -nearest neighbor queries

Let R and S be disjoint weighted sets of n and m points within a bounded domain D of the Euclidean plane.

A *bichromatic reverse k -nearest neighbor query* for a point $r \in R$ seeks to determine the set $BRNN_k(R, S, r)$ of points $s \in S$ for which r is a k -nearest neighbor in R :

$$BRNN_k(R, S, r) = \{s \in S \mid r \in NN_k(R, s)\} = \{s \in S \mid 0 < d_r(s) \leq d_k(R, s)\}.$$

The number of bichromatic reverse k -nearest neighbors of a query point varies between 0 and m . The points of $BRNN_k(R, S, r)$ are the most highly influenced by r .

Conventional k -nearest neighbor queries are asymmetric. Many practical applications require a symmetric neighborhood relationship. We introduce (k, k') -mutual nearest neighbor queries, that are symmetric and moreover define a tighter neighborhood relation than the usual k -nearest neighbor queries.

Bichromatic mutual (k, k') -nearest neighbor queries

A *bichromatic mutual (k, k') -nearest neighbor query* for the point $r \in R$ determines the subset $BMNN_{k,k'}(R, S, r)$ of points $s \in S$ in the k -nearest neighbor of r in S which have the query point r in their k' -nearest neighbors in R :

$$\begin{aligned} BMNN_{k,k'}(R, S, r) &= \{s \in S \mid s \in NN_k(S, r) \cap BRNN_{k'}(R, S, r)\} = \\ &= \{s \in S \mid s \in NN_k(S, r) \wedge r \in NN_{k'}(R, s)\} = \\ &= \{s \in S \mid 0 < d_s(r) \leq d_k(S, r) \wedge 0 < d_r(s) \leq d_{k'}(R, s)\}. \end{aligned}$$

The number of bichromatic mutual (k, k') -nearest neighbors of a query point varies between 0 and k . Points $s \in BMNN_{k,k'}(R, S, r)$, are the points of S that have the highest influence on r and the most highly influenced by r .

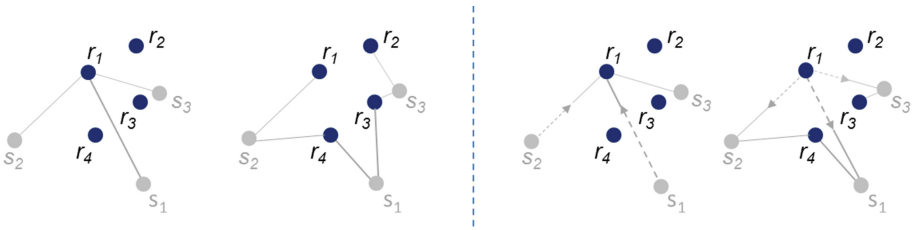


Fig. 1. Bichromatic mutual (2, 2)-nearest neighbor for: (a) Euclidean distances; (b) multiplicative weighted Euclidean distances.

In Fig. 1(a), where Euclidean distances are considered, we have $NN_2(S, r_1) = \{s_3, s_2\}$, $BRNN_2(R, S, r_1) = \{s_2\}$, consequently it is $BMNN_{(2,2)}(R, S, r_1) = \{s_2\}$. In Fig. 1(b), we consider distances with weights $w_{r_1} = 2$, and $w_{r_2} = w_{r_3} = w_{r_4} = 1$, and $w_{s_1} = 4$, $w_{s_2} = 2$ and $w_{s_3} = 1$, and we have $NN_2(S, r_1) = \{s_1, s_2\}$, $BRNN_2(R, S, r_1) = \{s_1, s_2, s_3\}$, thus $BMNN_{(2,2)}(R, S, r_1) = \{s_1, s_2\}$.

Multiple bichromatic mutual(k, k')-nearest neighbor queries

In many applications, a query does not consist of a single point but a whole set of points, for each of which a query has to be performed. To the best of our knowledge, this multiple operation has not received attention in the context of bichromatic mutual nearest neighbor queries.

Given two non-empty subsets $R' \subseteq R$ and $S' \subseteq S$, a *multiple bichromatic mutual (k, k')-nearest neighbor query* computes for each point $r \in R'$ the subset of $BMNN_{k,k'}(R, S, r)$ of its bichromatic mutual (k, k')-nearest neighbors in S' .

Notice that, even though the subsets R' and S' are used, the multiple bichromatic mutual (k, k')-nearest neighbor queries are always referred to R and S .

3 Solving Multiple Bichromatic Mutual Nearest Neighbor Queries with the GPU

Given two disjoint weighted sets R and S within a bounded domain D of the Euclidean plane, and two non-empty subsets $R' \subset R$ and $S' \subset S$, a way to solve a multiple bichromatic mutual (k, k')-nearest-neighbor query is to sequentially compute for each point $r \in R'$ the set $NN_k(S, r) \cap S'$, and then verify whether each point s in $NN_k(S, r) \cap S'$ has r as one of its k' -nearest neighbors in R . We adapt this solution, that is highly parallelizable, to solve the multiple mutual nearest neighbor queries efficiently in the GPU.

From now on we will denote $n = |R|$, $m = |S|$, $n' = |R'|$ and $m' = |S'|$.

Step 1. Computing the k -nearest neighbors in S of the points of R' .

Since the multiplicative weighted distance does not obey the triangle inequality we cannot take advantage of any usually used index structure to find the k -nearest neighbors to filter out irrelevant facilities during the search. Hence, we compute the k -nearest neighbors in S of each $r \in R'$ by processing all the points of R' in parallel. A kernel executed by one thread per point $r \in R'$ is used. Each thread considers its corresponding point $r \in R'$ and finds the k -nearest neighbors in S of que query point, and its k th-nearest distance to S , $d_k(S, r)$. Thus, the points of S and R' are transferred from the CPU to the GPU, and then the threads in a block cooperate to transfer S to shared memory. The indices of the k -nearest neighbors of each point $r \in R'$ to S are stored in a global memory array n_{S_k} of size kn' .

Note that, for the subsequent steps we are not interested in all the k -nearest neighbors of each $r \in R'$ but in those contained in S' . As well as, we are only

interested in the significant points of S' , which are those points $s \in S'$ that are k -nearest neighbors of at least one point $r \in R'$.

Step 2. Finding the significant points of S' . This step is performed in two different kernels. In the same kernel of Step 1, each thread, after having obtained the k -nearest neighbors of its corresponding $r \in R'$, checks whether they belong to S' . With this aim, an integer array sp of size m initialized so that $sp[i] = 0$ if $s_i \notin S'$ and 1 otherwise, is used. Thus, each thread considers the indices, j , of the obtained k -nearest neighbors, and if $sp[j] = 0$ instead of storing the index j we store a -1 . Meanwhile, if $sp[j] \neq 0$, we label $sp[j]$ as a significant point by setting it to 2. When all the points $r \in R'$ have been processed, the significant point of S have been labeled in sp with a 2, and n_{S_k} contains either indices of the k -nearest neighbors of the points of R' that belong to S' or -1 s.

Finally, by using another kernel we count and store in consecutive positions of a new array, s_{sig} , the indices of the significant points of S' , by using a kernel executed by m threads. The thread idx checks whether $sp[idx] = 2$. If it is so, it increments a global counter by one with an atomic operation, and stores its index idx in the first empty position of s_{sig} .

The following steps 3 and 4 are only processed if there are significant points.

Step 3. Computing the k' -nearest neighbor distance in R of the points of S_{sig} . Now, we compute the k' -nearest neighbor distance in R of each significant point S_{sig} by using the strategy explained in the first step using one thread per point in S_{sig} . Now, the points of R are transferred first to global memory, and then to shared memory. Now, we only store the k' -nearest neighbor distance, $d_{k'}(R, s)$, of each significant point $s \in S_{sig}$ to R , in an array $d_{R_{k'}}$.

Step 4. Obtaining the mutual (k, k') -nearest neighbors. To find the mutual (k, k') -nearest neighbors, we check, for each pair (r_i, s_l) with $r_i \in R'$ and $l = n_{S_k}[ki + j] > 0$ with $0 \leq j < k$, whether they are actual mutual (k, k') -nearest neighbors. This is done by using kn' threads, the thread idx considers $l = n_{S_k}[idx]$ and analyzes the corresponding pair or does nothing if $l = -1$. To determine whether a pair (r_i, s_l) is a mutual (k, k') -nearest neighbor pair, the thread checks whether $d_{r_i}(s_l) \leq d_{R_{k'}}[l]$. If the inequality is fulfilled, it increments by one n_M , a counter initialized to zero, and stores the pair (i, l) in the first two consecutive empty positions of an integer array $m_{k,k'}$ of size $2kn'$. At the end of the process, the number T stored in n_M is the number of mutual pairs, and the pairs are stored in the first $2T$ positions of $m_{k,k'}$.

To obtain the solution in the CPU we transfer T , which is stored in n_M , from the GPU to the CPU and the first $2T$ integer values of $m_{k,k'}$.

3.1 Complexity Analysis

In this section we analyze the amount of work performed by each thread and the worst case total work by studying each kernel separately. Finally, the total work done by the whole algorithm is provided.

In the first kernel, corresponding to the obtention of the k -nearest neighbors of R' of Step 1 and the labeling part of Step 2, $O(kn'm)$ total work is done by using n' threads doing at most $O(km)$ work each. Each thread analyzes the m points of S and stores the k -nearest neighbors and distances in a sorted way, by doing $O(km)$ work in the worst case. The work done in the second kernel, which extracts the significant points S_{sig} in Step 2, is $O(m)$ and is done by m threads doing $O(1)$ work each. In this kernel $O(m'')$, with $m'' = |S_{sig}|$, atomic operations are done in the global counter. The third kernel, which corresponds to Step 3, does $O(m''nk')$ total work by using m'' threads doing $O(nk')$ work each. Finally the fourth kernel, which determines the real mutual (k, k') -nearest neighbors in Step 4, requires $O(n'k)$ total work executed by $n'k$ threads doing $O(1)$ work each and preforms T atomic operations.

By summing up all, solving the multiple bichromatic mutual (k, k') -nearest neighbor problem requires $O(kn'm + k'nm'')$ total work.

4 Experimental Results

In this section we present preliminary experimental results obtained with the implementation of the provided algorithm. We used CUDA C, and the executions are done using a i7-3610 2.30 GHz 8 GB of memory and a Nvidia GTX 660 M. The sets R and S , that are used in the experiments, contain points uniformly distributed on a squared domain, and the subsets R' and S' are obtained by randomly choosing points of R and S . The experiments investigate the influence of parameters k, k', n, m, n' and m' . For k and k' values, which in real cases would be chosen by experts, we consider 15 and 30. Parameters n and m vary between 100 and 1000000, n' between 50 and 1000, and m' between 5 and 1000.

In Table 1, the running times needed and the number of significant points S_{sig} obtained when solving several multiple mutual (k, k') -nearest neighbor queries are presented. The running times, measured in seconds, include the transferring times from CPU to GPU and all the time needed to solve the problem. From them we can see how the presented algorithms are scalable. For example, sets with $n = m = 10.000$ and subsets of $n' = m' = 1000$ can be handled for

Table 1. Running times and number of significant points.

$k = k'$	n'	m'	Running times (s)									
			$n = m$				m''					
			100	1000	10000	100000	100	1000	10000	100000		
15	50	5	0.003	0.006	0.009	0.050	0.210	39	3	0	1	0
	100	50	0.003	0.008	0.014	0.030	0.212	49	33	10	2	0
	100	100	0.003	0.007	0.013	0.050	0.212	94	56	10	1	0
	1000	1000	-	0.010	0.021	0.060	0.440	-	896	601	128	19
30	50	5	0.006	0.014	0.026	0.048	0.239	5	4	1	0	0
	100	50	0.007	0.016	0.030	0.072	0.245	48	37	12	2	0
	100	100	0.008	0.016	0.030	0.072	0.442	96	71	29	3	1
	1000	1000	-	0.026	0.048	0.108	0.508	-	946	731	225	23

$k = k' = 30$ with at most 0.048(s). Note that, as the theoretical complexity analysis predicts, the running times increase with k, k', n, m, n' and also, but in least measure, with $m'' = |S_{sig}|$.

5 Conclusions and Future Work

In this paper we presented a parallel algorithm, to be run on a Graphics Processing Unit, for solving multiple bichromatic mutual nearest neighbor queries in the plane considering multiplicative weighted Euclidean distances. We provided the complexity analysis of our algorithm, together with some initial experimental results that show the scalability of our approach.

References

1. Aichert, E., Böhm, C., Kroger, P., Kunath, P., Pryakhin, A., Renz, M.: Efficient reverse k-nearest neighbor search in arbitrary metric spaces. SIGMOD (2006)
2. Brito, M.R., Chavez, E.L., Quiroz, A.J., Yukich, J.E.: Connectivity of the mutual k-nearest neighbor graph in clustering and outlier detection. Stat. Probab. Lett. **35**(1), 33–42 (1997)
3. Barrientos, R.J., Gómez, J.I., Tenllado, C., Matias, M.P., Marin, M.: kNN query processing in metric spaces using GPUs. In: Jeannot, E., Namyst, R., Roman, J. (eds.) Euro-Par 2011, Part I. LNCS, vol. 6852, pp. 380–392. Springer, Heidelberg (2011)
4. Brown, S., Snoeyink, J.: Gpu nearest neighbors using a minimal kd-tree, In: Second Workshop on Massive Data Algorithmics, (MASSIVE) (2010)
5. Cayton, L.: A nearest neighbor data structure for Graphics Hardware. VLDB-ADMS pp. 1–6 (2010)
6. Cheung, K.L., Fu, A.W.-C.: Enhanced nearest neighbour search on the R-tree. SIGMOD **27**(3), 16–21 (1998)
7. Chen, Y., Patel, J.: Efficient evaluation of all-nearest-neighbor queries. ICDE pp. 1056–1065 (2007)
8. Drezner, T.: Optimal continuous location of a retail facility, facility attractiveness, and market share: an interactive model. J. Retail. **70**(1), 49–64 (1994)
9. Drezner, T., Drezner, Z.: Validating the Gravity-Based Competitive Location Model Using Inferred Attractiveness. Annals OR **111**(1–4), 227–237 (2002)
10. Fort, M., Sellarès, J.A.: Finding influential location regions based on reverse k-neighbor queries. Knowl.Based Syst. **47**, 35–52 (2013)
11. Gao, Y., Chen, G., Li, Q., Zheng, B., Li, C.: Processing mutual nearest neighbor queries for moving object trajectories. In: Proc. 9th Int. Conf. on Mobile Data Management, pp. 116–123 (2008)
12. Garcia, V., Debreuve, E., Nielsen, F., Barlaud, M.: k-nearest neighbor search: fast GPU-based implementation and application to high-dimensional feature matching. In: Proceedings IEEE 17th Int. Conf. on Image Processing (ICIP) pp. 3757–3760 (2010)
13. Gowda, K.C., Krishna, G.: Agglomerative clustering using the concept of mutual nearest neighborhood. Pattern Recog. **10**(2), 105–112 (1978)
14. Gowda, K.C., Krishna, G.: The condensed nearest neighbor rule using the concept of mutual nearest neighborhood. IEEE Trans. Inf. Theory **25**(4), 488–490 (1979)

15. Gao, Y., Zheng, B., Chen, G., Li, Q., Chen, C., Chen, G.: Efficient mutual nearest neighbor query processing for moving object trajectories, *Information Sciences*, 180(11), pp. 2176–2195 (2010)
16. Gao, Y., Zheng, B., Chen, G., Li, Q.: On efficient mutual nearest neighbor query processing in spatial databases. *Data Knowl. Eng.* **68**(8), 705–727 (2009)
17. Hjaltason, G.R., Samet, H.: Distance browsing in spatial databases. *ACM Trans. Database Syst.* **24**(2), 265–318 (1999)
18. Jin, W., Tung, A.K.H., Han, J., Wang, W.: Ranking Outliers Using Symmetric Neighborhood Relationship. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) *PAKDD 2006. LNCS (LNAI)*, vol. 3918, pp. 577–593. Springer, Heidelberg (2006)
19. Korn, F., Muthukrishnan, S.: Influence sets based on reverse nearest neighbor queries. *SIGMOD* (2000)
20. Miranda, N., Chávez, E., Piccoli, M.F., Reyes, N.: (Very) Fast (All) k -Nearest Neighbors in Metric and Non Metric Spaces without Indexing. In: Brisaboa, N., Pedreira, O., Zezula, P. (eds.) *SISAP 2013. LNCS*, vol. 8199, pp. 300–311. Springer, Heidelberg (2013)
21. Stanoi, I., Riedewald, M., Agrawal, D., Abbadi, A.E.: Discovery of influence sets in frequently updated databases. In: *Proceedings 27th Int. Conf. on Very Large Data Bases (VLDB)* pp. 99–108 (2001)
22. Wong, R.C.-W., Tao, Y., Fu, A.W.C., Xiao, X.: On efficient spatial matching. In: *Proceedings 33rd International Conference on Very Large Data Base*, pp. 579–590 (2007)
23. Wu, W., Yang, F., Chan, C.Y., Tan, K.: FINCH: evaluating reverse k -Nearest-Neighbor queries on location data. In *Proceedings of VLDB 1*(1), pp. 1056–1067 (2008)
24. Yao, B., Li, F., Kumar, P.: K -nearest neighbor queries and knn-joins in large relational databases (almost) for free. In *Proceedings of ICDE 2010*, pp. 4–15 (2010)
25. Zhang, J., Mamoulis, N., Papadias, D., Tao, Y.: All-nearest-neighbors queries in spatial databases. In: *Proceedings of 16th International Conference on Scientific and Statistical Database Management (SSDBM)*. pp. 297–306 (2004)

A Kernel Density Method for Aggregating Boundary Collision Data into Areal Units

Ge Cui¹(✉), Xin Wang¹, and Dae-Won Kwon²

¹ Department of Geomatics Engineering, University of Calgary,
Calgary, Canada

{cuig, xcwang}@ucalgary.ca

² Office of Traffic Safety, City of Edmonton, Edmonton, Canada
dae-won.kwon@edmonton.ca

Abstract. Boundary collisions are motor accidents which occur on the boundaries of areal units. In some areas, boundary collisions may account for a large proportion of the total collisions. Generally, it is a critical step to aggregate boundary collisions into areal units before collision analysis. Exaggerated or underestimated aggregation results may hamper traffic safety analysis and management. In this paper, we propose a boundary collision aggregation approach based on the collision density ratio. The proposed method is compared with two other boundary collision aggregation methods. Two regions (downtown region and south region) in City of Edmonton are selected as the study area. The assessment result shows that the proposed method outperforms the other two methods.

Keywords: Boundary data aggregation · Collision density ratio · Boundary zone size

1 Introduction

Transportation collisions are the leading cause of death in Canada. Transportation collision threatens many people's life, resulting in a lot of injuries and even fatality. For example, in the City of Edmonton, there were 23,442 collisions in 2011 and 23,237 collisions in 2012 [5]. For requirements of traffic analysis or traffic management, collision data are often aggregated into area units, such as neighborhoods. Therefore, aggregating collisions into areal units has a large impact on traffic analysis and management.

Boundary collisions are motor accidents which occur on the boundaries of areal units. As boundaries of neighborhoods are generally main roads where most collisions happen, boundary collisions usually account for a large proportion of the total collisions. Hence, traffic analysis and management would be remarkably affected by these boundary data, which is called the *boundary effect*.

Boundary effect is often determined by the following factors: (1) the accuracy of reported spatial boundary collisions; (2) the method of aggregating boundary data into adjacent areal units. The first factor is related to the data collection process, which is not considered in this study. As for the second factor, some study has been conducted.

Some researches suggest that boundary effect have little influence on traffic analysis. Ladrón de Guevara et al. and Khondokar et al. examined the issue when they developed macro-level collision models of Tucson, Arizona, USA and Greater Vancouver, Canada successively. They found that the number of collisions involved on traffic analysis zones (TAZs) boundaries was about 5 %, which did not significantly impacts their collision models results [2, 3].

Nonetheless, other studies indicate that boundary effect might affect traffic analysis. Fotheringham and Wegner observed that spatial data located near zone boundaries might have an inter-zonal influence [1]. Lovegrove aggregated the boundary data in an automatic geo-spatial precision way in developing a macro-level collision distribution model for the Great Vancouver Regional District, Canada and supposed a potential boundary effect in aggregating geo-coded data [4]. Feng made use of 5 boundary data aggregation approaches for geo-coded data on TAZ boundaries and built 8 groups of collision prediction models (CPMs), concluding that the different aggregation methods for boundary data impact the CPM results significantly [7].

In this paper, it aims to conduct a preliminary study to identify whether collisions around boundaries play a significant role, in terms of quantity, in road safety in Edmonton. We propose a collision density ratio method to aggregate boundary data into areal units.

2 Problem Statement

Currently, it can be often observed that boundaries are on conspicuous natural or artificial ground objects, such as rivers, roads, rails and trails. For neighborhoods, roads are often adopted as their boundaries. Hence, boundary collisions are actually those taking place in the roads which are taken as boundaries of neighborhoods. However, boundary collisions might be difficult to identify after digitalization because, (a) boundaries may not be coincide with the corresponding roads, which leads to boundary collisions are not located on the boundary lines; (b) representing real world features into features in a GIS (e.g. 10 m width roadways are often represented single lines in GIS), may result in that collisions deviate from the roads. Problems (a) and (b) are shown in Fig. 1, where roads are solid lines, boundaries are dash lines and collisions are black points.

To deal with the problem, boundary zone, a buffer centering at the boundaries of neighborhoods, is proposed. Collisions, located within the boundary zone, are assumed as boundary collisions. One issue of boundary zone is how to determine its size (buffer size). It is obvious that some non-boundary collisions will be contained mistakenly if the boundary zone size is set too large. These non-boundary collisions are usually not far from the boundary but they occur on the roads which should not be used as the neighborhood boundary. Therefore, it requires a criterion for setting a proper boundary zone size to guarantee most boundary collisions and least non-boundary collisions are contained in the buffer.

After identifying boundary collisions, it needs to aggregate them into neighborhoods around. One-to-One ratio method and Half-to-Half ratio method are two commonly used aggregation methods proposed in [7]. Nonetheless, it seems that these methods fail to

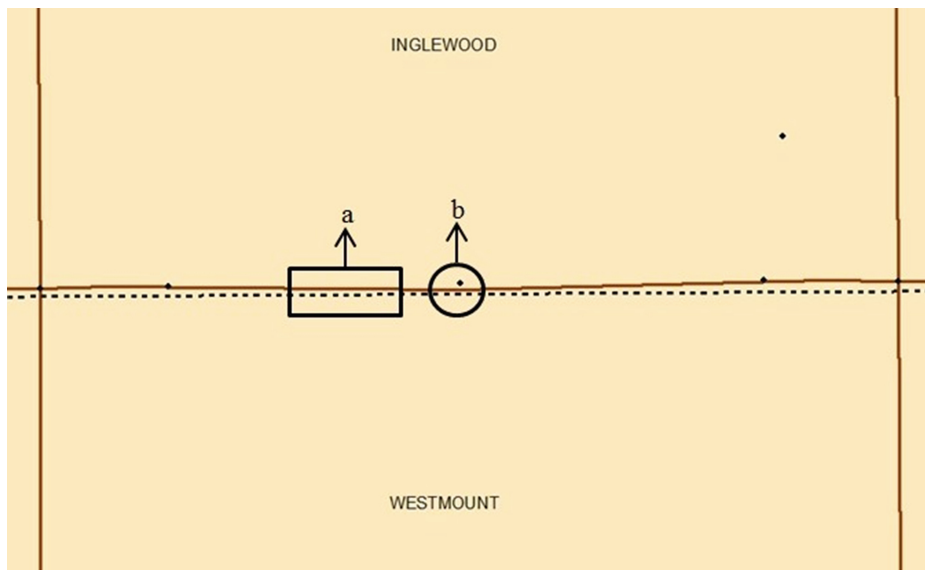


Fig. 1. Boundary collisions are difficult to identify after digitalization.

deal with the aggregation very well. One-to-One ratio method counts the boundary collisions once for every neighborhood adjacent to the boundary areas. If a collision (four neighborhoods around it) is located at a four-way intersection, the collision will be counted four times. Apparently this method will exaggerate the collision number in total. Half-to-Half ratio method works in a similar way as One-to-One ratio method, but it just counts a collision once and assigns the collision value to the neighborhoods by average. The weakness of this method lies at that it treats all neighborhoods evenly when considering the impact of neighborhoods on boundary collisions. However, after scrutinizing the distribution of collisions around the boundary, there exists a very common collision distribution pattern around the boundary (as shown in Fig. 2), which shows the variation of impact of neighborhoods on boundary collisions.

In Fig. 2, it could be observed that many collisions are located around the boundary in the north neighborhood, while almost no collisions occur around the boundary in the south neighborhood. One possibility is that the driving conditions (such as road surface, street lamp and etc.) of north neighborhood are poorer than that of south neighborhood. Furthermore, it means driving conditions of neighborhoods may have different degrees of impact on a boundary collision.

3 Methodology

3.1 Boundary Zone Size Determination

In this section, we aim to determine a suitable boundary zone size to maintain correct boundary data. Firstly, it needs to exclude the non-boundary collisions. We assume that the distance $D_{c,b}$ between a non-boundary collision location and the boundary of

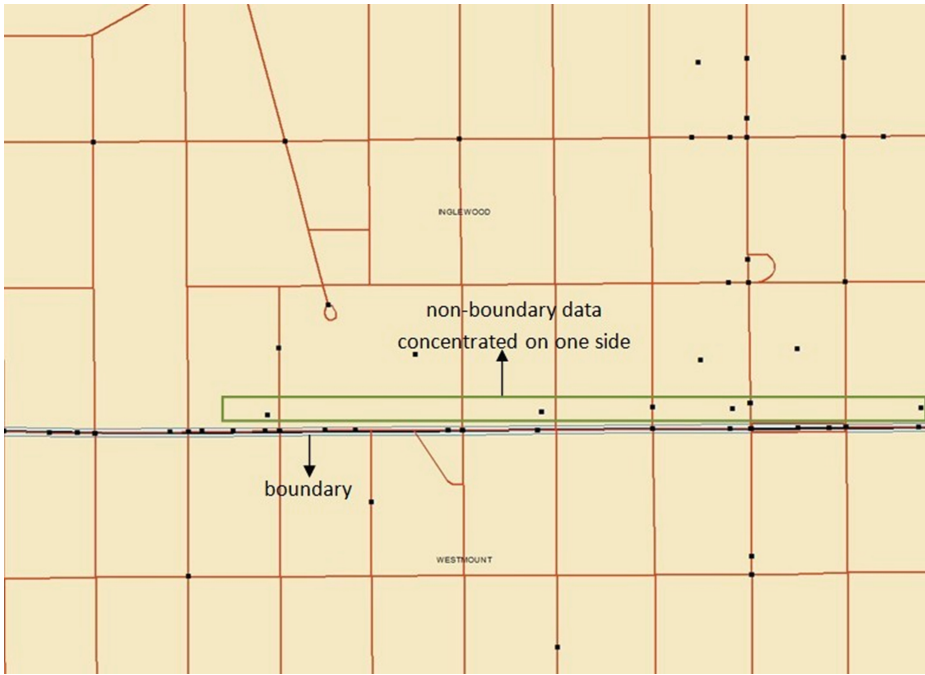


Fig. 2. Collisions are concentrated on one side of the boundary

the neighborhood is greater than the distance $D_{c,r}$ between its location and its nearest road. Therefore, the collisions whose $D_{c,b} > D_{c,r}$ will be classified to non-boundary collisions, which will be ignored in our study. Secondly, there are some outliers which are non-boundary collisions but still satisfy the condition $D_{c,b} < D_{c,r}$ (as shown Fig. 3). These outliers should be excluded as well.

To remove the outliers and determine the boundary zone size, a histogram will be created to represent the distribution of $D_{c,b}$ of collisions in the buffer zones. The horizontal axis is the distance interval between collisions and neighborhood boundaries, and the vertical axis is the frequency of collisions falling into the interval. As most non-boundary collisions have been filtered, the frequency of outliers is quite low. Therefore, there is a drastic drop of collision frequency when the distance reaches a threshold which is selected as the boundary zone size.

3.2 Collision Density Ratio Method

In this paper, a collision density ratio method is proposed to aggregate boundary data. This method considers the interactions of proximate collisions. It assumes the fact that collisions have a specific density distribution pattern due to the different driving conditions of the surroundings. For some spots with poor driving conditions, the collision density may be higher relatively. Furthermore, every areal unit has a

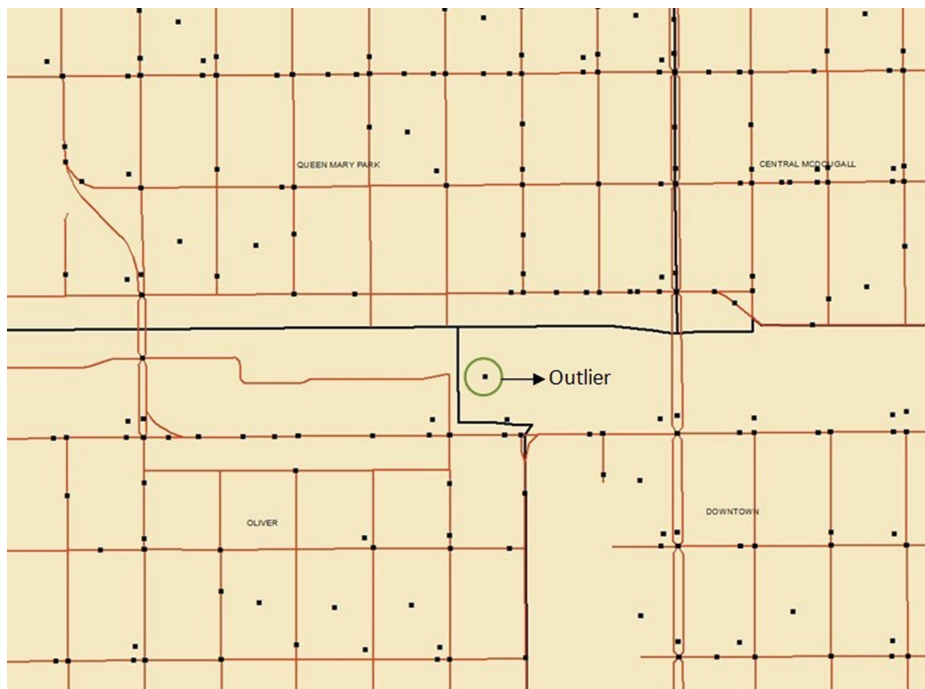


Fig. 3. The outlier of the boundary collisions

collision density distribution. As a boundary collision could be taken as one part for all adjacent neighborhoods, it has distinct density value from the collision density distributions of different neighborhoods. If a collision location gets a high density value from a neighborhood, it means the neighborhood has a large influence on the collision occurrence at this spot. Therefore, a collision is assigned to the neighborhoods by the ratio of their impacts on this collision.

Collision density distribution is extended from kernel density estimation approach. The kernel function is based on the quartic approximation of a true Gaussian kernel function [6]. For density distribution of each collision, the density value obey Gaussian distribution, rising to the peak at the location of the collision and decreasing with growing distance from the collision point, reaching zero at the search radius distance from the collision point. A continuous smooth surface is fitted over each collision. For every neighborhood, its collision density distribution could be deemed to be the mutual contribution of inner zone and boundary zone as shown in Fig. 4. In this figure, there are two adjacent neighborhoods A and B. The **orange line** stands for the common boundary, and the region within **red edge** is the boundary zone. The inner zone is the rest part of neighborhood by removing boundary zone. The collision density distribution of neighborhood A and B will be established based on **black collisions** and **red collisions**, respectively.

The aggregation principle of the collision located in the boundary zone is as follows: for any collision in the boundary zones of neighborhoods, the assigned

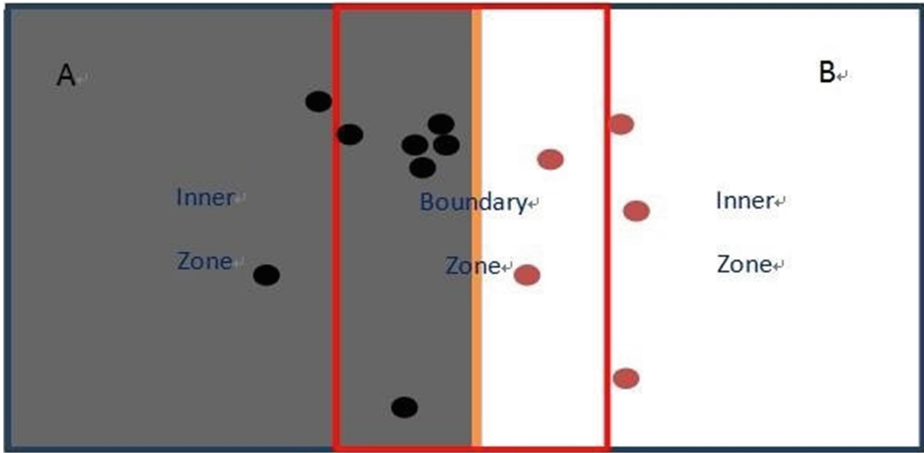


Fig. 4. The structure of a neighborhood: boundary zone and inner zone (color figure online)

collision value into the corresponding neighborhood is based on the ratio of the collision density:

$$V_{N_i} = \frac{D(x, y|N_i)}{\sum_{i=1}^m D(x, y|N_i)} \tag{1}$$

where V_{N_i} is the assigned value of the collision to the neighborhood N_i ; x, y is the coordinate of the collision; $D(x, y|N_i)$ is the collision density of the location (x, y) estimated from neighborhood N_i .

3.3 Evaluation Method

It is very important to validate the effectiveness of boundary collisions aggregation methods. The common way is that the accuracy of collision prediction models (CPMs) established on aggregation results could reflect their effectiveness. In this research, we provide an alternative statistical hypothesis test method. As a critical criterion for collision aggregation is that the number of collisions aggregated into neighborhoods should not be exaggerated or underestimated too much, the two sample paired t-test could be applied to evaluate whether the average difference between before and after aggregation is greater than zero or not. The number of collisions in each neighborhood before aggregation could be obtained by counting the collisions within the boundary line.

It is a two-tailed test since we are trying to prove that there is a difference between the number of collisions before and after aggregation. As the number of samples is larger than 30, it could be assumed that a t-distribution that approaches a normal curve with $n - 1$ degrees of freedom where n is the number of the observations. A $\alpha = 0.05$ confidence level is used.

4 Experiment

4.1 Data Description

The study areas are selected from Edmonton (the capital city of Alberta), consisting of two regions: downtown region and south region. These two regions are covered by dense road network and collision accidents take place frequently there.

(1) Downtown region. Neighborhoods include: McCauley, Queen Mary Park, Central McDougall, Westmount, Downtown, Oliver, and Boyle Street.

(2) South region. Neighborhoods include: Parsons Industrial, Calgary Trail South, Rideau Park, Duggan, Strathcona Industrial Park, Steinhauer, and Ermineskin.

As it aims to study the boundary data of the study areas, the neighborhoods around the study areas should be utilized in the experiment either. Hence, there are 21 neighborhoods in downtown region, and 25 neighborhoods in south region (Fig. 5).

The collision data of Edmonton from year 2006 to year 2012 are utilized in the experiment.

4.2 Data Analysis

Table 1 illustrates the distribution of the distances between collisions and boundaries. It could be observed that there are a large amount of collisions which are located near the boundaries. There are 25 % collisions for which the distances to boundaries are less than 1.006 m in downtown area, and 2.079 m in south area. The boxplots (as shown in Fig. 6) show that lots of collisions are densely distributed around the boundary. Therefore, the conclusion could be drawn that boundary collisions could take account for a large proportion of total collisions in the study areas.

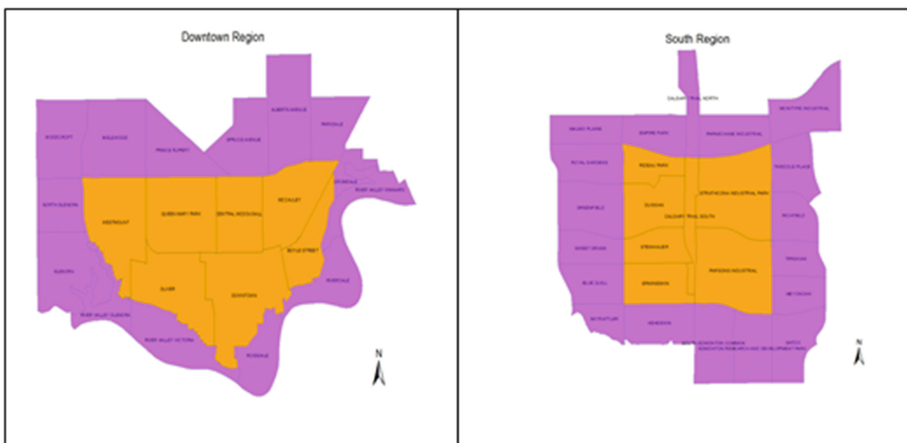


Fig. 5. The experiment regions: downtown region and south region

Table 1. Statistics of distances (m) between collisions and boundaries in downtown region and south region

	Downtown area					South area				
	Min.	1st quar.	Medium	3rd quar.	Max.	Min.	1st quar.	Medium	3rd quar.	Max.
2006	0	0.964	89.803	217.16	608.89	0	2.079	10.536	65.159	737.46
2007	0	0.921	83.086	215.04	608.42	0	2.079	13.800	75.388	737.46
2008	0	1.208	90.673	227.99	610.37	0	2.079	13.771	65.825	737.46
2009	0	1.496	95.806	232.89	606.96	0	2.101	14.078	83.707	737.46
2010	0	1.087	83.086	216.91	608.89	0	2.101	22.085	86.565	737.46
2011	0	0.923	76.115	214.89	606.96	0	1.708	12.766	73.517	737.46
2012	0	1.017	83.086	216.71	608.42	0	2.049	12.766	68.482	737.46
Total	0	1.006	87.700	220.64	610.37	0	2.079	13.164	73.656	737.46

4.3 Boundary Zone Size

With the criterion $D_{c,b} < D_{c,r}$ that the distance between boundary collision and boundary should be less than the distance between boundary collision and its nearest road, most non-boundary data are removed. The histograms as shown in Figs. 7 and 8 present the distribution of the distances of boundary collisions and boundaries. The horizontal axis represents the distance interval and the vertical axis represents the collision frequency which $D_{c,b}$ falling into the intervals. The unit of the distance interval is set as 1 m.

There are clear thresholds for the two histograms. For downtown region, the optimum buffer distance is 5 m; for south region, the buffer distance is around 8 m. For the two figures above, the majorities of boundary collisions are densely concentrated within 1 m distance from the boundary. The number of collisions declines with the increase of the distance to boundary. Beyond a certain threshold of distance, the boundary data turns very rare, and they could be taken as outliers. Boundary collisions in downtown region ($D_{c,b} < 5$ m) occupy the percentage of 30.80 % and in south region ($D_{c,b} < 8$ m) take the percentage 44.40 %.

4.4 Boundary Collision Assignment Results

In this paper, three boundary collision aggregation methods (collision density ratio method, One-to-One ratio method and Half-to-Half ratio method) are utilized and their results are compared. As downtown region and south region contains 7 neighborhoods respectively from 2006 to 2012, there are 49 observations in total for every region. Figure 9 gives an example of the aggregation result in QUEEN MARY PARK, downtown region.

In Fig. 9, “Original” represents the collision number in the neighborhood before applying the aggregation method. The results of three aggregation methods are showed in the figure as “Collision density ratio”, “Half-to-Half ratio” and “One-to-One ratio” respectively.

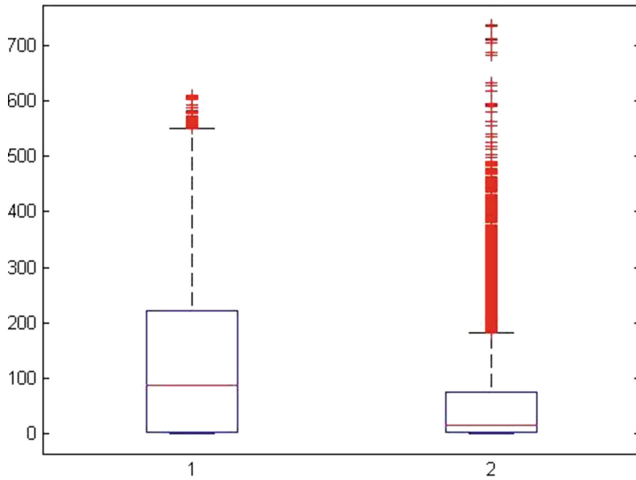


Fig. 6. The boxplot of downtown region (left, 1) and south region (right, 2)

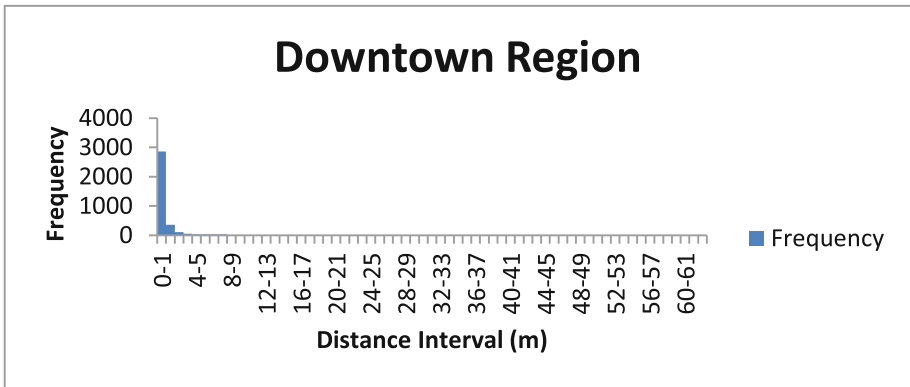


Fig. 7. Downtown region: the histogram about the distribution of collision frequency

4.5 Evaluation Results

The results of two samples paired t-test for downtown area and south area are shown in the Tables 2 and 3, respectively. For neighborhoods in each region, the numbers of collisions before and after using aggregation method are compared. In downtown region, the p-value is far less than the significance level 0.05 with Half-to-Half ratio method and One-to-One ratio method, so the null hypothesis could be rejected safely and there is a significant change of the number of collisions after applying the two aggregation methods. For collision density method, as the p-value is larger than 0.05, we cannot say there is a large variance of collision number with this method. In south region, it shows there is a significant change of collision number with One-to-One ratio method and seems slight variance after using collision density ratio method and Half-to-Half ratio method.

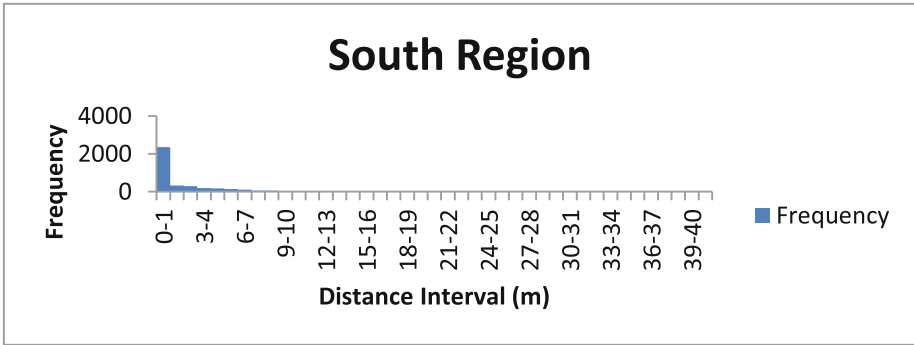


Fig. 8. South region: the histogram about the distribution of collision frequency

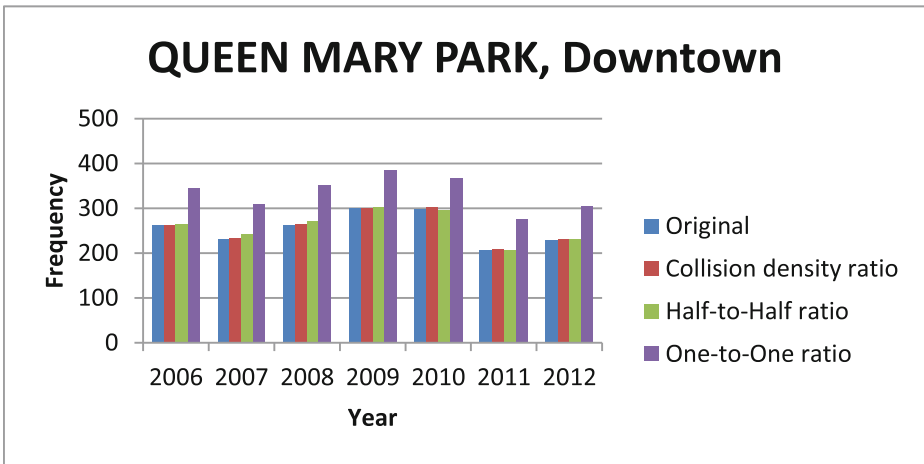


Fig. 9. The aggregation results of QUEEN MARY PARK in downtown region with 5 m buffer

Table 2. Two samples paired t test of collision number before and after aggregation for downtown region, 5 m buffer

	Mean of differences	Std. of differences	Degree of freedom	Test statistic	P-value
Collision density ratio	1.080	8.118	48	0.930	0.357
Half-to-half ratio	9.721	18.108	48	3.758	0.000
One-to-one ratio	89.010	57.202	48	10.892	0.000

Table 3. Two samples paired t test of collision number before and after aggregation for south region, 8 m buffer

	Mean of differences	Std. of differences	Degree of freedom	Test statistic	P-value
Collision density ratio	2.807	13.997	48	1.404	0.167
Half-to-half ratio	4.806	23.645	48	1.422	0.161
One-to-one ratio	95.541	65.513	48	10.315	0.000

5 Conclusion and Discussion

For the two study areas in Edmonton, downtown region and south region, the distribution of distances between neighborhoods' boundary and collisions shows that more than one quarter of the total collisions occur around the neighborhoods' boundary. Therefore, to process boundary collisions is quite significant for traffic safety analysis in the study areas of Edmonton.

As boundary collisions happen on the roads which are adopted as the neighborhoods' boundary, generally non-boundary data can be filtered out by comparing the distance to its nearest road and the distance to the boundary. With a histogram of boundary collision distribution, the boundary zone size is set as 5 m for downtown region and 8 m for south region. The possible explanation for the different buffer distances may result from the fact that the widths of the roads in these two regions are different. In most situations, roads in downtown region are a bit narrower than the roads in other regions. Therefore the boundary zone size in downtown region is smaller than that in south region correspondingly.

The two samples paired t-test is used to evaluate the effectiveness of three boundary collision aggregation methods. For One-to-One ratio method, it is very clear that p-value for both downtown area and south area approximates to zero, and this method actually exaggerates the number of collisions for every neighborhood, which is the deficiency of this method. Half-to-Half ratio method is not stable enough. For neighborhoods of downtown region, the small p-value indicates there is a large variance between the collision number after aggregation and the true value, which means the method would cause inaccuracy; while it has a p-value exceeding the significance level 0.05 and achieves a great performance in south region. For collision density ratio method, the p-value for both downtown region and south region is larger than 0.05, and therefore, there is little variance of collision number compared with true value by using this method. On the other hand, the higher the t-value is, the more likely it is that the two means are different. Out of the three methods, collision density ratio method gets the lowest t-value, which means this method would less likely to exaggerate or underestimate the number of collisions in neighborhood compared with the two other methods.

6 Future Work

In this paper, the boundary buffer size is selected when there is a drastic drop of collision frequency in the histogram. In future research, we will try to provide a

quantitative criterion to determine the buffer size. Besides, in addition to the statistical method used in this paper, we will use macro-level CPMs to give a solid evaluation about the effectiveness of the aggregation method.

References

1. Fotheringham, A.S., Wegener, M.: *Spatial Models and GIS: New Potential and New Models*. Taylor & Francis, London (2000)
2. Khondakar, B., Sayed, T., Lovegrove, G.: Transferability of community-based collision prediction models for use in road safety planning applications. *J. Transp. Eng.* **136**(10), 871–880 (2010)
3. Ladron de Guevara, F., Washington, S.P., Juteak, O.: Forecasting crashes at the planning level: simultaneous negative binomial crash model applied in Tucson, Arizona. *Transp. Res. Rec.* **1897**, 191–199 (2004)
4. Lovegrove, G.: *Road Safety Planning: New Tools for Sustainable Road Safety and Community Development*. Verlag Dr. Müller, Germany (2007)
5. OTS. Motor vehicle collisions (2012). http://www.edmonton.ca/transportation/traffic_safety/motor-vehicle-collisions.aspx
6. Silverman, B.W.: *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, London (1986)
7. Wei, F.: *Boundary effects in developing macro-level CPMs: a case study of city of Ottawa*. Civil Engineering, University of British Columbia, Okanagan (2010)

Integrated Indoor Positioning with Mobile Devices for Location-Based Service Applications

Bei Huang and Yang Gao^(✉)

Department of Geomatics Engineering, Schulich School of Engineering,
University of Calgary, 2500 University Dr. N.W. Calgary,
AB T2N 1N4, Canada
{huanb, ygao}@ucalgary.ca

Abstract. Location information in both outdoor to indoor environments is essential for location-based service applications from critical services such as E911 and emergency fleet management to daily activities such as personal navigation and social networking. This paper describes recent technology advances and limitations for indoor positioning and the need for integrated solution for improved accuracy and availability. An integrated system based on Wi-Fi, camera image and floor plan database is proposed for indoor positioning and the test results demonstrate significantly improved system performance.

Keywords: Indoor positioning · Location-based service · GPS · Wi-Fi · Camera image · Floor plan

1 Introduction

The concept of location-based services (LBS) originates from E911 project decades ago, which were applied for dispatching, tracking and managing emergency resources. As the most popular and reliable positioning system, GPS serves E911 by finding accurate location information in real-time. Driven by continuous efforts on the miniaturization of GPS receivers, very small chipsets and OEM boards are becoming increasingly available to meet the design constraints of mobile devices including smart phones. In the past decade, GPS started to find wide use in people's daily life with handset LBS applications. For mobile GPS/GIS handhelds with good quality of antenna, high precision positioning at dm to cm is also feasible today using the latest precise point positioning (PPP) technologies. The impact on GIS applications is significant since precise GIS data acquisition can be carried out in real-time which improves the sharing and collaboration between the field and the office and has made mobile GPS/GIS applications one of the fastest growing markets using satellite navigation technology. To extend their applicability into more challenging environments such as forest and urban canopy and even indoor environments, the GPS chipsets on mobile devices are all implemented with high-sensitivity GPS signal acquisition technologies. But high sensitivity GPS does not work in all indoor environments because GPS signals can be heavily attenuated by the building materials or

reflected as in multipath, which will make a viable position solution impossible. In recent years, mobile devices are also increasingly installed with various enabling sensors such as barometer, gyroscope, compass, accelerometers, digital camera which can be applied to derive location information at indoor environments when no GPS signals are available.

In addition to positioning based on various navigation sensors, wireless communication networks have been widely used to provide location information at indoor environments. Currently, the most popular indoor positioning system is Wi-Fi due to the system's dense hot spots. Although the accuracy of Wi-Fi is not yet satisfactory, the great indoor availability of Wi-Fi provides great potential to further improve Wi-Fi positioning accuracy by adding other enabling sensors and database. A fusion of data from all available sensors in mobile devices and integrated with wireless networks presents a promising solution for seamless indoor/outdoor positioning in the future.

In this paper, an integrated system based on Wi-Fi, floor plan and digital camera is proposed for indoor positioning. The integrated system first uses Wi-Fi position as the initial location which is usually not precise, and the system then uses the camera image and floor plan database to provide refined positioning solutions. An iOS App has been developed and implemented into iPhone and iPad to demonstrate the applicability for practical applications. The positioning accuracy by the integrated system can be improved from tens of meters with Wi-Fi to about 5 m, which can satisfy most indoor LBS application requirements.

2 LBS Applications and Positioning Methodologies

While LBS mobile Apps have been around for years, big brands including Facebook, Foursquare etc. all catch up with market demand to introduce their reward programs with new LBS features. The major chipset manufactures such as Qualcomm, Marvell and CSR, delivering Wi-Fi, GPS and communication chipsets to most smart devices on market, are also adding integrated positioning algorithms as the built-in feature to provide versatile interface and seamless positioning solution for LBS developers. Some top pick LBS Apps voted by millions of users have redefined people's lifestyle and boosted local market: Yelp is the original LBS App that featured for its comprehensive database for local business including restaurant, bar, apparel, saloon, grocery etc., which is tagged with geodetic locations, customer reviews, photo gallery and contact information. Best direction is also suggested to place immediate connection between customers and stores. Point Inside, a small but thriving LBS company, focuses on Wal-Mart alike for their giant grocery markets, which push out discount voucher notifications and accurately lead customers to find individual item out of huge product inventories. Besides these practical E-business LBS Apps, the rise of geosocial networking gives the idea of sharing user's location and activity to friends network. The traditional social networking platforms like Facebook and Twitter recently join the LBS campaign by enabling geo-tagged posts. The most trendy lifestyle and fitness Apps such as Fitbit, Jawbone, MotoActv by Motorola and FuelBand by Nike have innovated personal activity tracker Apps for smart phone, as well as creating a brand new personal portable devices market for intelligent watch

and wristband. In another word, user locations recorded by smart devices exhibits certain structural patterns of customer behavior, distribution, preference and other valuable information for LBS Apps. Eunjoon et al. [1] has studied the users' mobility and network through Wi-Fi and cellular based locations which provides practical model for LBS Apps to promote popularity. An accurate positioning and navigation engine is the premise of delivering qualified location-based services in above-mentioned Apps. Considering the fact that the most user active areas are GPS-denied environments such as indoor and urban canyon, Wi-Fi positioning has become the most popular technology that most LBS Apps are relying on to provide position information with excellent availability. Wi-Fi positioning is especially applicable on low-cost handheld platform because of low power consumption and large coverage of hot spots. Wi-Fi positioning is based on the finger printing technology, in which a database of highly dense Wi-Fi hot spots with geodetic positions is employed. The Wi-Fi signals scanned by user devices are considered as a unique map of the local Wi-Fi hot spot distribution and are compared to the Wi-Fi database to derive user location. However, due to the complexity of indoor structures and uneven distributions of the Wi-Fi hot spots, the finger print quality could be significantly degraded because of multipath and signal attenuation etc.

To improve Wi-Fi's accuracy, researchers have attempted to integrate other sensors requiring no Radio Frequency (RF) signals to enhance indoor performance. One widely applied is inertial measurement unit (IMU). With the fast development of Micro-Electro-Mechanical Systems (MEMS) technology, MEMS-based IMU sensors such as accelerometer and gyroscopes nowadays are also integrated into smart devices as chipsets. These sensors, originally designed to provide motion measurements for gaming applications, are now implemented to assist pedestrian indoor navigation. For example, the above-mentioned Point Inside has adopted GPS/Cellular/Wi-Fi and MEMS IMU integrated algorithm to achieve satisfactory indoor positioning performance. When users search their product inventory, Point Inside retrieves the item location from database and plans the best route from user location. In order to lead user to the correct item, MEMS IMU provides continuous and extra-smooth solution on the basis of Wi-Fi positioning. But in order to avoid accuracy degradation due to attenuated Wi-Fi signal and noisy IMU measurements, Point Inside suggests business partners to install their indoor signal anchors to improve accuracy.

More methods have been developed to improve the performance of using IMU to measure user motion. Melania [2] developed a pedestrian gait estimation method, in which the inertial measurements are used to estimate the pedestrian gait frequency and lengths of strides. Moreover, with special focus on providing context of unpredictable human movements encountered by most LBS Apps, [3] introduced an algorithm based on IMU to be aware of different scenarios such as walking, reading, stationary, in pocket etc. But these systems suffer from accumulative sensor bias and drift especially when using the embedded low-cost MEMS sensors in pedestrian devices which would result in positioning errors at tens of meters in less than five minutes run. As LBS applications demand indoor location accuracy at room-level or aisle-level, it is still a significant challenge to apply Wi-Fi and motion sensors to meet this requirement.

Given the limitations indicated above with the Wi-Fi and motion sensors, vision navigation technology developed for robotic intelligence applications provides

potential to improve indoor positioning accuracy [4]. Various researches have applied digital cameras, which are available in mobile devices, to aid navigation because it brings not only enhancement to user experience but also positioning accuracy [5, 6]. Geo-reference database has been frequently combined with vision measurements to assist retrieving user location in near real-time. A geo-tagged photo database is employed with a single camera to easily derive user location by matching a snapshot to the database. Huang and Gao [7] has employed an indoor robotic vehicle mounted with stereo cameras, tactical grade IMU, Wi-Fi and GPS receiver to go through an experiment area to collect indoor photos tagged with the ego-motion of the robotic vehicle. When users visit this area and take a snapshot with their smart phone, this photo is uploaded and compared with the geo-tagged photos in database to derive the user position and orientation. Another good example of geo-tagged photo database is the Google Street View, which uses survey vehicle equipped with GPS, laser scanner, camera and IMU to drive through streets and collect geo-tagged photos. This program has extended their database to other places which uses survey backpack, trolley, bike and even snowmobile. Google also encourages individuals and organizations to contribute geo-tagged photos to their database by lending them survey products for free. The aim is to collect the global database in a collaborative manner to dramatically increase data coverage while reducing collection time. With a comprehensive geo-tagged photo database, the traditional map is revolutionarily substituted by stitching the real photos to form the real 3D view. However, the investment on collecting indoor geo-tagged photo database is enormous, which requires high-end sensors to generate accurate geo-tags. As a result, few practical application of indoor geo-tagged photo database can be found and new methods are required for indoor positioning with improved accuracy.

3 An Integrated Indoor Positioning System

3.1 Generation of Floor Plan Database

Floor plan is a scaled drawing that depicts the indoor arrangement of rooms, hallways and other indoor objects. The scale of a floor plan comes from real world measurements of lengths, angles and geodetic coordinates collected by survey equipment and the accuracy of measurements is typically at decimeter level. As for construction and facility maintenance, every building has stored a floor plan in online server. So comparing with other geo-reference database such as geo-tagged photos, using floor plan database saves the labor, time and cost of survey tasks since it can be generated from existing resources. Furthermore, wireless connection through Wi-Fi and 3G networks enables user download floor plan as an indoor map to supplement the insufficiency of traditional outdoor maps like Google Map. Again, taking the Point Inside as a commercial LBS example, they have been delivering reliable and accurate indoor navigation solutions for years with worldwide coverage of floor plans in major shopping malls and airports as shown in Fig. 1. The enrolled business partners who want to subscribe this innovative indoor LBS service are required to upload their floor plans. With this successful commercialized example of indoor navigation, building a



Fig. 1. Floor plan database worldwide coverage of the LBS App, Point Inside (source: <http://www.pointinside.com/solutions/mapped-locations/>)

floor plan database to support considerable area of interest has great feasibility in practice.

Although lots of commercials provide floor plans for customers as indoor map, these floor plans are not survey-level floor plans and do not contain much geo-reference information. As a result, most of such indoor maps merely serve as pictures of indoor structure, which cannot be used to improve the indoor positioning accuracy. In order to develop a ubiquitous system no matter if the commercials have survey-level floor plan database or not, the procedures to generate geo-reference information for a newly added floor plan and updating an old floor plan database should be easily implemented in practice. The example of the Google Floor Plan project launched by Google Maps team has provided a standard paradigm for generating floor plan database. The Google Floor Plan project is a milestone for traditional outdoor Google Maps, which aims to realize its outstanding outdoor map and service for the indoor floor plans. The procedure of generating a customized floor plan and geo-reference information includes five steps:

- Step 1: Search the target building on Google Maps and select the floor layer to add a floor plan;
- Step 2: Upload the floor plan picture, which is widely available for download in most commercials such as shopping mall, airport, library, hospital etc.;
- Step 3: Fix several landmarks on the floor plan picture, and these landmarks should be visible on the framework of the building, for example, the wall corners;
- Step 4: Stretch the landmarks to coincide with the common points on the outdoor Google Maps, to align the floor plan with the outdoor map as shown in Fig. 2;
- Step 5: Image processing is applied to improve the alignment between the floor plan picture and the outdoor Google Maps, and a customized floor plan with geo-reference information is then generated.

The above mentioned procedure for generating a floor plan database required in our developed system has addressed at least 3 limitations suffered by the database collection for other popular LBS Apps: (1) No field survey is required, which saves the



Fig. 2. Google Maps floor plan program example to add floor plan to traditional outdoor Google Maps

investments of expensive survey equipment and labors. (2) To combine the existing Google Map and service subscriber's floor plan, the procedure consists of manually adjustments and image processing, which significantly shorten the processing time required by the field survey based database collection. (3) Updating database according to business partner's renovation is immediate that merely requires their provision of new floor plans used in the constructions. However, to achieve the improved accuracy of our system, several aspects are investigated:

- Accuracy of the Google Map: The Google Map data source comes from their survey vehicle equipped with high-quality GPS antenna, tactical grade IMU and panoramic cameras. Typical open sky accuracy of the data source is at centimeter level when GPS signal is continuously visible. However, in urban canyon environments where GPS visibility is badly affected, the average accuracy could still be as good as the decimeter level while the maximum error is close to 5 m. In this paper, a decimeter level accuracy is required for the Google Map although the areas with large errors will encounter outlier or even failure.
- Accuracy of floor plan picture: The uploaded floor plan picture is expected to be accurately proportional to the reality. The scaled-up floor plan picture is required to have a decimeter level accuracy, otherwise outlier or even failure will occur.
- Accuracy of the overlaying: To reduce the probability of human error, the process to overlay the floor plan with the Google Map subjects to manual adjustment and image processing should be repeated by randomly choosing landmarks (see description in Step 3), and average the overlaying results.

3.2 Geo-Reference Information of Floor Plan and Indoor Hallway Features

From the abovementioned paradigm of how Google Maps add floor plans to outdoor maps, there is some important geo-reference information available in the generated floor plan database. Using the geo-reference information summarized in the following, a floor plan frame can be established as shown in Fig. 3:

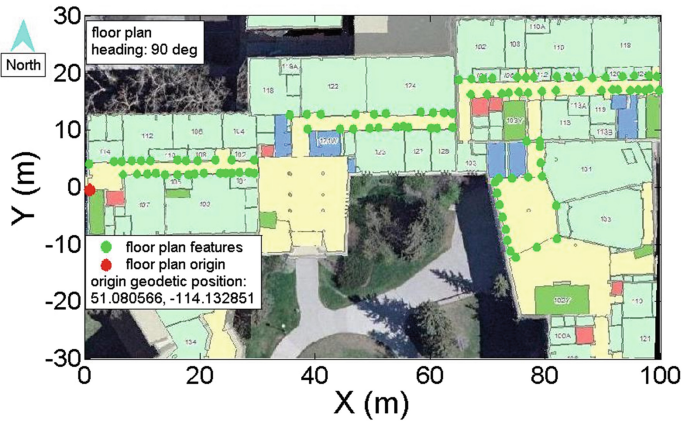


Fig. 3. Floor plan frame and geo-reference information

- Real scale of floor plan: Fixing three landmarks on floor plan picture to coincide with corresponding points on Google Maps produces the geodetic positions of these landmarks. Stretching the floor plan picture to align with Google Maps gives the range of latitude and longitude in the region covered by floor plan. Transforming the range of the spherical geodetic positions to the East-North-Up frame, the real scale of the floor plan picture can then be obtained, and the accuracy of the real scale of floor plan is typically good at a decimeter level. As shown in Fig. 3, the scale of its axes is determined by the real scale of floor plan;
- Geodetic position and heading of floor plan: An origin is selected on floor plan, shown as the red dot in Fig. 3, and its geodetic position is available by referring to the Google Maps. Furthermore, once the axes of the floor plan frame are determined, the heading of the floor plan is also available. In the example shown in Fig. 3, the heading angle is the angle between the x-axis and the true North, which is 90 degree. With the local origin and heading, the coordinates transformation between the floor plan frame and the geodetic coordinates is determined.
- Floor plan features: the indoor features and their positions in the floor plan frame are added to the floor plan database, e.g. rooms, paths, turnings and gates, shown as the green dots in Fig. 3. These floor plan features can be extracted by applying imaging processing methods on the floor plan pictures. In this paper, these features are manually selected. Referring to the real scale of the floor plan, pixel locations of indoor features are transformed to their positions in the floor plan frame.

3.3 Correspondences of Camera Image and Floor Plan Database

The system integrates the geo-reference information in the floor plan database with camera image to derive the camera position and orientation. When user takes an image of the indoor environment, the same scene is contained in the floor plan database with the floor plan features and their geo-reference information. By finding correct and reliable correspondences between the image feature and the floor plan feature, the

geo-tag of the floor plan feature can be used to derive accurate camera position and orientation. However, in an indoor image, many objects such as trash bin, bulletin board, lights are easily detected, which are not considered as reliable features. In this paper, only the features exist in both the camera image and the floor plan database are of interest, such as the corner of doorway and walls, namely, the indoor hallway features. The reason of choosing these indoor hallway features are based on two considerations: first, these indoor hallway features will remain static all the times but the arrangement of other objects like trash bins and furniture can be frequently moved or changed. Therefore indoor hallway features are much more reliable which reflect the indoor structure and arrangement; second, unlike the furniture plan or other detailed plan, the floor plan only depicts the abstract interior view while ignoring detailed objects and appliances, therefore only indoor hallway features can be found with correspondences in the floor plan. Figure 4 shows a picture of a hallway which is a very common indoor scene. A few indoor objects are noticeable such as door, wall, furniture and lights. The red dots are indoor hallway features and the green lines connect pairs of image-to-floor plan feature correspondences.

3.4 System Framework

Definition of frames

All coordinates are expressed in three types of frames, the floor plan frame, the camera frame and the image frame, which are defined as below:

- Floor plan frame: a three-dimensional frame with a local origin selected on the floor plan, x-axis pointing along the hallway, z-axis pointing up and y-axis orthogonal with both; the generation of the geo-reference information of the floor plan frame has been introduced in previous section;
- Camera frame: a three-dimensional frame with origin at the camera perspective center, x-axis pointing right, y-axis pointing up, z-axis orthogonal to the imaging plane;
- Image frame: a two-dimensional frame of the camera imaging plane. It departs from the camera perspective center with a distance of the focal length.

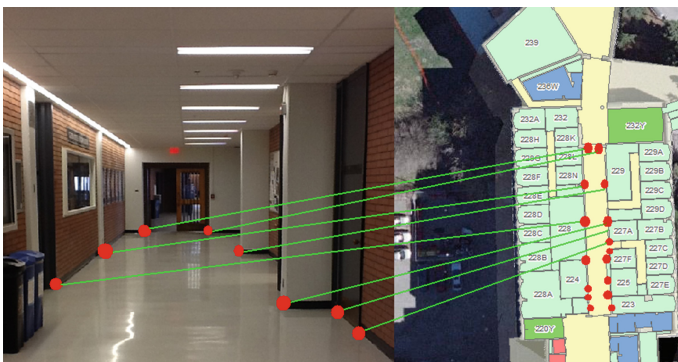


Fig. 4. Indoor hallway features in camera image and floor plan

System Components

The system consists of six major components and their functions are described in the following:

- Initial position and accuracy: The first component is to collect initial indoor position. Since GPS signal is totally unavailable in most deep indoor environments, the initial position relies on the Wi-Fi network and its accuracy is typically at tens of meters;
- Floor plan database: The second component is to download the floor plan geo-reference information from the database, and this is completed with the geo-fencing function. When a user sends request message to server including the initial position and its accuracy, the areas of interest are determined. A feedback message is sent to the user device which includes the floor plan pictures and geo-reference data of the indoor hallway features in the areas of interest;
- Photo of indoor scenario: The third component requires user to take a picture of the indoor scenario containing as many indoor hallway features as possible (usually 15 features are sufficient to derive reliable camera position and orientation) and touch on screen to specify indoor hallway features;
- Image feature detection: The fourth component is to search the user-touched areas to detect image features. Usually there are more than one feature detected but only the most intensive feature is selected.
- Robust feature matching: The fifth component is to identify the image-to-floor plan correspondences between the floor plan features and the image features.
- Navigation algorithm: The sixth component is to derive camera position and orientation. It includes two steps: first it takes the image-to-floor plan correspondences as inputs to derive the features' three-dimensional positions in the camera frame; then it takes the image features' 3D positions in the camera frame and the floor plan feature 3D positions in the floor plan frame as the inputs to derive the camera position and orientation in the floor plan frame.

4 Test Results and Analysis

4.1 Development of iOS App

An iOS App has been developed to implement the Wi-Fi/camera image/floor plan integrated system on the iPhone and iPad platforms. The software structure and the objective-C frameworks being used in software development are illustrated in Fig. 5. The initial indoor location is collected by using the CoreLocation framework, which outputs user's current latitude, longitude, and accuracy in unit of meter. The CoreLocation framework output is by default the combined results from GPS, cellular network and Wi-Fi fingerprinting; downloading floor plan data requires the communication with server through Wi-Fi connection, and the remote server is simulated with the Application Programming Interface (API) of a cloud storage Dropbox. The floor plan database is stored in Dropbox, and all necessary communication methods like sending request to server and receiving feedback to user are supported by API functions; once the user takes a picture of the indoor scenario, the feature detection

implements the image processing library OpenCV, which provides the FAST corner detection function; the navigation algorithm and RANSAC matching involve lots of matrix manipulations and a linear algebra library LAPACK is employed.

4.2 Test Description

To evaluate the performance of the integrated indoor positioning system, online interactive maps and manually collected geo-reference floor plan database are used. In order to verify the accuracy of the geo-reference information, the distances of the feature positions in the floor plan frame are compared with the length measurements collected in the Engineering Building at The University of Calgary. The accuracy of the floor plan feature positions is at decimeter level. This database currently only covers the first level of Engineering Building A, B, C, D and E Block. An iPad without 3G cellular module was adopted for indoor tests in the Engineering Complex, where GPS signal is totally unavailable. Therefore, the initial positions during the indoor tests were only Wi-Fi network based position solutions whose accuracy is on average at tens of meters.

The structure of the indoor environments can be extremely irregular and complex, and it is necessary to demonstrate the integrated system can work in general scenarios. In order to verify the performance in various indoor environments, the test areas are categorized into the following three scenarios:

- Standard Scenario such as D, C and B Blocks which have parallel hallways with recognizable indoor features;

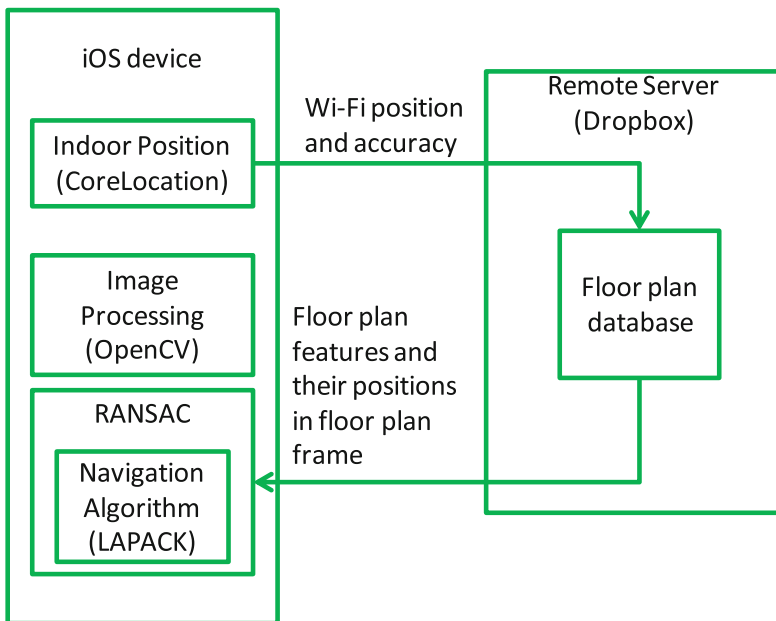


Fig. 5. iOS app and software structure

- Irregular Scenario such as E Block which has irregular hallways;
- Open Scenario such as A Block which does not have hallways.

Ten landmarks are selected in each test area, with a total of 50 landmarks. The App was run 10 times at each landmark, with a total of 500 indoor tests conducted. The test results are used to assess the positioning accuracy using the system.

4.3 Positioning Accuracy Improvement

In order to analyze the repeatability and positioning accuracy, both the Wi-Fi positions and the results derived by the integrated system are compared with the landmark reference positions. The 50 indoor landmarks are marked with color dots on the floor plan picture as shown in Figs. 6 and 7. From the 10 times of repeatability tests at each landmark, the mean and STD position error of both the integrated system and Wi-Fi positions are calculated. The radius of the dots indicates the mean position errors. The color scales indicate the the STD position errors, from cold color representing small STD error to warm color representing large STD error.

The STD errors are first compared in the following to assess the repeatability of the positioning solutions.

- STD errors of Wi-Fi position solutions: The overall STD errors vary from 1.28 m to 8.17 m, which is very typical Wi-Fi positioning performance. There are significant performance degradations in some areas. For example, the dots in END, ENE and ENA blocks are with relatively warm colors but the most dots in ENB and ENC blocks are blue. The reasons causing the inconsistent STD position errors of Wi-Fi

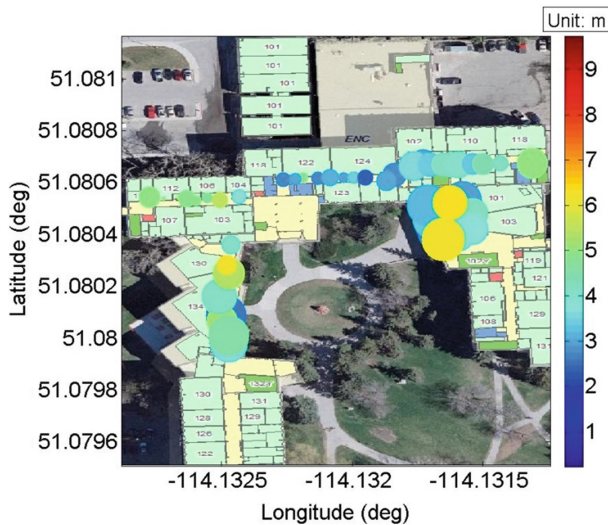


Fig. 6. Mean and STD errors of Wi-Fi positions

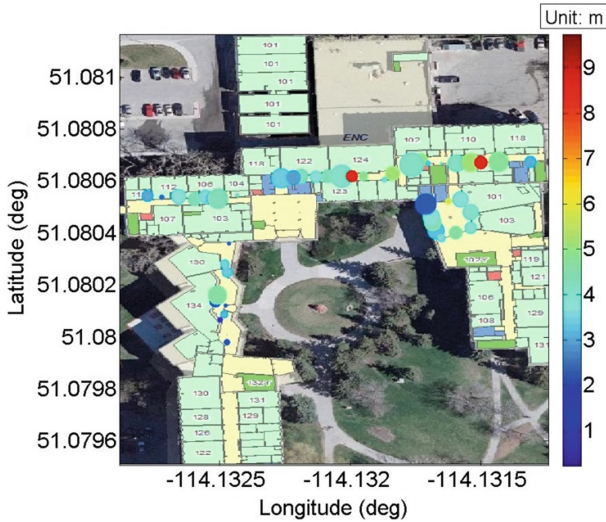


Fig. 7. Mean and STD errors of integrated solutions

positions are attributed the quality of the Wi-Fi AP database and the received signal strengths of Wi-Fi APs.

- STD errors of integrated position solutions: Two significant improvements are noticed compared to Wi-Fi solutions: first, the overall color of dots in Fig. 7 is cooler than Fig. 6. Especially in the areas of ENA, END and ENE where the Wi-Fi position repeatability is poor, the STD errors of the integrated position solutions are significantly reduced; second, the consistency of the colors in Fig. 7 is much better than Fig. 6.

The positioning errors with respect to the reference landmark positions are analyzed in the following.

- Mean errors of Wi-Fi position solutions: In Fig. 6, the dots in ENC block are on average smaller than the dots in other blocks. It means the mean position errors in ENC block are smaller than other blocks. A reasonable explanation to this phenomenon is attributed to uneven density of Apple’s Wi-Fi AP database. The database probably has stored highly dense APs in ENC area but relatively sparse APs in other blocks.
- Mean errors of integrated position solutions: Comparing with Figs. 6 and 7 has shown great reduction in mean position errors. On the one hand, in ENC block where the Wi-Fi positioning accuracy is good, the dot sizes in Fig. 7 are as small as those in Fig. 6. On the other hand, in the areas of ENB, END, ENE and ENA where the mean errors of Wi-Fi positioning are large, the dot sizes are much smaller for the integrated position solutions. The integrated system works well not only in the typical indoor scenarios with parallel hallways such as ENB, ENC and END blocks, but also perform well in irregular scenarios like ENE block.

Table 1. Position RMS errors in different test areas

RMS (m)	END	ENC	ENB	ENE	ENA
Wi-Fi positions	6.74	5.60	11.47	19.02	24.60
Integrated positions	4.56	6.63	7.20	3.24	5.99

Table 1 summarizes the position RMS error in each test area. It further demonstrates three contributions of the integrated system: first, the integrated system can bring position accuracy improvement to Wi-Fi positions; second, in the area where Wi-Fi performance is good, comparable accuracy is achieved by the integrated system; third, unlike the inconsistent performance of Wi-Fi positioning, the accuracy of the integrated system have shown great consistency in different blocks.

5 Conclusions

An integrated indoor positioning system based on integration of Wi-Fi, camera image and floor plan database has been described. Camera image is integrated with a ubiquitous database of building floor plan to derive 3D camera position and orientation with Wi-Fi position as the initial solution. An iOS App has been developed and tested with iPad in different indoor environments. The performance of the integrated system has been assessed by comparing to the Wi-Fi based positioning solutions. The test results have indicated significant improvement in terms of positioning accuracy and reliability in indoor environments using the integrated system.

Acknowledgement. The research is supported by Canada NSERC Discovery Grant.

References

1. Eunjoon, C., Seth, A.M., Jure, L.: Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, USA, pp. 1082–1090 (2011)
2. Melania, S.: Gait analysis for pedestrian navigation using MEMS handheld devices. M.Sc. Thesis, Geomatics Engineering, University of Calgary, UCGE Report 20359 (2012)
3. Yohan, C., Hojung, C.: LifeMap: a smartphone-based context provider for location-based services. *IEEE Pervasive Comput.* **10**(2), 58–67 (2011)
4. Hide, C., Botterill, T., Andreotti, M.: Vision-aided IMU for handheld pedestrian navigation. In: Proceedings of ION GNSS 2010, Portland, Oregon, USA, pp. 534–541 (2010)
5. Laura, R.: Visual gyroscope and odometer for pedestrian indoor navigation with a smartphone. In: Proceedings of ION GNSS 2012, Nashville, Tennessee, USA (2012)
6. Yuan, Z., Li, X., Wang, J., Yuan, Q., Xu, D., Diao, J.: Methods of 3D map storage based on geo-referenced image database. *Trans. Nonferrous Met. Soc. China* **21**(3), 654–659 (2011)
7. Huang, B., Gao, Y.: Indoor navigation with iPhone/iPad: floor plan-based monocular vision navigation. In: Proceedings of ION GNSS 2012, Portland, Oregon, USA (2012)

A Hybrid Scale-Out Cloud-Based Data Service for Worldwide Sensors

Tania Khalafbeigi¹(✉), Chih-Yuan Huang¹, Steve Liang¹,
and Mea Wang²

¹ Geomatics Engineering, University of Calgary, Calgary, Canada
{tkhalafb, huangcy, steve.liang}@ucalgary.ca

² Computer Science, University of Calgary, Calgary, Canada
meawang@ucalgary.ca

Abstract. We are living in a sensor-rich world. However, managing, accessing and analyzing the collective worldwide sensors' spatio-temporal observations in a coherent manner is very challenging. That is because the large number of sensors are distributed all over the world and each sensor provides large volume of continuous observations over the time. Our objective in this paper is to construct a scalable data service for gathering and accessing the worldwide sensors' collective observations. Our proposed solution has a hybrid architecture consisting of local services and a Cloud storage. In our solution, we combine a cloud-based scale out geospatial data stream architecture with the LOST-tree indexing structure. Our initial experiment shows that such hybrid structure is scalable and efficient for sensor data write, local search and global historical search.

1 Introduction

Cisco estimated that there will be more than 50 billions Internet-connected sensors by 2020 [6]. With the growth of the number of sensors all over the world, managing the information from the worldwide sensors will become an issue because of the very large volume of data they create over time. For example, the volume of sensor data is predicted to cross the volume of social media data before 2015 [5].

In this paper, we design a sensor data service to handle the following three functionalities. First the system needs to be able to accommodate the collective sensor observations from the worldwide sensors. In order to make the observations useful, each observation must have a temporal attribute (time) and a spatial attribute (space). We named this operation as *sensor check-in*. The second functionality of our proposed system is the ability to answer queries about the most recent sensor observations around a specific location. It is a common operation in location-based systems and often called nearby search [2, 7, 8]. We called this operation *local live query*. The third functionality of our proposed data service is the ability to answer queries about sensor observations in a specific large area during a specific time period. We call this operation *global historical query*.

Our proposed solution consists of two major parts: a local data service and a Cloud storage. The responsibility of the local service is to gather the observations from

sensors, and index them in a spatiotemporal indexing data structure. To prevent the unlimited growth of the local service's storage, the observations stay in the local service for a limited amount of time and then the local service publishes them to a Cloud storage for global accesses (hence the scale out design).

Our proposed hybrid architecture has the following advantages. First of all, our hybrid architecture strategically distributes the computational and storage loads to different components. Our Cloud storage provides an easily manageable and scalable central storage for worldwide sensor observations. The distributed local services remove the performance bottleneck of the continuous check-ins submitted by the worldwide sensors. Secondly, as the system grows, additional local services can be easily added into the system according to the spatial density of the sensors. At the meantime, the complexity of the system architecture does not increase because all sensor observations are still stored in the central Cloud storage. In addition, as the load is shared between the local services and Cloud storage, the proposed architecture doesn't require server-class machine. Instead, a PC-class machine is sufficient and as a result, this hybrid architecture is cost-effective and economically scalable. Another key advantage of our hybrid architecture is that we can use a simple key-value Cloud storage (e.g., the Amazon S3) instead of a compute Cloud (e.g., running a relational database in the Cloud) that has the following advantages. First, it has a very short response time as it does not perform complex queries. Second, It is also very scalable because the hash-table-like structure allowing it to scale. Third, simple key-value Cloud storage is very cost-effective comparing to a compute Cloud.

2 Related Work

Wang *et al.* [4] proposed a method for retrieving and indexing spatial data in the Cloud computing environment. Their main contribution in spatial data services is using Well-Known Text (WKT) and Well-Known Binary (WKB) for handling spatial data. However, in our solution we want to store and retrieve the spatial data itself, because transforming spatial data to WKT and WKB and vice versa add computational overhead to the data service. Li *et al.* [3] proposed an architecture for processing intensive floating car data. They used a temporal indexing method and a spatial indexing method two parts of their system. However, their system is different from our domain in the case that they spatial index is almost static and doesn't change and their structure is specifically designed for analytics. In our system the locations of sensors in our system are dynamic and also we focus on sensor data service common operations rather than analytics. Lee *et al.* [2] proposed a Cloud-based geolocation data service for mobile applications named Geopot. Our purposed solution is extending the Geopot architecture by adding the global historical query operation. Moreover, Geopot's focus was mainly on spatial data. However, we want our system to support spatio-temporal queries by using a spatio-temporal indexing structure. Huang *et al.* [1] proposed a spatio-temporal structure for efficient sensor data loading in a sensor web browser named Loading Spatio-Temporal tree (LOST-tree). LOST-tree spatio-temporal indexing structure can be useful for our purpose.

3 A Hybrid Scale-Out Cloud-Based Data Service for Worldwide Sensors

Our proposed hybrid architecture contains a number of local services and a Cloud storage. Local services handle users' requests while the Cloud storage contains the whole spatio-temporal data derived from the worldwide sensors over time. A R-tree index is created over the data storage to facilitate retrieving observations and answering the queries. In addition, the local data storages periodically publish the historical data to the Cloud storage. An additional attribute, that summarizes the to-be-published data, will also be submitted to the Cloud storage along with the data. Our proposed service provides three basic functionalities: check-in, local live query, and global historical query as defined in Sect. 1.

3.1 The Proposed Approach that Distributes the Local Services

Servers for the local services (i.e., local servers) should spread all over the world to manage observations from sensors in the different geographical areas. We assign a unique identifier to each of these local servers as references in the central Cloud storage. We need an identifier generation strategy so that the identifiers can represent the geographical area covered by the local services. There are different strategies to achieve the goal, such as the space-driven spatial indexing methods and the data driven indexing methods [8]. In this paper we decide to use the space-driven indexing methods because they are simple and do not require the system to rebuild local service topology when the system grows. We use quad-tree and z-ordering [8] on the world map to distribute our local servers. Quad-tree sub-divides the geographical area to four equal-size regions recursively. Using z-ordering traversing over quad-tree, each quad-cell is assigned a unique quad-key. To distribute our local servers, we use a pre-defined quad-tree structure configured with a specific depth. Then we put a local server in each quad-cell. Local servers are responsible for all queries from their corresponding quad-cell region. The quad-key of each cell is used as an identifier for a local server. This quad-key not only gives each local server a unique and informative identifier, but also the id represents a geographical area.

3.2 Local Service and Cloud Storage Architecture

Local services are responsible for interacting with the users, *i.e.*, receiving incoming requests and replying responses. As shown in Fig. 1a, each local server has its own key-value based data storage. When a sensor wants to check in its observation, it sends a check-in request, containing a sensor observation with its spatial (location), to the corresponding local server. In the local server, the observation is added to the data storage and its R-tree index based on the location. In order to save bandwidth and unnecessary processing overhead, we assume that the time of check-in is the same as observation time, sending sensor observation with only its spatial attribute is sufficient.

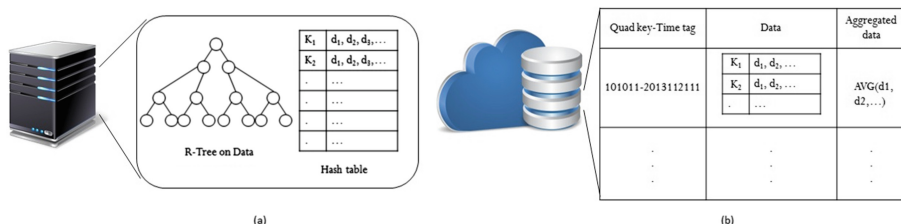


Fig. 1. (a) Local service architecture, (b) Cloud storage architecture

After a specified amount of time period, e.g., one hour or one day, the local service uploads its data to the Cloud storage. A time tag will be created based on the time period such as YYYYMMDD for a day or YYYYMMDDHH for an hour. The combination of the time tag and the quad-key identifier of the local server creates a unique key for uploading the data to the Cloud storage. The aggregate attribute of interest also needs to be calculated by local server and upload with the data to the Cloud storage. After uploading the data to the Cloud storage, the data can be deleted from the local server data storage to protect the local server from out of memory issue.

For the local live query, a user sends a request to the local service containing the location and the radius of interest. Based on the user location, the local server searches its R-tree and retrieves the result data from its data storage. Using the R-tree in our architecture improves the performance of local live query as demonstrated in Sect. 4. For the global historical query, the user sends a request to the local service containing time period and area of interest. Local server calculates the time tags and the quad-keys overlapping with the time period and area of interest respectively and fetches the data from the Cloud storage. Depend on users' desired granularity of the result, the Cloud storage returns the raw data or the aggregate attribute of the data. The result contains a series of region-time/observations that can be displayed on the map.

Our proposed architecture for the Cloud storage, as shown in Fig. 1b, is a simple key-value table contains the collective raw data submitted from all the local servers. In addition to the local data from local servers, aggregate attributes are also stored in the Cloud storage key-value table that facilitates answering the global historical queries. We believe that by combining our hybrid structure with the indexing structure, our service is scalable and efficient for sensor check-ins, local live queries and global historical queries.

4 Evaluation Results

For evaluating our proposed sensor data service, we simulated the local server and connected it to the cost-effective Amazon Simple Storage Service (S3)¹. We used random, linearly distributed, simulated data for evaluating our implementation. Our implementation consists of a local server that is implemented with Java programming

¹ <http://aws.amazon.com/s3>

language. We used Jersey² library to provide HTTP access to our local service and Tomcat6 hosts our service. The service resides on an Intel Core i5 PC with 1 TB disc and 6 GB memory running Windows 7 Professional. We also used Redis³, an open source advanced key-value store, for local data storage. In addition, we used the deegree R-tree library⁴ for our R-tree implementation in local service. Using our local service is as simple as sending HTTP requests. Check-in operation is done with HTTP POST requests and local live queries and global historical queries are done with HTTP GET request. In our implementation data of the local service is uploaded to the Amazon S3 using Amazon AWS Java SDK every hour.

4.1 Experiments

Check-in Response Time. Our first experiment for evaluating scalability is check-in response time (Fig. 2a). X axis shows index of the check-in request, means the i^{th} check-in request comes to the service, and the Y axis shows the time in nanoseconds. The experiment is evaluated using ten ms check-in requests. This experiment shows that the overall response time of check-in is less than 5 ms and is not changed with more check-ins. However, when a node in the R-Tree index splits, the check-in response time is higher and this is the reason that some of the points in the diagram have different response time with the overall response time.

Local Live Query Response Time. The other experiment for evaluating scalability is testing local live queries response time with the change of the number of stored checked-in data (Fig. 2b) We did this experiment with ten million data records stored in the local service. We see that the response time for the local live queries change linearly (with a very low coefficient) with the number of checked-in data. As we see the response time is reasonable for ten million data and is less than 4 ms.

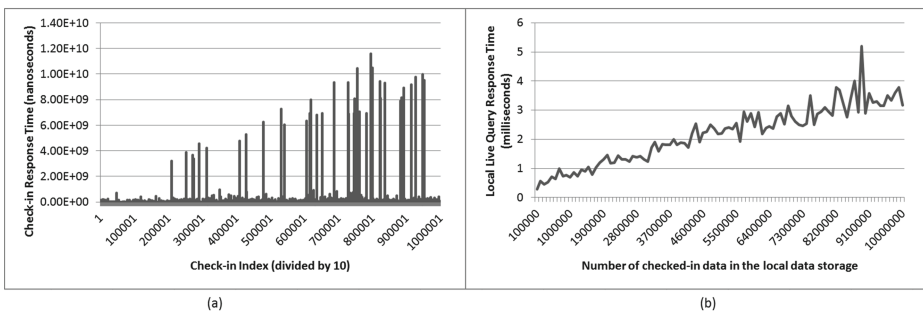


Fig. 2. (a) Response time for check-ins based on the index of check-ins, (b) Response time for local live queries based on number of checked-in data

² <https://jersey.java.net/>

³ <http://redis.io>

⁴ <http://download.deegree.org/deegree2.5/api/org/deegree/io/rtree/RTree.html>

Global Historical Query Response Time. The global historical query performance depends on the Amazon S3 reading time and also the size of the desired bounding box and length of the desired time period in the query. We tested Amazon S3 reading time. The average reading time from Amazon S3 with the 1Gbps network bandwidth of University of Calgary is 113.3 ms. The aggregate attribute that we store in the Cloud storage helps improving the performance of global historical query significantly. Because instead of fetching the whole data for the desired area and time period, only the aggregate attributes need to be fetched from Amazon S3 that save the bandwidth and improve the performance.

Local Server Distribution. The number of the local servers depends on a specified quad-tree depth. There are trade-offs for choosing this depth. First of all this level affects the performance of the global historical query. For each quad-cell and for each time period we have a data in the Cloud storage. And that means the deeper the depth of quad-tree is, the worse the performance of the global historical query will be. However, a deeper level of quad-tree also means a better estimation of sensor observations, since the aggregate attribute summarizes the data for a finer grained area and time period. A similar situation occurs when choosing the time period (*i.e.*, time tag) for submitting data to the Cloud storage.

5 Conclusion and Future Works

In this paper we proposed a hybrid scalable architecture for designing the worldwide sensor data service. The experimental results showed that our proposed data service is scalable and also performs well. Moreover, we believe that our proposed data service is cost-effective due to use Cloud storage and PC-class machines for local servers. Our future works include a number of improvements for our data service design. Using static quad-tree for distributing local services may cause unbalanced workload between local services. One of the solutions to address this problem is using dynamic quad-tree instead of the static one. In addition, our current data service is designed for one sensor type. The support of multiple sensor types can be simply added by either adding a sensor registry or encode the sensor types in the local server's unique identifiers.

References

1. Huang, C.-Y., Liang, S.H.L.: Lost-tree: a spatio-temporal structure for efficient sensor data loading in a sensor web browser. *Int. J. Geogr. Inf. Sci.* **27**(6), 1190–1209 (2013)
2. Lee, D.W., Liang, S.H.L.: Geopot: a cloud-based geolocation data service for mobile applications. *Int. J. Geogr. Inf. Sci.* **25**(8), 1283–1301 (2011)
3. Li, Q., Zhang, T., Yu, Y.: Using cloud computing to process intensive floating car data for urban traffic surveillance. *Int. J. Geogr. Inf. Sci.* **25**(8), 1303–1322 (2011)
4. Wang, Y., Wang, S., Zhou, D.: Retrieving and indexing spatial data in the cloud computing environment. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) *Cloud Computing*. LNCS, vol. 5931, pp. 322–331. Springer, Heidelberg (2009)

5. Liang, S.H.L., Huang, C.Y.: Geospatial Cyberinfrastructure for Addressing the Big Data Challenges on the Worldwide Sensor Web. *Big Data: Techniques and Technologies in Geoinformatics*. CRC Press, Boca Raton (2014)
6. Evans, D.: The internet of things: How the next evolution of the internet is changing everything. CISCO white paper (2011)
7. Zheng, Y.: Tutorial on location-based social networks. In: *WWW 2012*, (2012)
8. Rigaux, P., Scholl, M., Voisard, A.: *Spatial Databases: with Application to GIS*. Morgan Kaufmann, Burlington (2001)

**DASFAA Workshop on Uncertain
and Crowdsourced Data (UnCrowd)**

Uncertainty in Crowd Data Sourcing Under Structural Constraints

Antoine Amarilli¹(✉), Yael Amsterdamer², and Tova Milo²

¹ Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI, Paris, France
`antoine.amarilli@telecom-paristech.fr`

² Tel Aviv University, Tel Aviv, Israel

Abstract. Applications extracting data from crowdsourcing platforms must deal with the *uncertainty* of crowd answers in two different ways: first, by deriving estimates of the correct value from the answers; second, by choosing crowd questions whose answers are expected to minimize this uncertainty relative to the overall data collection goal. Such problems are already challenging when we assume that questions are unrelated and answers are independent, but they are even more complicated when we assume that the unknown values follow hard *structural constraints* (such as monotonicity).

In this vision paper, we examine how to formally address this issue with an approach inspired by [2]. We describe a generalized setting where we model constraints as linear inequalities, and use them to guide the choice of crowd questions and the processing of answers. We present the main challenges arising in this setting, and propose directions to solve them.

1 Introduction

Crowd data sourcing leverages human knowledge to obtain information which does not exist in conventional databases. This may be done by posing targeted questions to crowd users, through conventional crowdsourcing platforms such as Amazon Mechanical Turk [6]. Contrary to many works that use the crowd as a *means* to perform different tasks, here the crowd serves as a *source* of information.

Many challenges arise when using the crowd as a data source. First, human answers have a high latency and are usually provided against some (monetary) compensation, so we must minimize the number of posed questions. Second, answers collected from the crowd may be erroneous and noisy, so we must control and improve answer quality, e.g., pose the same question to multiple workers.

A vast body of research has tackled these issues for various data procurement tasks (e.g., [1, 2, 7, 8, 11, 12]). For example, [7] studied the number of answers that must be obtained to reach sufficient confidence in the final answer of a given Boolean question, and mentions the problem of deciding, when there are several questions to answer, which is the *next best question* to ask the crowd. In different situations [2, 12], this selection of questions is performed by comparing

the expected contribution of the answers to some data acquisition goal. However, in such situations, the answers to the various questions are independent, so that we can choose the next best question by looking at each question in isolation.

In this paper, we study the problem of collecting numerical values from the crowd under *hard a-priori constraints* on the final answers, caused by inherent data dependencies.

For instance, suppose that we have devised a lossy compression algorithm for e.g. music files, and that we wish to estimate the average quality rating of different compression ratios in a user population. We can ask a few random crowd workers to evaluate the quality q_1, \dots, q_n for each of n compression ratios of *increasing* lossiness. The quality ratings of any given person are not independent: we can assume that every person will consider q_1 (the quality of the least lossy compression) to be at least as high as q_2 , and so on. Consequently, the average q_1 in the entire population is higher than q_2 , and so on. However, the quality q_1 for *some* people might be lower than q_2 for others; hence, by asking random workers we may obtain an estimation of the quality ratings that is not perfectly monotone. This use case will be our running example throughout the paper.

As another example, consider the estimation of the price that people are willing to pay for varying combined deals. In fields such as auction study in game theory [9], it is customary to assume that the price function for each user is monotone, i.e., adding products cannot decrease the deal price. For instance, we know that in the entire population, the average value of a flight and hotel cannot be lower than that of the flight alone. But again, if we sample different users for each deal, we may obtain a non-monotone estimation for the average price.

A similar problem occurs in [2], where the crowd is used to estimate the frequency of patterns in user habits. While these frequencies are dependent for patterns with overlapping activities (e.g., if someone never swims, they also never swim and dive), such dependencies are not accounted for in [2]. In general, existing work on crowd data sourcing has mostly ignored the problem of uncertainty when dealing with dependent questions [1, 8]. There are works that deal in a non-trivial way with the interaction between uncertainty and dependency [4, 10], but they assume that the individual outcomes observed are Boolean and not numeric like in the present paper.

We consider here two important problems that arise in the context of dependent crowd questions. First, can we *improve the variable estimation* by taking dependencies into account? For instance, in our running example, if we estimate that the quality q_1 is lower than q_2 (which contradicts our monotonicity assumption), we may attempt to correct our estimation by increasing q_1 and/or decreasing q_2 ; or, to begin with, we can only consider estimations that comply with our monotonicity requirement. What is the right way to enforce this monotonicity, and how does it increase the quality of our estimation?

Second, we use the dependencies to *reduce the number of questions* posed to the crowd. For instance, if we estimate that the average quality rating q_1 and q_4 of the compression ratios 1 and 4 are both 6 out of 10, we do not need to ask

people about q_2 and q_3 . Or, as another example, if we wish to find the lossiest compression with rating at least 6 out of 10, and we estimate that q_5 and q_{10} are 8 and 3 respectively, we can interpolate q_6, \dots, q_9 (using monotonicity) and estimate that the rating with value closest to 6 is most likely q_7 .

Paper structure. We first give a formal definition of the considered problem in Sect. 2. We next present in Sect. 3 a general scheme to solve the problem in the absence of dependencies, and turn in Sect. 4 to how dependencies should be handled. For numerous dependent variables, interpolating samples is crucial: we discuss this in Sect. 5. Last, we conclude in Sect. 6.

2 Problem Statement

We wish to learn n numerical values $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)$ from the crowd. We model the *distribution of crowd answers* to questions about these values using n random variables X_1, \dots, X_n . We assume that the *mean* of X_i is μ_i for every i .¹

We further assume that $\boldsymbol{\mu}$ satisfies a certain known set of *linear inequalities*, represented as a matrix E of reals such that $E \cdot \boldsymbol{\mu} \leq (0)$, where (0) is the zero vector and \cdot denotes the product of matrix E and vector $\boldsymbol{\mu}$. We assume that the inequalities E are feasible, namely, that there is some vector \mathbf{e} satisfying E .

Example 1. In our running example, the random variables Q_1, \dots, Q_n , with unknown means q_1, \dots, q_n , denote the ratings obtained for the compression ratios. The inequalities represent a decreasing order: $q_2 - q_1 \leq 0$, $q_3 - q_2 \leq 0$, etc.

We consider a known *loss function* $L_{\boldsymbol{\mu}}$ which associates to a prediction \mathbf{v} for the unknown values $\boldsymbol{\mu}$ some nonnegative value $L_{\boldsymbol{\mu}}(\mathbf{v})$. We assume that $L_{\boldsymbol{\mu}}$ can be written as the sum of nonnegative functions $L_{\mu_i}^i$, that is, the error function for all values is the sum of the errors of individual values. We require that for all i , $L_{\mu_i}^i(\mu_i) = 0$ and $L_{\mu_i}^i(x) \leq L_{\mu_i}^i(y)$ for all $\mu_i \leq x \leq y$ and $y \leq x \leq \mu_i$. (In other words, the loss is 0 for the correct value, and increases with the absolute error.)

Example 2. The loss function depends on the target application. For compression ratios, if our task is to find which is the lossiest compression with rating at least 6, a reasonable loss function for all variables is the *threshold* loss $L_{\boldsymbol{\mu}, \tau}$ with $\tau = 6$. The value $L_{\boldsymbol{\mu}, \tau}(x)$ is defined to be 1 if the x and μ are miscategorized with respect to threshold τ (formally, $\mu < \tau < x$ or $x < \tau < \mu$) and 0 otherwise. The overall loss function is the sum of the $L_{q_i, \tau}$ which counts the number of ratios that are miscategorized with respect to the threshold $\tau = 6$.

For any i , we can obtain a sample of variable X_i (we say that we *sample* X_i or *draw* X_i) by asking the corresponding question to a random crowd worker; we

¹ This assumption holds when we are interested in the *average* crowd answer, e.g., the average rating for a compression quality; and in the many cases where the errors of worker answers tend to cancel out so that the average is close to the truth [2].

assume that all draws are independent both between variables and between two draws of the same variable. Our goal is to choose draws carefully and, based on the obtained samples, try to provide a prediction \mathbf{v} which minimizes $L_\mu(\mathbf{v})$: we phrase this in a *fixed-budget* formulation, namely minimize $L_\mu(\mathbf{v})$ in expectation after a fixed number of samples.

Example 3. In the running example, sampling the variable Q_i is achieved by providing a random crowd user with a sound sample compressed with ratio i and asking for a rating for this sample. The overall objective is to choose the right ratios for which to request more ratings, in order to minimize the number of average quality ratings that are miscategorized with respect to $\tau = 6$.

We next review the problem of minimizing the loss by choosing the “right” questions. We first study an approach for a simplified setting where there are no order constraints on the estimated values, before we consider the general case.

3 Without Order Constraints

Let us present a general scheme inspired by [2] for the case with no order constraints, before we extend it to order constraints in the next section.

With no constraints, as the variables are independent and the loss is the sum of the individual losses of variables, our goal is to find which one of the variables is such that one more sample for it would yield the largest loss reduction. Hence, we first focus on an individual variable X_i to describe how we predict its mean value v_i from the samples S_i observed for this variable, and how we estimate the loss reduction that we may achieve by taking one more sample.

Estimating the parameter. Our approach for a variable X given a set S of samples of this variable is to fit a model for X from the family of normal distributions, as they are a simple and general way to represent real-life data. Denote by $\Theta = \mathbb{R} \times \mathbb{R}_+$ the *parameter space*, such that every $\theta \in \Theta$, with $\theta = (\mu, \sigma^2)$, represents the normal distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2 . Denote by Pr_θ the probability density function of this distribution.

As the samples S of X are assumed to be independent, we can define the probability of S according to $\mathcal{N}(\theta)$ as the product of $\text{Pr}_\theta(s_i)$ for all $s_i \in S$. The *likelihood function* $\mathcal{L}_S : \Theta \rightarrow [0, 1]$ is then simply defined as $\mathcal{L}_S(\theta) = \text{Pr}_\theta(S)$: it describes, as a function of θ , the probability² of the sample under θ .

Our way to fit a normal distribution to the random variable X is then the standard method of choosing the *maximum likelihood estimator* (MLE):

$$\hat{\theta} = \arg \max_{\theta \in \Theta} \mathcal{L}_S(\theta)$$

In the case of normal distributions, it is easily checked that we have $\hat{\theta} = (\hat{\mu}, \hat{\sigma}^2)$, where $\hat{\mu}$ and $\hat{\sigma}^2$ are the *sample mean* and *sample variance* defined by:

$$\hat{\mu} = \frac{1}{|S|} \sum_i s_i \qquad \hat{\sigma}^2 = \frac{1}{|S|} \sum_i (s_i - \hat{\mu})^2$$

² Note that likelihood cannot, however, be seen as a probability distribution on Θ .

Hence, we take $v = \hat{\mu}$ as our current guess of the mean of variable X .

Example 4. Assume that we ask 3 users to evaluate sound samples compressed with ratio 3, and obtain the grades 3, 5, and 7. This means that our sample mean and variance for variable Q_3 are respectively $\hat{\mu}_3 = 5$ and $\hat{\sigma}_3^2 = 8/3$.

Estimating the error. How to estimate the loss of our prediction $\hat{\mu}$? Because the true value is unknown, we estimate the loss by assuming that our current guess $\hat{\theta}$ is correct, and finding out what its expected error is. We do this by examining the range of samples that could have been obtained instead of S under the assumed distribution and computing the loss of the MLE obtained from them.

By the central limit theorem, the distribution of the mean of N samples of $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$ can be approximated by $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2/N)$. Hence, under the assumption that $\hat{\theta}$ is correct, we can define the average error obtained through the MLE method from $|S|$ samples, as follows:

$$E(\hat{\theta}, |S|) = \int_{x \in \mathbb{R}} \Pr_{(\hat{\mu}, \hat{\sigma}^2/|S|)}(x) L_{\hat{\mu}}(x) dx$$

This integral can be numerically approximated by sampling.

Example 5. The estimated error for Q_3 under the samples S_3 of the previous example is the probability that the sample mean, distributed according to $\mathcal{N}(5, (8/3) \cdot (1/3))$, is above threshold $\tau = 6$ (as the loss is then 1, and is 0 otherwise, relative to our estimate $\hat{\mu}_3 = 5$). Numerically we have $E(\hat{\theta}_3, |S_3|) = 0.144$.

Estimating the error decrease. Now that we can estimate the parameter of a distribution from the samples, and the expected error according to this parameter, we can easily devise an estimation of how this error may decrease when an additional sample is requested from variable X .

Let us assume that we obtain a new sample of X with value x , and call S' the $|S| + 1$ samples obtained by adding x to S . Call $\hat{\theta}'$ the MLE obtained by maximizing $\mathcal{L}_{S'}$, and define the *error decrease* as $D(S, x) = E(\hat{\theta}, |S|) - E(\hat{\theta}', |S| + 1)$. This gives us an estimation of how error decreases for one more sample with value x . Of course, we cannot know if we would indeed obtain value x , but we can compute its probability according to our current hypothesis for the underlying distribution, namely $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$. We therefore define the *expected error decrease*:

$$D(S) = \int_x \Pr_{\hat{\theta}}(x) D(S, x) dx$$

This is our estimate of the expected loss reduction when sampling this variable.

Example 6. If we obtain one additional sample of 5 for Q_3 (yielding S'_3), the estimated mean $\hat{\mu}_3 = 5$ is unchanged, but the estimated variance decreases to 2 so the estimated error under the new MLE $\hat{\theta}'_3$ becomes $E(\hat{\theta}'_3, |S'_3|) = 0.079$. We estimate the expected error decrease by averaging the decrease under possible additional samples drawn from our estimated distribution $\mathcal{N}(\hat{\theta}_3)$ for Q_3 .

Multiple variables. With the above method, we can compute the expected error decrease of each variable, and sample the one whose expected error decrease is highest. It is easy to see that this greedy approach is optimal in terms of reducing the expected error over any fixed number of requests, as samples for one variable do not change the estimated parameter or expected error of other variables.

4 With Order Constraints

Under order constraints, the problem is more challenging. Though the loss function remains a sum of loss functions over individual parameters, it is not possible anymore to manage variables separately, because information obtained for one variable gives us additional information about the other variables. Reconsidering our running example, under the objective of identifying the lossiest compression ratio with average quality at least τ , it makes little sense to consider the results of every variable independently, and we should examine the results globally to locate where the decreasing sequence of qualities intersects the threshold τ . The challenge is how to formalize such a global strategy, under general constraints.

To this end, we propose a greedy strategy inspired by that of the previous section, but integrating the order constraints and considering the variables globally rather than in isolation. Because additional samples on one variable give us information about other variables, such a greedy approach is no longer guaranteed to be optimal over multiple draws. Because of space constraints, we only sketch the principles of our initial approach; we plan to study this further and examine possible alternative approaches in future work.

We consider the parameter space $\Theta = (\mathbb{R} \times \mathbb{R}_+)^n$, covering all parameters of all random variables simultaneously, and we define the likelihood of $\theta \in \Theta$ (with $\theta_i = (\mu_i, \sigma_i^2)$) as a function of $\mathcal{S} = (S_1, \dots, S_n)$, the set of all samples for all variables, using the fact that all draws are still independent. We exclude parameters which violate order constraints by defining the likelihood as follows:

$$\mathcal{L}_{\mathcal{S}}(\theta) = \begin{cases} \prod_i \prod_{s \in S_i} \text{Pr}_{\theta_i}(s) & \text{if } E \cdot \theta \leq (0) \\ 0 & \text{otherwise} \end{cases}$$

The main problem is now to determine the maximum likelihood estimator for θ by maximizing this expression. We next propose a possible approach to the problem, and the challenges yet to be resolved.

Estimating the means. We propose to maximize the expression as a function of the means μ , while making the assumption that the variances are the sample variances $\widehat{\sigma}$ for every individual variable. Under this approximation, the maximization problem can be rewritten as maximizing a quadratic expression with a positive definite matrix under the inequalities E . Such a problem is tractable [5], so we can solve it and obtain a set of candidate means \mathbf{v} for the underlying distributions. Technical details are omitted for lack of space.

Example 7. Assume that we have obtained the same number of samples for Q_1 , Q_2 and Q_3 , that their sample variances are equal ($\widehat{\sigma}_1 = \widehat{\sigma}_2 = \widehat{\sigma}_3$), and that the

sample means are $\widehat{\mu}_1 = 9$, $\widehat{\mu}_2 = 7$, and $\widehat{\mu}_3 = 8$. Observe that we have $\widehat{\mu}_2 < \widehat{\mu}_3$ even though we know that $q_3 \leq q_2$. In this specific setting, our estimation of the means is the solution \mathbf{v} of a quadratic programming problem amounting to minimizing the sum of squares $\sum_i (v_i - \widehat{\mu}_i)^2$ subject to the inequalities: its solution is $v_1 = 9$, $v_2 = 7.5$, and $v_3 = 7.5$.

Estimating the variances. We have computed the MLE estimator for the means of the distributions subject to the inequality constraints, up to the approximation of substituting the individual sample variances instead of integrating them in the maximization problem. Since the estimations of the means and variances are inter-dependent, we may now need to reestimate the variances.

Example 8. Assume that we have samples $S_2 = \{0.1, 0.2\}$ for Q_2 , and numerous samples for Q_1 and Q_3 which convince us that $v_1 = 9$ and $v_3 = 8.5$ are very good estimates for q_1 and q_3 . We know that we must have $8.5 \leq v_2 \leq 9$ (we will probably choose $v_2 = 8.5$ given S_2), but then our estimation of the variance of Q_2 should be much higher than the sample variance $\widehat{\sigma}_2^2$ of S_2 in isolation.

We estimate the variance of each X_i under the computed means \mathbf{v} (and thus estimate the complete parameter θ) as the sample variance relative to the computed mean v_i of X_i (instead of relative to the sample mean). The solution thus obtained may not be optimal, as we have fixed and optimized the means and variances separately rather than simultaneously. Estimating how much this approach deviates from the true solution is a challenge for future work.

Estimating the error and error decrease. The overall method now follows Sect. 3 except that we follow the above³ to fit a family of distributions to the variables.

5 Interpolation

In some real-life scenarios, we may have a very large number of questions to ask the crowd; for instance, the number of possible compression ratios may be very high, almost continuous. In such cases, we may have many variables X_i with no samples at all: those variables thus do not appear in the optimization problem, so that we know nothing about them (except that they satisfy the order constraints). However, we could then perform *interpolation* to estimate more precisely a large proportion of the variables with a limited number of questions to the crowd.

In the general case where E is an arbitrary set of inequalities, it is hard to define how to interpolate a value for a variable with no samples. We leave this general question to future work, and only focus on the case where E expresses the total order $\mu_1 \geq \dots \geq \mu_n$. For simplicity, up to renumbering indices, we assume that we have a model for X_1 and X_n , namely (μ_1, σ_1^2) and (μ_n, σ_n^2) , and that we wish to derive a model for X_k , $1 \leq k \leq n$, for which we have no samples.

³ Note that this also changes the way of fitting distributions when computing the error decrease under possible additional samples.

Example 9. If we estimate $v_1 = 8$ and $v_5 = 4$, our best guess for q_3 in the absence of samples should be $v_3 = 6$. Likewise, our best guess for q_4 should be $v_4 = 5$.

Interpolating the mean. We interpolate the mean μ_k by a linear interpolation between μ_1 and μ_n according to the rank k , as presented in Example 9

Interpolating the variance. We want to interpolate σ_k^2 by combining both the variances of X_1 and X_n , and the uncertainty arising from the interpolation itself: the further away k is from 1 and n , the least certain we are about μ_k .

To do so, we consider that μ_k has been chosen by picking $n - 2$ random uniform values between μ_1 and μ_n (the means μ_2, \dots, μ_{n-1}), sorting them, and choosing the $(k - 1)$ -th value to be μ_k . Now, this means that μ_k is the $(k - 1)$ -th order statistic of $n - 2$ uniform and independent random variables in $[\mu_1, \mu_n]$, so that it follows a *beta distribution* [3] whose variance has a closed form.

Example 10. Pursuing Example 9, for $\mu_5 = 8$ and $\mu_9 = 4$, we estimate the variance on μ_7 to be $4/5$ for this outcome (that of the adequate beta distribution).

We can thus estimate a variance for X_k for fixed values μ_1 and μ_n : those values are unknown, but can be sampled according to our model for X_1 and X_n to yield an overall variance for X_k . We omit details for lack of space, and leave to future work the study of other possible interpolation methods for variance.

6 Conclusion and Perspectives

In this paper, we have studied the problem of learning numerical values from the crowd, leveraging ordering constraints on those values to mitigate the uncertainty on crowd answers. We have presented an abstract framework inspired by [2] ignoring the order constraints, and presented an approximate method to take those constraints into account, along with a way to interpolate values for yet unsampled variables. We have identified further challenges to be explored.

Our main direction for future work is to study more carefully the approximations and design choices that we made, noting that our overall generic approach could be adapted to other probability distribution families than normal distributions; and to implement our approach to evaluate its effectiveness. We plan to evaluate, over various datasets and objectives, the importance of accounting for order constraints and performing interpolation, and compare our approach to round-robin or random baselines, as well as ad-hoc strategies for specific scenarios such as total orders.

Acknowledgements. This work has been partially funded by the European Research Council under the FP7, ERC grant MoDaS, agreement 291071, and by the Israel Ministry of Science.

References

1. Amarilli, A., Amsterdamer, Y., Milo, T.: On the complexity of mining itemsets from the crowd using taxonomies. In: Proceedings of ICDT (to appear), Athens (2014)
2. Amsterdamer, Y., Grossman, Y., Milo, T., Senellart, P.: Crowd mining. In: Proceedings of SIGMOD, New York, USA, pp. 241–252 (2013)
3. David, H.A., Nagaraja, H.N.: Order Statistics, Chapter 2, p. 14. Wiley, New York (2013)
4. Karp, R.M., Kleinberg, R.: Noisy binary search and its applications. In: Proceedings of 18th ACM-SIAM Symposium on Discrete Algorithms (2007)
5. Kozlov, M.K., Tarasov, S.P., Khachiyan, L.G.: The polynomial solvability of convex quadratic programming. USSR Comp. Math. Math. Phys. **20**(5), 223–228 (1980)
6. Amazon Mechanical Turk. <https://www.mturk.com/>
7. Parameswaran, A., Garcia-Molina, H., Park, H., Polyzotis, N., Ramesh, A., Widom, J.: Crowdscreen: Algorithms for filtering data with humans. In: Proceedings of SIGMOD (2012)
8. Parameswaran, A., Sarma, A., Garcia-Molina, H., Polyzotis, N., Widom, J.: Human-assisted graph search: it’s okay to ask questions. Proc. VLDB **4**(5), 267–278 (2011)
9. Parkes, D.C., Ungar, L.H.: Iterative combinatorial auctions: theory and practice. In: Proceedings of AAAI/IAAI (2000)
10. Triantaphyllou, E.: Data Mining and Knowledge Discovery by Logic-Based Methods, Chapter 10. Springer, New York (2010)
11. Trushkowsky, B., Kraska, T., Franklin, M.J., Sarkar, P.: Crowdsourced enumeration queries. In: Proceedings of ICDE (2013)
12. Yang, X., Cheng, R., Mo, L., Kao, B., Cheung, D.: On incentive-based tagging. In: Proceedings of ICDE (2013)

Integration of Web Sources Under Uncertainty and Dependencies Using Probabilistic XML

M. Lamine Ba¹ (✉), Sebastien Montenez¹, Ruiming Tang²,
and Talel Abdesslem¹

¹ Institut Mines-Télécom, Télécom-ParisTech, LTCI, Paris, France
{mouhamadou.ba,sebastien.montenez,
talel.abdessalem}@telecom-paristech.fr

² National University of Singapore, Singapore, Singapore
tangruiming@nus.edu.sg

Abstract. We study in this vision paper the problem of integrating several web data sources under uncertainty and dependencies. We present a concrete application with web sources about objects in the maritime domain where uncertainties and dependencies are omnipresent. Uncertainties are mainly caused by imprecise information trackers and imperfect human knowledge. Dependencies come from the recurrent copying relationships occurring among the sources. We answer the issue of data integration in such a setting by reformulating it as the merge of several uncertain versions of the same global XML document. As an initial result, we put forward a probabilistic XML data integration model by getting some intuitions from the versioning model with uncertain data we proposed in [5]. We explain how this model can be used for materializing the integration outcome.

1 Introduction

Uncertain Data Integration. Data integration with uncertainty, in the form of a probabilistic mediated schema [3, 7, 8] or probabilistic reconciled databases [2, 17], was previously dealt in both relational and XML settings. Probabilistic mappings [7, 8], yielding a probabilistic mediated schema, specify the different possible ways of matching the attributes in multiple relational schemas with respect to their semantics. Query views in [2] through containment constraints are used to define the mappings between a set of uncertain sources and a probabilistic mediated database having a fixed schema. Probabilistic trees in [17] with possibility and probability nodes enable to synchronize several uncertain XML documents by enumerating the different alternatives in data values. Used mappings, query views and reconciliation methods do not care about possible dependencies between sources during the integration process, and thereby they may fail in modeling the set of possibilities and probabilities on the presence of dependencies. One reason is they only assume, in general, that sources describe information about the same real-world objects in an *independent* manner.

Studied Problem. We consider in this paper the problem of integrating several web data sources under *uncertainty* and *dependencies*. On the web, many systems collect and keep up-to-date as much as possible a vast amount of information covering various real-life areas. Uncertainty in web data sources is a well-known issue, on the one hand. On the other hand, the existence of sources that copy (or crawl) data from some others is also a reality. The latter observation translates, as shown in [9, 10, 13], dependencies among the sources in terms of (independent) providers and copiers. Uncertainty and dependencies are two issues particularly true in web sources for objects in the *maritime domain* as we will detail in the next.

A large amount of information related to objects in the maritime domain can be found and extracted from numerous web sources with different nature. Web platforms such as marinetraffic.com, grosstonnage.com and shippingexplorer.net maintain specifications of ships (or boats) and monitor in real-time their locations and routes. Wikipedia, the most popular and successful collaborative content-based web editing platform, contains basic information about some kinds of ships. It harnesses for this the power of the crowd, i.e., its contributors around the world. Last but not least, we have also social networks like Twitter and Flickr that inform their followers (who are interested) about the ship routes with user posts and tweets. Through the aforementioned source examples, we can easily see that there is enormous knowledge on the web which might be valuable for maritime actors, especially for monitoring the traffic. In order to take advantage of all this knowledge, users within such a context may probably want to be able to transparently query and visualize the integration of information from these sources. Unfortunately, uncertainties and dependencies are omnipresent when talking about web data sources in the maritime domain. Uncertainties are mainly caused by (a) imprecise trackers, also known as automatic identification systems (abbr. AIS), which send at constant time intervals ship data like their positions and; (b) unreliable users sharing information about ships in a collaborative or social manner. Dependencies correspond to the copying relationships between sources as we illustrate in Sect. 2. Setting up the outcome of the integration of these sources requires thus to deal with uncertainty and dependencies.

Consider as a running example three sources S_1 , S_2 and S_3 sharing a subset of objects in the maritime domain. Let us focus on values they give for the *draft* and the *actual port* for a particular ship named “Costa Serena”. The first source S_1 independently reports that the draft of this ship is 8.2m and it is currently located at the port of “Hamburg”. The second source S_2 which relies on S_1 revises the data it copies by updating the draft and the current location to 8.3m and “Marseille”, respectively. Finally, the third source S_3 independently sets the draft to 8.3m while being not aware about the location. Observe that S_3 , taken separately, retains *incomplete* information about the considered ship. Now, assume that one issues a query Q requesting the draft and the current

port of this ship based on the three sources. A deterministic data integration system will fail (or eventually it will make an arbitrary choice) by trying to merge the set of answers obtained from sources because there are contradictions resulting from two possible values for the draft and the port name. Eventually, one may prefer to know the set of all possible answers together with their probabilities. Indeed, the user may trust some specific sources and based on this he (or she) may filter the result. Observe that, the user may specify its preferences beforehand with the query. Filtering can also be done according to probabilities. A probabilistic approach for data integration eases this interaction while enabling to capture *contradictions*, *corroborations* and *incompleteness*. Modeling contradictions and corroborations, and thereby the correct probability values, need to take into account dependencies between the sources. Considering again our example, information about this ship can be obtained independently from S_1 and S_3 whereas requesting some piece of data from S_2 require to use S_1 . On the other hand, according to dependencies S_2 disagrees with S_1 while being corroborated by S_3 for the draft. Both scenarios call for being able to maintain the history of the evolution of each piece of information about shared objects during the integration process and according to source dependencies.

Envisioned Approach. Source dependencies, in terms of copying relationships, enquire about the history of the evolution of data items about shared objects (copiers may revise collected data as shown in our running example). This setting is very similar to a versioning process on the *web* implying a set of shared objects with uncertainty. Consequently, the problem of integrating web sources under uncertainty and dependencies can be answered by reformulating it as merging several uncertain versions of the same data. We proposed in [5] an uncertain version control model for tree-structured documents with uncertainty that computes the set of *possible versions* together with their *probabilities* by modeling data incompleteness, contradictions and corroborations with respect to the derivation links between data versions. As an enumeration of the set of all possible versions can be unfeasible at a certain limit, our model also includes an efficient compact way to reconcile as a whole all these possible versions.

In this vision paper, we present initial ideas towards a probabilistic XML approach for integrating web data sources under uncertainty and (deterministic) dependencies by getting some intuitions from our uncertain versioning model in [5]. As a data model, we choose XML because the tree-like structure of web data can be easily described with this format. We envision a probabilistic integration model able to represent and to assess the amount of uncertainty in both data and sources by modeling possible dependencies between sources. Those dependencies might enable (a) to properly detect data provenance, contradictions, correlations, etc. and based on this; (b) to trace the history of the evolution of each piece of information about shared objects. To do so, we associate *event variables* to uncertain sources in order to manage the amount of uncertainty in their own data (its really provided data) and their reliabilities. We adopt a *reconciled* data integration approach which abstracts the result of

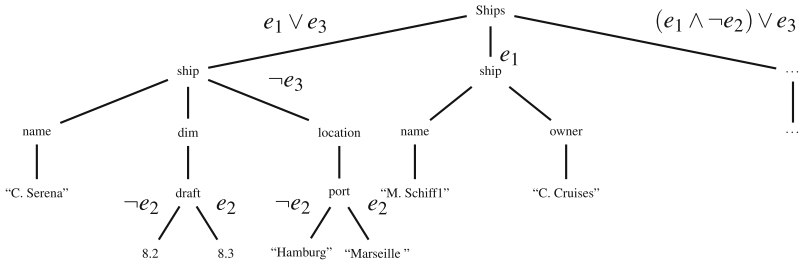


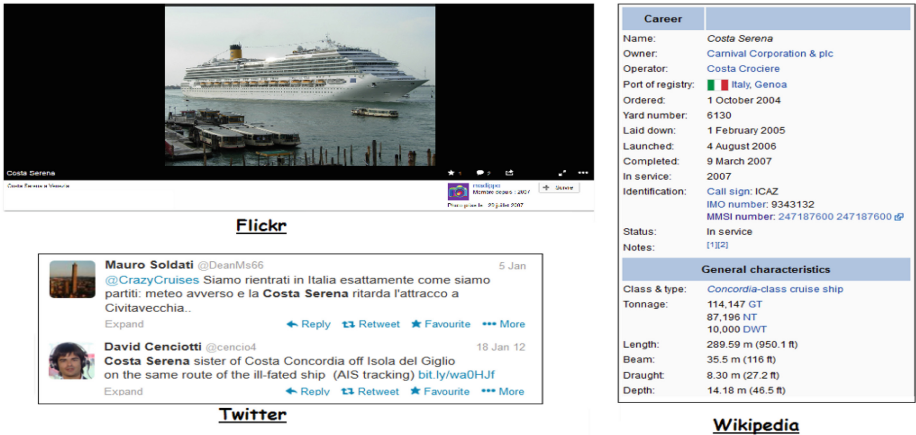
Fig. 1. Probabilistic XML encoding of the integration of shared (uncertain) objects

the integration of several web data sources under uncertainty and dependencies as a set of possible XML documents corresponding to sets of valid events. These sets of events can be used to estimate the probability values of these possible integration results given a probability distribution over the event variables. Concretely, we define the outcome of the integration, i.e., all possible XML integration trees, together with their mapping event sets, as a PrXML^{fe} probabilistic XML document (see Sect. 3 about its definition) similarly to [5]: nodes in this special tree are annotated by proportional formulas over event variables which track possible integration results and their probabilities with respect to dependency constraints. Figure 1 is the PrXML^{fe} probabilistic XML encoding of the result of the integration of sources S_1 , S_2 and S_3 of our running example with e_1 , e_2 and e_3 their respective associated events.

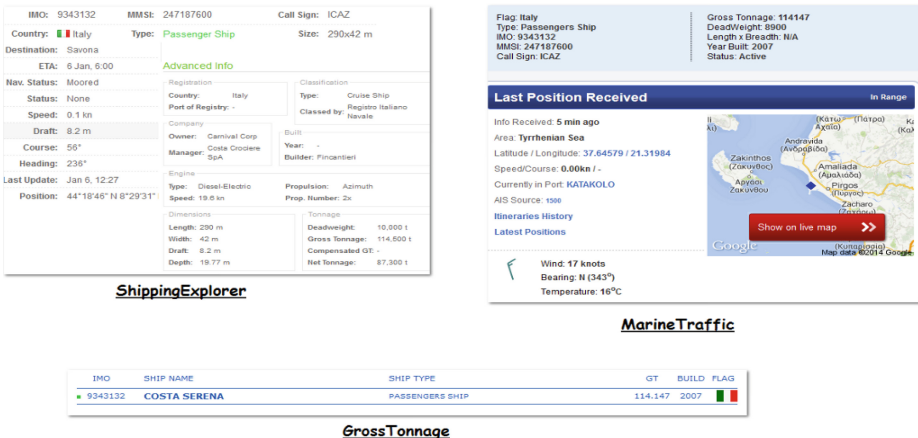
Outline. The rest of this paper is organized as follows. Section 2 motivates more the problem investigated in this paper by giving concrete examples from the maritime domain. Section 3 revisits some definitions pertaining for the modeling of web semi-structured data, semi-structured uncertain data and multi-version web data with uncertainty. Section 4 presents initial ideas towards a probabilistic XML approach for integrating uncertain tree-structured data sources under dependency constraints. Section 5 concludes the paper and presents some further work.

2 Motivating Application

Our motivating application is the integration of web sources providing information about objects (e.g., ships or ports) in the maritime domain. As we shall show, web sources in the maritime domain are various, uncertain and dependent. Therefore, users may want to simultaneously query or navigate through all these sources via a unique access point (or global reconciled view); to get answers from its trusted sources and; to know the real provenance of data.



(a) Social networks & collaborative platforms



(b) Sources monitoring in real-time maritime activities

Fig. 2. Example of web sources about objects in the maritime domain

2.1 Numerous Web Sources

We found that there are a lot of potential web sources by searching information about ships. Figure 2 shows a sample list of those sources¹. The sources in Fig. 2(a) consist of the social networks Flickr and Twitter, and the content-based collaborative platform Wikipedia. In Flickr and Twitter, users

¹ All the screen-shots given in Fig. 2 were captured January 8th, 2014 from <http://www.flickr.com/search/?q=CostaSerena>, http://en.wikipedia.org/wiki/Costa_Serena, <http://www.shippingexplorer.net/en/vessels/view/14429-costa-serena>, <http://www.marinetraffic.com/ais/details/ships/247187600>, and <http://www.grosstonnage.com/>.

share photos about ships and some useful information such as their names and their current locations in the form, for instance, of a tweet. In Wikipedia, users collaborate in order to collect the maximum amount of information about the general description of some particular boats. The sources in Fig. 2(b) are ShippingExplorer, MarineTraffic and GrossTonnage. These sources are mainly dedicated to the monitoring in real-time of the current locations and itineraries of ships even though they also give specifications of these latter. Current locations and itineraries can be known based on data transmitted by AIS systems on ships, for example. There are different levels of *heterogeneity* between the example sources, notably at *schema* level – fortunately, we can observe an homogeneity at object level because sources used in general same unique identifiers (e.g., IMO or MMSI) for objects such as ships. In the rest of the paper, we will assume that heterogeneity at schema and value levels is manually resolved with a certain amount of uncertainty.

2.2 Uncertain Web Data Sources

For the maritime domain, web sources are mostly uncertain due to imprecise and imperfect used data extraction methods. The AIS systems are inherently imprecise (e.g., they may transmit incomplete information) whereas human knowledge is imperfect. Other important features in sources revealing the presence of uncertainties are contradictions and incompleteness. Let us analyze the examples in

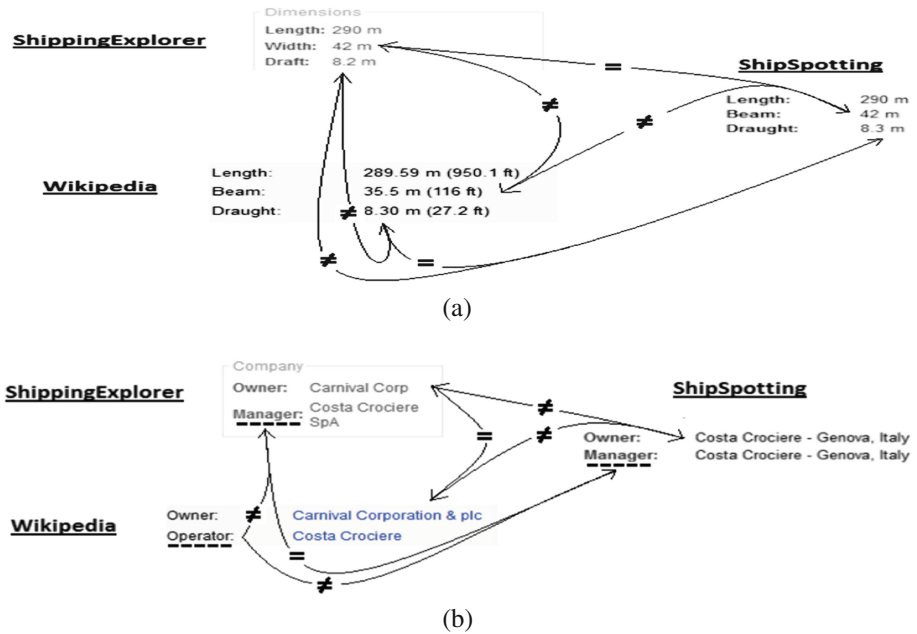



Fig. 3. Uncertain web sources: discrepancies and incompleteness

Figs. 3(a) and (b) which respectively give the company (owner and manager) and the dimensions (length, width and draft) of the same ship from ShippingExplorer, Wikipedia and ShipSpotting. Obviously, we can observe that ShippingExplorer and Wikipedia agree on the owner of this ship (even though Wikipedia seems to be more precise) whereas ShipSpotting gives a different owner. In contrast, all three sources agree on the manager for which the information from ShipSpotting is more complete. As for the dimensions of the ship, it also appears some contradictions between the values given by the sources. For instance, ShippingExplorer and ShipSpotting provide 42 m for the width while Wikipedia indicates 35.5 m. On another side, Wikipedia and ShipSpotting provide 8.3 m for the draft whereas ShippingExplorer indicates 8.2 m. For the draft, one may consider that the difference between the indicated values is not very important, and choose the value given by the majority of sources. Unfortunately, determining the correct values for the owner and the width seems to be a more complicated task. A rigorous way to manage contradictions is to keep all the possible values by estimating their *correctness* according to the reliability of the sources.

2.3 Dependent Web Sources via Copying Links

We observed based on the example in Fig. 4 that dependent web sources in the maritime domain is a reality. Indeed, there are some sources that collect their information from other ones by copying their data, or by aggregation in the case of several sources simultaneously involved. Such a copying relationship can be explicitly mentioned by the copier, for instance as in Fig. 4 (sketched screen-shot was captured January 8th, 2014 from <http://www.shipspotting.com/gallery/photo.php?lid=1825000>). However, in practice the copiers are not all known beforehand. Therefore, sometimes we are constrained to compute these dependencies (see [9,10] for details about the detection of copying relationships

Vessel Identification	Technical Data	AIS Information
 Name: Costa Serena IMO: 9343132 Flag: Italy MMSI: 247187600 Callsign: ICAZ	Vessel type: Passengers Ship Gross tonnage: 114,147 tons Summer DWT: 8,900 tons Length: 290 m Beam: 42 m Draught: 8.3 m	Last known position: 37°36'36.72" N, 20°50'12.48" E Status: Underway Speed, course (heading): 18.3kts, 89° (89°) Destination: Location: Katakolon Arrival: 8th Jan 2014 11:00:06 UTC Last update: 2 hours 25 minutes ago Source: AIS (AirNav ShipTrax)
	Additional Information	
	Home port: Genova Class society: Registro Italiano Navale Build year: 2007 Builder (*): Fincantieri Sestri Genova, Italy Owner: Costa Crociere - Genova, Italy Manager: Costa Crociere - Genova, Italy	


 Ship information by [AirNav ShipTrax](#) and [GrossTonnage.com](#). [Report error in ship details.](#)

Fig. 4. Example of dependence between Shipspotting, AirNavShipTrax and GrossTonnage

between multiple web sources). Since copiers may revise the collected data based on their own knowledge about the shared real-world objects, having the set of dependencies may help to find the real provenance of each data item and detect more easily contradictions, correlations etc. The detection of copying relationships between a set of web sources is beyond the scope of this paper. We consider in the following that the dependencies are given.

3 Data Model

We briefly present in this section some definitions pertaining for the modeling of web semi-structured data, semi-structured uncertain data and multi-version web data with uncertainty. We start by introducing unordered XML trees and a specific model of probabilistic XML trees we use in our integration system.

3.1 Unordered XML and p-Documents Based on Random Events

We model web data as unordered XML trees for convenience of the exposition. The consideration of an order between data items is left to future work.

Unordered XML. Let us consider a finite set L of strings (i.e., labels or text data) and a finite set I of identifiers such that their intersection is empty. We assume also given a labeling function Φ and an identifying function α .

Definition 1. *An unordered XML document is an unordered, unranked, labeled tree T of identifiers in I . The functions α and Φ map each node $x \in T$ respectively to a unique identifier $\alpha(x) \in I$ and to a string $\Phi(x) \in L$. In trees, nodes having at least one child refer to internal nodes, whereas nodes without children are leaves with data values.*

For modeling reasons, we use a same node (same label, same identifier) as root of all XML trees referring to the similar object. We omit node identifiers in tree examples for simplicity.

p-Documents Based on Random Events. A probabilistic XML document (abbr. *p-document*) is a compact way of representing a probability distribution over a set of possible unordered XML trees; in the case of interest here, this distribution is finite. A p-document is usually denoted by $\widehat{\mathcal{P}}$ and must be distinguished with a regular XML document as we will see next.

Definition 2. *A probabilistic XML distribution space over a set of uncertain XML trees is a pair (D, p) where D is a nonempty finite set of possible documents and $p : D \rightarrow (0, 1]$ is a probability function mapping each document d in D to a rational number, i.e., its probability, $p(d) \in (0, 1]$ with $\sum_{d \in D} p(d) = 1$.*

We draw our attention on the most expressive and succinct family of p-documents from [11], namely PrXML^{fie} model with fie standing for formula of independent events. For a complete insight about existing probabilistic XML models, see [1, 12]. Let B be a set of independent random Boolean variables (abbr. *event variables*) $b_1 \dots b_m$. The truth of each event variable b_i is given by its probability value $\Pr(b_i)$ of being valid. We revisit below the *syntax* and the *semantics* of the encoding of a probability distribution using a p-document of the PrXML^{fie} family.

Definition 3. A p-document $\widehat{\mathcal{P}}$ based on independent random variables is an unordered, unranked, and labeled XML tree in which (i) the root is always certain and; (ii) all other node x may be annotated with a propositional formula $\text{fie}(x)$ of events $b_1 \dots b_m$.

A proportional formula represents and estimates the amount of uncertainty in its attached node. Some distinct formulas may be correlated by sharing common events. At last, the number of event variables in the formulas is not necessarily the same. The set of all possible XML trees obtainable from $\widehat{\mathcal{P}}$ defines its *possible worlds*. Those possible worlds are produced in function of the different possible ways of valuating the event variables. A valuation ν of variables $b_1 \dots b_m$ is a mapping of each b_i to true or false. This valuation generates, when it is evaluated over $\widehat{\mathcal{P}}$, one particular XML tree $\nu(\widehat{\mathcal{P}})$ consisting only of nodes from $\widehat{\mathcal{P}}$ whose formulas are valuated at true with ν . We denote the possible worlds of $\widehat{\mathcal{P}}$ by $\mathcal{D}(\widehat{\mathcal{P}})$. Let $\mathcal{h}[\nu]$ be the set of all possible valuations over variables $b_1 \dots b_m$. The probability of a possible world $d \in \mathcal{D}(\widehat{\mathcal{P}})$ is given hereafter.

$$\Pr(d \mid d \in \mathcal{D}(\widehat{\mathcal{P}})) = \sum_{\substack{\nu' \in \mathcal{h}(\nu) \\ \nu'(\widehat{\mathcal{P}})=d}} \Pr\left(\bigwedge_{\substack{b_i \in \mathbf{B} \\ \nu'(b_i)=\text{true}}} b_i \wedge \bigwedge_{\substack{b_j \in \mathbf{B} \\ \nu'(b_j)=\text{false}}} \neg b_j\right). \quad (1)$$

Definition 4. The semantics $\llbracket \widehat{\mathcal{P}} \rrbracket$ of a p-document $\widehat{\mathcal{P}}$ in the probabilistic XML model based only on formula of independent random variables is the distribution (D, p) defined in such that (a) $D = \mathcal{D}(\widehat{\mathcal{P}})$ and (b) for all $d \in D$, $p(d) = \Pr(d \mid d \in \mathcal{D}(\widehat{\mathcal{P}}))$.

3.2 Semi-structured Multi-version Data with Uncertainty

In a XML setting, a semi-structured multi-version data with uncertainty (typically, shared web data in our context) is defined in [5] as evolving through uncertain updates and leading to uncertain versions. We summarize here this model which describes such a multi-version data with the help of two components: the *derivation graph* between the data versions (or version space) and a *probability distribution* over a set of possible XML tree versions. For more details about the model and its original context of use, we refer to [4, 5].

We suppose a set of complex event variables $e_1 \dots e_n$, each representing a conjunction of atomic event variables $b_1 \dots b_m$. Considered events model the

different uncertain states of the multi-version document. As a result, an event has also contextual information about a given version, in particular the edit script δ_i (i.e., a sequence of insertions and deletions) leading to it.

Definition 5. *A multi-version XML document with uncertainty is a pair (G, ω) where G is a directed acyclic graph (DAG) of events $\{e_0\} \cup \{e_1 \dots e_n\}$ representing the derivation graph of the tree versions, and ω is a probability distribution over the set of possible document versions.*

The special event e_0 is the root of G and maps to the initial state of the multi-version document. A version is an unordered XML tree mapping to a set of events in G whose edit scripts together made this version happen. Given the infinite set \mathbb{D} of all unordered XML trees, we have $\omega : 2^{\{e_1 \dots e_n\}} \rightarrow \mathbb{D}$ with (a) $\omega(\{\})$ a root-only XML tree and; (b) for all i , for all $F \subseteq 2^{\{e_1 \dots e_n\} \setminus \{e_i\}}$, $\omega(\{e_i\} \cup F) = [\omega(F)]^{\delta_i}$ ($[\omega(F)]^{\delta_i}$ results from applying δ_i on $[\omega(F)]$). This mapping corresponds to a probability distribution, compactly encoded in [5] as $\widehat{\mathcal{P}}$, over a set of possible trees versions.

Definition 6. *A compact representation system of a multi-version XML with uncertainty is a pair $(G, \widehat{\mathcal{P}})$ where (a) G is the DAG of events $e_0 \dots e_n$ and; (b) $\widehat{\mathcal{P}}$ is a $\text{PrXML}^{\text{fie}}$ p -document with random variables $b_1 \dots b_m$ encoding compactly all the possible tree versions and their mapping event sets.*

In this compact representation, the formula $\text{fie}(x)$ of a given node $x \in \widehat{\mathcal{P}}$ is of the form $\text{fie}(x) \leftarrow e_i \mid \text{fie}(x) \vee e_i \mid \text{fie}(x) \wedge \neg e_i$ for a certain event e_i . In this formula, *corroborations* and *contradictions* are captured as follows.

- If $e_i \models \text{fie}(x)$, then there is a corroboration of the presence (or validity) of x at this event e_i .
- If $e_i \not\models \text{fie}(x)$, then the event e_i contradicts the existence of x (or invalid x).

4 Heterogeneous Web Data Integration Using Probabilistic XML

We design in this section our probabilistic XML model dealing with the integration of web data sources under uncertainty and dependencies. We first present some challenges underlying the set up of the intended model. We then put forward a model and explain how it can be used for materializing the integration in the scenario where the dependencies between sources are *deterministic*.

Consider a set S of n web sources S_1, \dots, S_n under uncertainty and dependencies. For the convenience of the exposition, we assume that (a) all considered sources retain data about real-world objects in the same domain; (b) objects are distinguished each other by a unique identifier and; (c) each local source S_i provides data about its subset of tracked objects in the form of a global unordered XML tree that we denote also S_i . Moreover, we assume that, first, a dependency relationship involving two sources, if it occurs, is directed and there is no cycle; second, each dependency relationship is deterministic, that is, it is known with certainty.

4.1 Main Challenges

Three main requirements underlie the design of our intended data integration model for uncertain web data sources with dependencies.

1. At first, since we have uncertainties on the sources and on the data, we need a way to represent and to evaluate these uncertainties during the integration. The used model must be flexible enough to enable (a) correlating the uncertainty about the sources and the provided information; (b) tracking the provenance of each data item. Obviously, for instance, one may trust a given source but considers that its data are invalid. As a result, the model must enable *explanation* and *understanding* of obtained probability values.
2. Second, we have to introduce a technique for finding and representing the dependency relationships between sources; in this work we suppose that these dependencies are known beforehand.
3. Finally, the integration approach, formalizing the result of the integration and its semantics mapping with the local sources (or partial views of these latter), must be defined by focusing especially on the uncertain nature of our setting and the dependencies (a given source first copies other ones, and then it may revise the copied data with new knowledge). Therefore, the used model must enable the modeling of contradictions and corroborations in the integration outcome.

4.2 Probabilistic XML Integration System

Here we give a first attempt for formalizing our intended probabilistic XML integration framework. We go further on each requirement aforementioned by transposing the version control model with uncertainty, we designed in [5] (see Sect. 3 for a summary), in our data integration setting. We start by formalizing the problem.

A system with a set S_1, \dots, S_n of uncertain web sources with dependencies matches well with an uncertain versioning setting (G, ω) where ω covers all possible unordered XML trees over S_1, \dots, S_n with respect to the dependency graph G . As a result, we can reformulate the problem of data integration in this system as the integration of all the possible unordered trees defined by ω . In other terms, it will require a clear definition of the mapping between S_1, \dots, S_n and each possible unordered tree in ω together with its probability. Then, a materialization of the overall integration result. As we shall see in the next, we come up to materialize this integration using a PrXML^{fie} p-document $\widehat{\mathcal{D}}$ according to the dependency graph G .

Modeling Uncertainties. Similarly to [5], we use random event variables in order to deal with uncertainties. Consider again B as a set of independent random Boolean variables $b_1 \dots b_m$ and their probability values $\Pr(b_1) \dots \Pr(b_m)$ of being true as well. We restrict B to two types of disjoint sets of variables which we denote B_r and B_s . We use variables in B_r to manage the uncertainty in the

content really provided by each source: the data it does not copy from others sources (its contribution in a certain sense). Variables in B_s are used to model the trust one can have on sources or simply the source reputation. Given a source S_i , we refer to the uncertainty on its content and its reliability level with $b_{r,i}$ and $b_{s,i}$ respectively. We consider now the set of events e_1, \dots, e_n . In order to represent and evaluate the overall amount of uncertainty in each source S_i from S , an event $e_i = b_{r,i} \wedge b_{s,i}$ with $b_{r,i} \in B_r$ and $b_{s,i} \in B_s$ is associated to it. Intuitively, e_i is true when it produces a correct content on a reliable source. The probability associated to it is obtained by computing the probability of the corresponding conjunction. It estimates numerically its correctness.

Modeling Dependencies. As shown in [9,10], the dependencies between a set of web sources follow a DAG structure G . For our case, we consider here a DAG of events (representing implicitly sources) as nodes. Given the set of events e_1, \dots, e_n associated to the sources S_1, \dots, S_n , formally we consider the directed acyclic graph $G = (V, E)$ where V is the set $\{e_0\} \cup \{e_1, \dots, e_n\}$ of nodes and; $E \subseteq V \times V$ is the set of edges maintaining (implicitly) the dependencies between sources.

Materializing the Integration Outcome. As some previous work on uncertain tree-structured data integration, such as the paper of Van Keulen et al. [17], we build our system on a probabilistic model in a *reconciled* fashion. For this purpose, we introduce the notion of a *probabilistic XML global view* (PrGView) \mathbb{M} which is a set of *possible* integrated XML trees m_1, \dots, m_k having probabilities $\Pr(m_1), \dots, \Pr(m_k)$. An integrated XML tree m_i is defined as a deterministic unordered XML tree in \mathbb{D} resulting from the integration of changes from multiple XML tree versions. There is not only one possible way to integrate these tree versions, especially in the presence of uncertainty, but *several* describing, first, the views on the trust one may have on the given sources and the data; second, the different way to deal with contradictions and incompleteness of the data. We show later that PrGView enables to capture all such possible results of this integration.

Definition 7. Let S_1, \dots, S_n be a set of uncertain sources with a dependency graph G . A PrGView \mathbb{M} of an integration process over $\{S_1, \dots, S_n\}$ is a set $\{(m_1, \Pr(m_1)), \dots, (m_k, \Pr(m_k))\}$ where (i) for each $1 \leq i \leq k$, m_i is a possible integrated XML tree over S_1, \dots, S_n constrained by dependencies; (ii) $0 < \Pr(m_i) \leq 1$ with $\sum_{i=1}^k \Pr(m_i) = 1$.

We need to define how we obtain \mathbb{M} based on the set of input sources. We will focus more on the representation of the set of possible worlds than on their probabilities. We start by defining the *contribution* of a given source within our integration setting.

Definition 8. Let S_1, \dots, S_n be a set of uncertain sources with a dependency graph G . The contribution of any given source S_i w.r.t G corresponds to the real

content provided by this source. We encode the contribution δ_i of the source S_i in the form of a sequence of edit operations over some initial data.

Given any source S_i , let $G_{|\rightarrow e_i}$ be the set of sources on which S_i depends w.r.t G . That is, for each $1 \leq l \leq n$ with $i \neq l$, $S_l \in G_{|\rightarrow e_i}$ when the relation (e_l, e_i) holds in G . Algorithm 1 computes the contribution of S_i giving its dependent sources and their documents; DIFF in the algorithm is a *differencing function* (see [6, 14, 15] for more details) computing the difference between two unordered XML tree versions. Its output is a sequence of edit operations over XML nodes.

```

Input:  $G_{|\rightarrow e_i}$ 
Output:  $\delta_i$ 
1 Set  $S_0 \leftarrow$  root-only tree in  $\mathbb{D}$ ;
2 if  $G_{|\rightarrow e_i} == \emptyset$  then
3   | Set  $\delta_i \leftarrow$  DIFF( $S_0, S_i$ );
4 else
5   | foreach  $S_l$  in  $G_{|\rightarrow e_i}$  do
6     | Set each  $\delta_{l,i} \leftarrow$  DIFF( $S_l, S_i$ );
7     | Insert all insertions shared by  $\delta_{l,i}$ 's in  $\delta_i$ ;
8     | Insert all deletions in each  $\delta_{l,i}$  in  $\delta_i$ ;
9 return ( $\delta_i$ );
    
```

Algorithm 1. Computation of the contribution of a source

Additionally to variables, we trace the contributions $\delta_1, \dots, \delta_n$ of the different sources S_1, \dots, S_n using their associated events $e_1 \dots e_n$, respectively. By doing so, the events are enough to fully describe the sources because they contain the information about both the amount of uncertainty and the data of the considered sources. In the rest, by events, we will also mean the sources. We introduce a mapping ω between possible integrated XML trees in \mathbb{M} and the sources S_1, \dots, S_n by following a construction similar to the one given in [5]. Let $\mathbb{D}' \subseteq \mathbb{D}$ such that \mathbb{D}' includes $\{m_1, \dots, m_k\}$ in addition to a root-only tree.

Definition 9. Assume a set of uncertain sources S_1, \dots, S_n with a dependency graph G . Let \mathbb{M} be the PrGView of the integration of these sources. We consider the mapping $\omega : \mathbb{D}' \rightarrow 2^{\{e_1, \dots, e_n\}}$ as specifying the possible integrated unordered XML trees in PrGView in terms of data contained in the sources such that $\omega(\{ \})$ corresponds to the root-only tree in \mathbb{D}' and; for each $F \subseteq 2^{\{e_1, \dots, e_n\}} \setminus \{e_i\}$, $\omega(F \cup \{e_i\}) = [\omega(F)]^{\delta_i}$, i.e., integration of data originated uniquely from sources $F \cup \{S_i\}$. Let us assume that $m_k = \omega(F)$ for a fixed $1 \leq k \leq m$. The probability of m_k is estimated as follows.

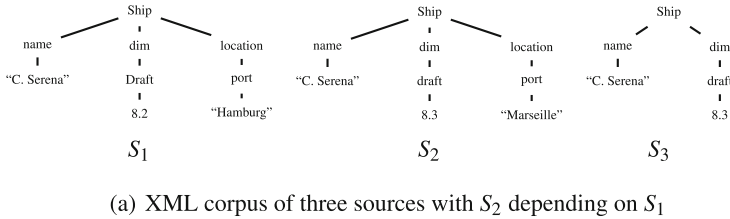
$$\Pr(m_k) = \sum_{\substack{F \subseteq \{e_1, \dots, e_n\} \\ \omega(F) = m_k}} \prod_{\substack{1 \leq i \leq n \\ e_i \in F}} \Pr(e_i) \times \prod_{\substack{1 \leq i \leq n \\ e_i \notin F}} 1 - \Pr(e_i). \tag{2}$$

The set of unordered trees specified with ω mapping can be reconciled as a PrXML^{fie} p-document $\widehat{\mathcal{P}}$, according to [5]. The ω mapping corresponds to the definition of PrGView.

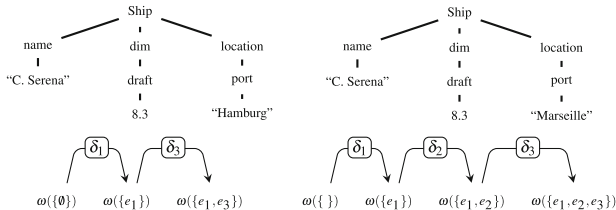
Definition 10. Assume a set of uncertain sources S_1, \dots, S_n with a dependency graph G . We introduce the PrXML^{fie} p-document $\widehat{\mathcal{P}}$ as a reconciliation in a global tree of the set of possible trees in the PrGView which defines the integration of S_1, \dots, S_n w.r.t G . This reconciled p-document is built w.r.t probabilistic encoding algorithm proposed in [5],

We conclude by first summarizing the formal model, and then showing through an example how it can be used in practice.

We abstract a probabilistic XML integration model over a set of sources S_1, \dots, S_n under uncertainty and dependencies with the help of a triple (G, \mathbb{M}, ω) where (i) G is a DAG of $\{e_0\} \cup \{e_1, \dots, e_n\}$ in which each node e_i , for $1 \leq i \leq n$, is associated to a source S_i in order to manage its overall amount of uncertainty and its contribution; (ii) \mathbb{M} is the PrGView of the integration of S_1, \dots, S_n and; (iii) ω is a mapping between the set of possible integrated trees in \mathbb{M} and the sources S_1, \dots, S_n through their associated events e_1, \dots, e_n . A reconciled representation system of this integration is a pair $(G, \widehat{\mathcal{P}})$ where (a) G remains



(b) Dependency DAG and contributions



(c) Sample of possible unordered XML integrated trees

Fig. 5. Probabilistic XML integration over XML corpus with uncertainty and dependencies

the same DAG of events e_0, \dots, e_n and; (b) $\widehat{\mathcal{P}}$ is the PrXML^{fie} reconciling based on events e_1, \dots, e_n and contributions $\delta_i, \dots, \delta_n$ the set of all possible integrated XML trees in \mathbb{M} .

Example 1. Figure 5 illustrates the probabilistic XML integration over our running example. Figure 5(a) shows XML corpus of the three sources S_1 , S_2 and S_3 where S_2 is a copier of S_1 . Figure 5(b) gives the dependency graph of the sources and their contributions δ_1 δ_2 , δ_3 which can be estimated with Algorithm 1. Figure 5(c) depicts two examples of possible integrated XML trees by reasoning on the validity or not of each event. The first integrated XML tree is obtained by only considering as valid e_1 and e_3 , thus corresponds, for instance, to the case where a user requests data from only the two independent uncertain sources S_1 and S_3 . The integrated version is obtained by evaluating δ_1 and δ_3 on $\omega(\{\})$ and $\omega(\{e_1\})$, successively – the evaluation of scripts follows the order in which events are given. We can generate all the possible XML integrated trees over the three given uncertain XML corpus under dependencies by following the same process.

5 Conclusion and Further Work

In this vision paper, we have exposed initial directions towards a probabilistic XML approach for integrating Web sources under uncertainty and dependencies. We first provided a concrete application of such a model. Then, we set up a first abstraction of our integration model by translating the problem in an uncertain version control setting. Besides a study of the effect of uncertain dependencies in the modeling of the set of possible worlds and a comparison of our approach against the literature (e.g., the probabilistic XML tree model in [17]), further work could explore the following issues.

- **Integration w.r.t. query views:** It could be also of interest to investigate the definition of the probabilistic XML global view in terms of query views, that is, defining the integration result w.r.t the type of queries issued over data. Such an approach is more suitable in the case for a virtual data integration where data reside in sources.
- **Uncertain integration and new knowledge:** Knowledge rules and user feedback can help to resolve a portion of uncertainties in the integration by refining the set of possible worlds (Cf. [16]). Relying on domain experts, e.g., opinions from maritime experts regarding our application domain, is a reliable way for obtaining such types of knowledge.

Acknowledgements. We are grateful to Pierre Senellart and Stephane Bressan for their precious remarks and suggestions. This work was partially funded by the NOR-MATIS project, and the French government under the STIC-Asia program, CCIPX project.

References

1. Abiteboul, S., Kimelfeld, B., Sagiv, Y., Senellart, P.: On the expressiveness of probabilistic XML models. *VLDB J.* **18**, 1041–1064 (2009)
2. Agrawal, P., Sarma, A.D., Ullman, J., Widom, J.: Foundations of uncertain-data integration. *Proc. VLDB Endow.* **3**, 1080–1090 (2010)
3. Ayat, N., Afsarmanesh, H., Akbarinia, R., Valduriez, P.: An uncertain data integration system. In: Meersman, R., et al. (eds.) *OTM 2012, Part II. LNCS*, vol. 7566, pp. 825–842. Springer, Heidelberg (2012)
4. Ba, M.L., Abdessalem, T., Senellart, P.: Merging uncertain multi-version XML documents. In: *Proceedings of DChanges, Florence, Italy* (2013)
5. Ba, M.L., Abdessalem, T., Senellart, P.: Uncertain version control in open collaborative editing of tree-structured documents. In: *Proceedings of Document Engineering* (2013)
6. Cobena, G., Abdessalem, T., Hinnach, Y.: A comparative study for XML change detection. In: *BDA* (2002)
7. Das Sarma, A., Dong, X., Halevy, A.: Bootstrapping pay-as-you-go data integration systems. In: *Proceedings of SIGMOD* (2008)
8. Dong, X., Halevy, A.Y., Yu, C.: Data integration with uncertainty. In: *Proceedings of VLDB* (2007)
9. Dong, X.L., Berti-Equille, L., Hu, Y., Srivastava, D.: Global detection of complex copying relationships between sources. *Proc. VLDB Endow.* **3**, 1358–1369 (2010)
10. Dong, X.L., Berti-Equille, L., Srivastava, D.: Integrating conflicting data: the role of source dependence. *Proc. VLDB Endow.* **2**, 550–561 (2009)
11. Kharlamov, E., Nutt, W., Senellart, P.: Updating probabilistic xml. In: *Proceedings of EDBT/ICDT Workshops* (2010)
12. Kimelfeld, B., Senellart, P.: Probabilistic XML: models and complexity. In: Ma, Z., Yan, L. (eds.) *Advances in Probabilistic Databases for Uncertain Information Management*. Springer, Heidelberg (2013)
13. Li, X., Dong, X.L., Lyons, K., Meng, W., Srivastava, D.: Truth finding on the deep Web: is the problem solved? In: *Proceedings of VLDB*, Sept 2013
14. Lindholm, T., Kangasharju, J., Tarkoma, S.: Fast and simple XML tree differencing by sequence alignment. In: *Proceedings on Document Engineering* (2006)
15. Peters, L.: Change detection in XML trees: a survey. In: *TSIT Conference* (2005)
16. van Keulen, M., de Keijzer, A.: Qualitative effects of knowledge rules and user feedback in probabilistic data integration. *VLDB J.* **18**, 1191–1217 (2009)
17. van Keulen, M., de Keijzer, A., Alink, W.: A probabilistic XML approach to data integration. In: *Proceedings of ICDE* (2005)

Skill Ontology-Based Model for Quality Assurance in Crowdsourcing

Kinda El Maarry¹(✉), Wolf-Tilo Balke¹, Hyunsouk Cho²,
Seung-won Hwang², and Yukino Baba³

¹ Institut für Informationssysteme, TU Braunschweig, Brunswick, Germany
{elmaarry, balke}@ifs.cs.tu-bs.de

² Department of Computer Science and Engineering, POSTECH,
Pohang-si, Korea

{prory, swhwang}@postech.ac.kr

³ The University of Tokyo, Tokyo, Japan
yukino_baba@mist.i.u-tokyo.ac.jp

Abstract. Crowdsourcing continues to gain more momentum as its potential becomes more recognized. Nevertheless, the associated quality aspect remains a valid concern, which introduces uncertainty in the results obtained from the crowd. We identify the different aspects that dynamically affect the overall quality of a crowdsourcing task. Accordingly, we propose a skill ontology-based model that caters for these aspects, as a management technique to be adopted by crowdsourcing platforms. The model maintains a dynamically evolving ontology of skills, with libraries of standardized and personalized assessments for awarding workers skills. Aligning a worker's set of skills to that required by a task, boosts the ultimate resulting quality. We visualize the model's components and workflow, and consider how to guard it against malicious or unqualified workers, whose responses introduce this uncertainty and degrade the overall quality.

Keywords: Crowdsourcing · Quality assurance · Skill ontology · Uncertain data

1 Introduction

The hope of being able to somehow benefit from “the wisdom of the crowd” [1] is the main driver for the rising popularity of crowdsourcing [2], coupled with the information flood and the flexible and relatively cheap solution that today's crowdsourcing platforms offer. Thus, more and more companies and organizations are turning to crowdsourcing. Some notable names include: NASA, Threadless, iStockphoto, InnoCentive, etc. [3]. Yet, the question of automatically assuring the returned quality of results [4] and the uncertainty that is associated with it, remains an unsolved question [5]. This is because checking every single submitted response is costly, time consuming and threatens to invalidate most of the crowdsourcing gains. This in turn encourages unethical workers to submit low quality results.

Most requestors end up relying on redundancy or repeated labeling as means of verification of user performance. A common approach tests the reliability of users by blending a set of questions for which the answers are known, so-called gold questions, into the workload. This instantly raises the question of how many gold questions should be included [6]. Another possibility is the assignment of multiple workers to the same task and then aggregating their responses. While both approaches pose problems for tasks where the comparison between individual workers' results is difficult (see [7] for a detailed discussion), the redundancy approach usually incurs monetary costs and places the costs for quality control at the task provider's doorstep. Moreover, popular techniques for aggregation like e.g., majority voting, have been shown to suffer from severe limitations [8].

To correctly tackle the issue of uncertainty, various factors affecting the quality of the responses were investigated [9]. While results show monetary incentives having an effect on quality in contrast to the experimental results in [10], it's still rather tricky; low paid jobs yield sloppy work, and highly paid jobs attract unethical workers. Another investigated factor was workers' qualification, where not only was it shown that qualified workers produce better quality and strive to maintain their qualification level, but in a setup that relies on qualifications for task assignments, unqualified workers are pushed to diligently work on improving their own qualifications.

To that end, we investigate a skill ontology-based model to be adopted by crowdsourcing platforms, which aims at identifying those qualified workers and assigning them to the tasks they're eligible to. This can be realized through identifying the skills required to adequately work on a task, and aligning it to the skills a certain worker has. Consequently, by excluding non-qualified workers or non-ethical workers who falsely try to build up their qualifications, the model would be practically excluding the sources of uncertainty introduced to the data altogether.

The rest of the paper is organized as follows. We start off by reviewing the current related work. Next, we define what quality stands for in a crowdsourcing setup and identify the different types of quality that our model needs to realize. Section 4 presents in details the proposed skill-ontology model. This is followed in Sect. 5 by an overview of the model's workflow. Finally, in the last section, we provide a summary and an outlook on future work.

2 Related Work

In recent years, many web-based collaboration platforms and marketplaces are relying on that same "wisdom of the crowd" ideology, where anonymous users' contributions are in some way combined to provide innovative and diverse services. Threadless (online t-shirt design contest) [11] and iStockPhoto, are two prominent examples exploiting that ideology. PodCastle [12] presents an audio document retrieval service "Pod-Castle", which collects anonymous transcriptions of podcast speech data to train an acoustic model. This was followed two years later by an alternative crowdsourcing-based approach [13]. These examples support the main argument in [14], i.e. that the way people collaborate and interact on the web has been so far poorly leveraged through the existing service-oriented computing architectures.

So instead, a mixed service-oriented system i.e. service-oriented crowdsourcing, is desirable, enabling a more seamless approach, which would also exploit the on-demand allocation of flexible workforces. This steers the trend ever more towards crowdsourcing, which is now being offered by many platforms: Amazon Mechanical Turk, Samasource, Crowdfunder, etc. However, every chance needs to overcome challenges, and the main challenge here is that crowdsourcing results are often questionable in terms of their quality, and the associated uncertainty introduced in aggregated results becomes an issue.

This is actually very similar to the missing confidence in third-party services which posed serious issues in the web services community, see e.g., [15]. One solution here was to adopt credentials proving the eligibility of each discovered service. Simply put, a service is eligible if it meets certain quality requirements (in functionality, as well as typical QoS parameters like availability or response time). When composing complex workflows out of individual services these quality requirements can be interpreted as mutual agreements. Such agreements can be expressed for example by Web Service Level Agreement Language (WSLA) [16], or Web Service Management Language (WSML) [17]. In our context, a service provider is none other than a worker who has some skills and a task provider's confidence in results would be based upon the worker's credentialed skills. These credentials can be attained by passing a standardized test or a personalized test that the provider designs for that particular task. An agreement is reached, when a worker's credentialed skills matches those listed by the task provider (requestor) as the exact skills required for the corresponding task.

A lot of work in crowdsourcing literature has already been devoted to mitigate such quality concerns. The solution of redundancy and repeated labeling was first expanded by Dawid and Skene [18], who took into consideration the response's quality based on the workers. Through applying an expectation maximization algorithm, the overall error rate for each worker can be computed. Other approaches that estimate these error rates includes: a Bayesian version of the expectation maximization algorithm approach [19] and a probabilistic approach that takes into account both the worker's skill and the difficulty of the task at hand [20]. A further step was taken in [21] with an algorithm separating the unrecoverable error rates from recoverable bias.

Here, rather than looking at the worker's error rates, we aim at identifying the workers who are a good match for the corresponding task. Each worker has a skill profile, and every skill in the ontology is associated with a library of assessments. These assessments validate whether a worker indeed possesses the necessary skill or not. Both can be viewed analogously to the competence profiles provided by learning objects – entities that are used for task-focused training or learning in the IEEE 1484.12.1–2002 Standard for Learning Object Metadata.

These skills can be managed and referred to in a skill ontology. This fortunately leads us to a rich literature to derive and adapt from, which has been devoted to building competencies models, see [22] and [23]. Competency covers: knowledge, experience, skill and willingness to achieve a task. Such models have been used for quite a long time in organizations to help identify and attract suitable workers, as well as to help the workers acquire the needed skills. In order to identify the skills required for a task, skill gap analysis can be used to create the task's corresponding

competency map [24]. Workers having the corresponding skills in a task's competency map could be then identified through competency matching. [25] Formalizes another approach that focuses on Ontology-based semantic matchmaking between demanded skills (skills required by a task in our case) and supply (the workers possessing that skill). However, competency models still have their own challenges. Given their complexity, competencies have to be precisely defined within the different specific domains. Moreover, developing assessments that can truly capture one worker's competency level is unfortunately very often underestimated [26].

But of course, assigning the right worker for a task involves much more than just choosing the workers based on their skills. A worker maybe be highly competent relative to the task he's assigned to, yet his work ethics may earn him a bad reputation This might simply boil down to wanting to finish a task as fast as possible and with the least effort incurred. So the overall quality is in fact affected by both the workers' skills and reputation. This elicits the need for deploying quality control measures, whether in design time, run time or both, see [27] for a more comprehensive list of these measures. Computing workers' reputations poses a real challenge, and many reputation approaches have been investigated whether it's based on a reputation model [28], on feedback and overall satisfaction [29], or on deterministic approaches [30], etc.

3 Types of Quality

Upon addressing the data uncertainty that arise with crowdsourcing tasks, different aspects of quality can be identified. This breakdown allows us to identify the corresponding quality assurance mechanisms, which needs to be addressed by the proposed model. A detailed description of each of those quality aspects follows next.

3.1 Result's Quality

Comes first to mind, and covers both the requester's expectations and the usefulness of the results. In terms of requestor's expectations, the structure of the returned results will be heavily influenced by what the requester wants and expects. Accordingly, the crowdsourcing task should be designed in a way that elicit that specific structure in the returned results (factual correctness in the form of a yes/no answer, consensus, opinion diversity, opinion quality, etc.). In terms of usefulness, requestors may also measure the quality in terms of how the results are consistent or abiding to the task description, or whether they are transparent and traceable .i.e. there exists a logical pattern the worker followed to give that response.

3.2 Platform's Quality

Refers to the usability of the platform, where a platform's interface and offered tools should equally support both workers and requesters. For workers, the platform should promote a fair working environment. Fairness encompasses: (1) guaranteed payments, (2) nondiscriminatory conduct, (3) payments matching the corresponding load of work. For the requesters, the platform should offer an adequate set of tools to easily

and efficiently: (1) upload data and download results, (2) design tasks, (3) automatically assign qualified workers, (4) block spammers, (5) train workers.

3.3 Task's Quality

At a lower granularity, the quality of the task directly affects the results' quality. A requestor should: (1) identify the set of skills required to accomplish a task, (2) describe the task clearly, (3) define the expected effort in terms of complexity or time required to finish, (4) design the task's interface to support an easier workflow for the worker.

3.4 Worker's Quality

Refers to how fit a work is for the task at hand. Namely, how qualified and prepared they are to do the task. On one hand, qualified can be mapped to skill levels and how relevant these skills are to the task. On the other hand, prepared can be translated into willingness to complete the task to the best of ones skills. Other contributing factors are: (1) workers' availability, (2) flexibility of working hours (Both can be easily monitored through activity logs), (3) workers' reputation. (Can be based on history and average satisfaction score attained upon the completion of a task).

These different aspects will often in reality be interleaved. For instance, the clarity of a task is not only related to a task's quality, but might also fall under the platform's quality, where a platform ensures that the workers get clear task description that helps them avoid getting penalized if they do the task incorrectly due to vague guidelines.

4 Skill Ontology-Based Model

Following the quality aspects we identified in Sect. 3, we propose a skill ontology-based model to be adopted by crowdsourcing platforms. The model aims to capture the different aspects of quality that helps diminish the resulting uncertainty by eliminating one of its major sources: unqualified workers. The skill ontology-based model roughly comprises of: (1) skill ontology, (2) ontology merger (3) skill's library of assessments, (3) Skill aligner, (4) reputation system and a (5) task assigner.

4.1 Basic and Temporary Skill Ontologies

At the model's core lies the skill ontology. The model maintains a dynamic ontology, which evolves with the crowdsourcing platform's demands. While some skills will be often required for many tasks e.g. language skills for translating tasks, other skills will be highly specific and tailored for a specific task e.g. identifying the family, genus and species a fish belongs to. Accordingly, two ontologies are maintained: a basic and a temporary one. The basic skill ontology retains those skills that are highly demanded by many tasks. The temporary skill ontology retains newly added skills. Later on, only

those skills that were frequently required by many tasks are transferred from the temporary ontology to the basic one.

A requestor is always presented with a single consolidated ontology (or an automatically generated taxonomy as presented in Subsect. 4.4), in which he can browse the skills required for the task he's designing. When the required skill isn't available in the ontology, the requestor can define a new skill.

4.2 Ontology Merger

A new skill, which has been newly defined by a requestor is initially added to the temporary skill ontology. Every defined skill must be associated with at least one assessment. The new skill resides in the temporary ontology until it: (1) has proven to be popular (2) has been verified. Popular skills are skills that were required not only by many tasks, but also by many different requestors. Verified skills are skills that are associated with at least one verified assessment as will be further explained next.

4.3 Skill's Library of Assessments

Identifying whether a worker has a certain skill or not, can be ascertained through an assessment. A skill's library of assessments may comprise two types of assessments: standardized and personalized assessments. For standardized assessments like: TOEFL for the English language, or MOOC (Massive Open Online Course) certificates, most requestors will approve and conclusively trust them. However, when a requestor doesn't, or when there is simply no standardized test for the required skill, the requestor can create a personalized assessment. We assume here that requestors, who're requiring more specific skills or who're not satisfied with the available assessments, are willing to invest time to enforce higher quality as per their own standards.

Standardized assessments are inherently verified, since their legitimacy are already proven. On the other hand, personalized assessments, require further investigation for verification. Consider the following scenario: A worker posing as a requestor, creates a new personalized assessment and uploads it for the skill he/she wants to attain. Providing the perfect answers for these personalized assessments becomes then trivial, and the worker can accumulate endless skills in this manner. Assessments' verification can be done manually or automatically (platform-wise or crowd-wise).

1. Manual verification: entails hiring an expert to look over the assessment, this however costs both time and money. Accordingly, as a rule of thumb, this should be limited to cases where a popular skill has only one personalized assessment or multiple personalized assessments from the same requestor.
2. Automatic verification: serves as an alternative to manual verification, when the skill has: at least one verified personalized assessment, or one standardized assessment.

- **Automatic platform-wise verification:** The platform creates a new personalized assessment, merging the original questions with those from different verified assessments available in the corresponding skill's library of assessments. If workers can also answer the newly merged questions, the assessment is verified and can be later on used on its own.
- **Automatic crowd-wise verification:** Workers who have the corresponding skill in their skill profile, can verify the assessment and earn a higher reputation. Note that, extra measures need to be taken, to avoid workers who maliciously aim at boosting their reputation by creating spam assessments and reporting them later as spam. Accordingly, unlike the task assignment, the workers are automatically assigned a random assessment, rather than choosing one.

Until a personalized assessment is verified, workers are allowed to take. If the workers suspect the assessment to be a spam, they must report it. If not, they may take it and acquire a pending-verification skill in their profile upon passing the assessment. When the assessment is verified, all the corresponding pending-verifications skills are updated. If the assessment was merely spam, workers' who failed to report the assessment as such are penalized, and the corresponding pending-verification skill is revoked.

4.4 Skill Aligner

Upon creating a task with a set of prerequisite skills, a requestor can choose to either use one of the available skills in the ontology or define a new one. Choosing one of the skills in the ontology can be a tiresome job, especially since the ontology grows with the needs of the crowdsourcing platform. Ideally, a requestor should be able to quickly see whether the required skill is available in the ontology or not. To that end, a taxonomy can be maintained on top of the ontology, which the requestor can quickly traverse. This taxonomy can be automatically built from the skill description and keywords the requestor inputs upon adding the new skill [31], and validated by the crowd. Given the set of prerequisite skills that the requestor specifies, the skill aligner should be able align those skills to the similar available skills in the taxonomy and present those to the requestor. This could be based on matching the skill keywords and descriptions.

4.5 Reputation System

To ensure high quality, only qualified workers should be assigned to the corresponding task. Qualified workers are those workers who: (1) have the required skills, (2) are willing and motivated to complete the tasks, (3) are available, and (4) are highly reputable. Each of those can be respectively measured as follows.

1. **Skill profile:** The skill profile holds the worker's list of skills. The profile acts as a primary filter, where only those workers having a task's prerequisite set of skills are considered. A Skill profile can hold two types of skills: (1) verified skills, (2)

pending-verification skills. For every skill in the worker's profile, a list of all the tasks that the worker utilized the corresponding skill in are compiled. Furthermore, an accompanying score is attached, reflecting this experience. This score can be derived from the compiled lists of tasks. Only completed tasks with a positive feedback are listed i.e. requestor was satisfied with the worker. Completed tasks with a negative feedback, are only reflected in the skill's score. This gives a chance for the worker to improve his skill, without having a permanent black spot in their skill profile.

2. **Willingness:** A worker's willingness can be captured from his crowdsourcing platform activity, the following can be observed: (1) time needed to finish a job versus that set by the requestor as the optimal processing time for the task to be done, (2) ratio of completed to aborted tasks.
3. **Availability:** A worker's availability can also be captured from the worker's activity log on the crowdsourcing platform, by specifically noting the number of hours the worker logs per day or month. The time zone a worker is in plays an important role, for urgent tasks i.e. assigning workers with different time zones to the requestor's saves time, where requested tasks can be simply finished overnight.
4. **Reputation:** A worker's reputation can be derived from the average requestor's satisfaction. Moreover, the worker's reputation is penalized, when a pending-verification skill proved to be spam. In addition to such a penalizing system, a reward system can also be in place e.g. Workers contributing in the automatic crowd-wise verification of assessments. To promote fairness and protect the workers from malicious requestors, a similar reputation could be built for the requestors, even if it's spanning only over one job. A malicious requestor will tend to be malicious in general, which can be easily tracked. In such a case, the requestor's feedback can be ignored.

4.6 Task Assigner

Initially only those workers with the required skills are considered for a task. A ranking based on the combination of the willingness, availability and reputation measures is then provided. The three measures are by default equally weighted. The requestor can however choose to give higher weight for any of those measures. E.g. availability is more critical than willingness. A requestor may also choose to completely disregard any of the measures e.g. availability is of no importance. Ultimately, workers exceeding the quality threshold defined by the requestor are assigned to the task. Furthermore, responses of workers with higher ranking are given a higher weight.

5 Workflow of the Skill-Ontology Based Model

The skill-ontology based model's workflow can be functionally broken down into: requestor-side, platform-side and worker-side for ease of illustration as follows.

Figure 1 gives a graphical overview of the various components of the model as well as the system’s interactions.

1. **Requestor-side:** After the requestor designs the task according to his needs, the list of skills required for that task has to be specified. To that end, the requestor checks the taxonomy of skills provided by the platform. When the required skill is found, the requestor simply adds it in the task’s list of required skills. Checking the skill’s library of assessments, the requestor chooses the assessments he approves and deems eligible for the task’s requirements. If no such assessment is found, the requestor is free to design an assessment of his own, which is then added to the skill’s library of assessments as an unverified assessment. On the other hand, if the requestor never finds the required skill from the start, he can add a new one along with at least one assessment. The new skill is initially added to the temporary Ontology. If the defined assessment is a standard assessment, no verification is needed, otherwise it’s added as an unverified assessment. In addition to the list of required skills, the requestor defines a threshold for the worker’s quality to be employed, as well as the measures of quality (willingness,

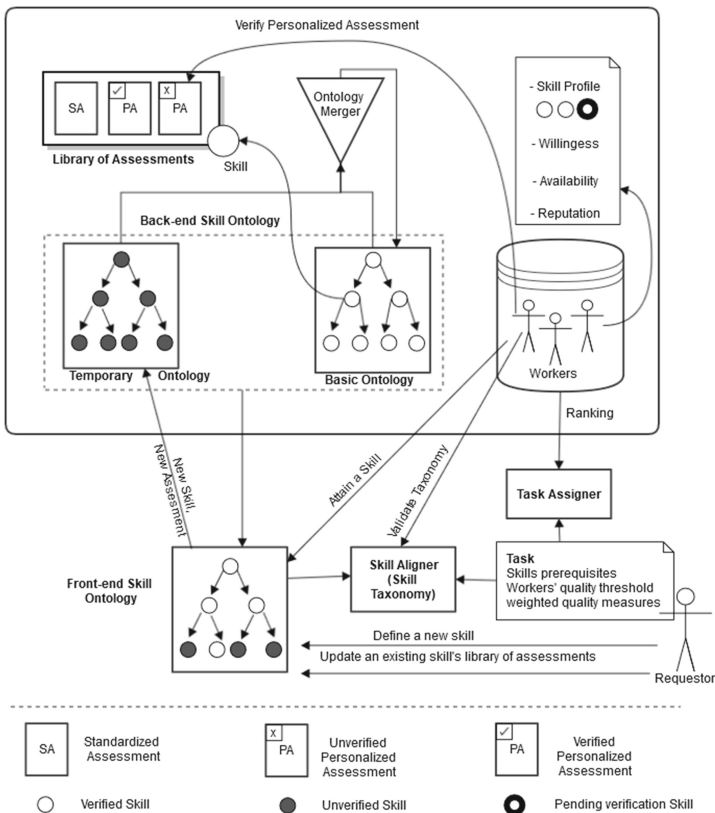


Fig. 1. Skill-Ontology based model workflow

availability, reputation) he wants to consider and their corresponding weights of importance.

2. **Platform-side:** The platform maintains at the back-end two ontologies: Temporary and Basic ontology. On the requestor's front end, a view that combines both ontologies is provided. The front-end ontology may or may not reflect the basic ontology at a given time, and may include both verified and unverified skills. Popular unverified skills that are in the temporary ontology are merged with the basic ontology upon verification. Every skill is associated with a library of assessments that holds either standardized assessments and/or personalized assessments. Furthermore, the platform maintains a database of workers, associating each work with a profile of skills (verified, pending verification) along with their computed measures of quality.
3. **Worker-side:** A worker is free to choose the tasks he wants to be considered for. Only when his skill profile contains the required skills for the corresponding task is he considered for the task. A worker can at any time expand his skill profile, by sitting assessments and attaining new skills. Workers may also boost their reputation by: (1) verifying personalized assessments (2) validating the platform's generated skill taxonomy.

6 Summary and Outlook

Uncertainty is inevitable when dealing with crowdsourcing results. We defined different aspects of quality to identify the corresponding quality assurance measures that should be present. Next, we proposed a skill ontology-based model to be adopted by crowdsourcing platform as a management technique. At its core, the model diminishes the existing uncertainty by eliminating unqualified workers. This is attained by maintaining a dynamically evolving ontology of skills, with libraries of standardized and personalized assessments for awarding credentialed skills. After aligning a worker's set of skills to that required by a task, the resulting quality is improved, where only qualified workers are assigned to the task. Furthermore, in such a setup, qualified workers strive to maintain their qualification level, and unqualified workers are pushed to diligently work on improving their own qualifications. We investigated the model and its workflow on a top level, however, the feasibility of maintaining such a model needs to be further investigated. As examined in the related work section, our model is closely related to web services, reputation-based systems and competency models. Further literature needs to be thoroughly examined, and accordingly adapted to leverage the current model. Furthermore, the proposed workers' quality measures that's to be computed should be formally defined.

Acknowledgments. We'd like to thank the organizers of NII Shonan 2013 meeting for *Intelligent Information Processing - Chances of Crowdsourcing*, which spurred this work. We'd also like to thank the reviewers for their careful examination and insightful comments and remarks, which we tried to adopt for improvements.

References

1. Surowiecki, J.: *The Wisdom of Crowds*, p. 336. Anchor, New York (2005)
2. Howe, J.: The rise of crowdsourcing. *North* **14**(14), 1–5 (2006)
3. Brabham, D.C.: Crowdsourcing as a model for problem solving: an introduction and cases. *Convergence Int J. Res. New Media Technol.* **14**(1), 75–90 (2008)
4. Kamps, J., Geva, S., Peters, C., Sakai, T., Trotman, A., Voorhees, E.: Report on the SIGIR 2009 workshop on the future of IR evaluation. *ACM SIGIR Forum* **43**(2), 13 (2009)
5. Zhu, D., Carterette, B.: An analysis of assessor behavior in crowdsourced preference judgments. In: *SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation*, no. Cse, pp. 17–20 (2010)
6. Scoring Workers in Crowdsourcing: How Many Control Questions are Enough?.pdf, 2013
7. Lofi, C., Selke, J., Balke, W.-T.: Information extraction meets crowdsourcing: a promising couple. *Datenbank-Spektrum* **12**(1), 109–120 (2012)
8. Kuncheva, L.I., Whitaker, C.J., Shipp, C.A., Duin, R.P.W.: Limits on the majority vote accuracy in classifier fusion. *Pattern Anal. Appl.* **6**(1), 22–31 (2003)
9. Kazai, G.: In search of quality in crowdsourcing for search engine evaluation. *SIGIR Forum* **44**(2), 165–176 (2011)
10. Mason, W., Watts, D.J.: Financial incentives and the ‘performance of crowds’. *ACM SIGKDD Explor. Newsllett.* **11**(2), 100 (2010)
11. Brabham, D.C.: Moving the crowd at threadless. *Inf. Commun. Soc.* **13**(8), 1122–1145 (2010)
12. PodCastle: Collaborative training of acoustic models on the basis of wisdom of crowds for podcast transcription, (2009). <https://staff.aist.go.jp/m.goto/PAPER/INTERSPEECH2009ogata.pdf>
13. Goto, M., Ogata, J.: Podcastle: recent advances of a spoken document retrieval service improved by anonymous user contributions. In: *Proceedings of the 12th Annual Conference of the International Speech Communication Association (Interspeech 2011)*, pp. 3073–3076 (2011)
14. Schall, D.: *Service-Oriented Crowdsourcing: Architecture, Protocols and Algorithms*, p. 105. Springer, New York (2012)
15. Lai, C.: Endorsements, licensing, and insurance for distributed system services. *J. Electron. Publishing* **2**(1) (1996)
16. Ludwig, H., Keller, A., Dan, A., King, R.: A service level agreement language for dynamic electronic services. In: *Proceedings of 4th IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002)* (2002)
17. Sahai, A., Machiraju, V., Anna, D.: Towards automated SLA management for web services (2002). <http://www.hpl.hp.com/techreports/2001/HPL-2001-310R1.pdf>
18. Dawid, A.P., Skene, A.M.: Maximum likelihood estimation of observer error-rates using the EM algorithm. *J. Roy. Stat. Soc.: Ser. C (Appl. Stat.)* **28**(1), 20–28 (1979)
19. Raykar, V.C., Yu, S., Zhao, L.H., Valadez, G.H., Florin, C., Bogoni, L., Moy, L.: Learning from crowds. *J. Mach. Learn. Res.* **11**, 1297–1322 (2010)
20. Whitehill, J., Ruvolo, P., Wu, T., Bergsma, J., Movellan, J.: Whose vote should count more: optimal integration of labels from labelers of unknown expertise. *Adv. Neural Inf. Process. Syst.* **22**(1), 1–9 (2009)
21. Ipeirotis, P.G., Provost, F., Wang, J.: Quality management on amazon mechanical turk. In: *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pp. 64–67. ACM, New York (2010)

22. Champion, M.A., Fink, A.A., Ruggeberg, B.J., Carr, L., Phillips, G.M., Odman, R.B.: Doing competencies well: best practices in competency modeling. *Pers. Psychol.* **64**(1), 225–262 (2011)
23. Shippmann, J.S., Ash, R.A., Battista, M., Carr, L., Eyde, L.D., Hesketh, B., Kehoe, J., Pearlman, K., Prien, E.P., Sanchez, J.I.: The practice of competency modeling. *Pers. Psychol.* **53**, 703–740 (2000)
24. De Coi, J.L., Herder, E., Koesling, A., Lofi, C., Olmedilla, D., Papapetrou, O., Siberski, W.: A model for competence gap analysis. In: *WEBIST 2007: Proceedings of the 3rd International Conference on Web Information Systems and Technologies*, pp. 304–312 (2007)
25. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Mongiello, M., Mottola, M.: A formal approach to ontology-based semantic match of skills descriptions. *J. Univ. Comput. Sci.* **9**(12), 1437–1454 (2003)
26. Koeppen, K., Hartig, J., Klieme, E., Leutner, D.: Current issues in competence modeling and assessment. *Zeitschrift für Psychologie/J. Psychol.* **216**(2), 61–73 (2008)
27. Allahbakhsh, M., Benatallah, B., Ignjatovic, A., Motahari-Nezhad, H.R., Bertino, E., Dustdar, S.: Quality control in crowdsourcing systems: issues and directions. *IEEE Internet Comput.* **17**(2), 76–81 (2013)
28. Allahbakhsh, M., Ignjatovic, A., Benatallah, B., Foo, N., Beheshti, S.M.R., Bertino, E.: Reputation management in crowdsourcing systems (2012)
29. Ignjatovic, A., Foo, N., Lee, C.T.: An analytic approach to reputation ranking of participants in online transactions. In: *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 1 (2008)
30. Noorian, Z., Ulieru, M.: The state of the art in trust and reputation systems: a framework for comparison. *J. Theor. Appl. Electron. Commer. Res.* **5**(2), 97–117 (2010)
31. Liu, X., Song, Y., Liu, S., Wang, H.: Automatic taxonomy construction from keywords. In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 1433–1441 (2012)

ProbKS: Keyword Search on Probabilistic Spatial Data

Feng Gao^{1,2,P}, Rohit Jain^{1(✉)}, Sunil Prabhakar¹, and Luo Si¹

¹ Department of Computer Sciences, Purdue University, West Lafayette, IN, USA
{gao141,jain29,sunil,lsi}@cs.purdue.edu

² Fudan University, Shanghai, China

Abstract. Many applications, like Twitter, Yelp, or Facebook, produce documents that are tagged with geolocations. For example, when a user *tweets* using Twitter, the *tweets* are tagged with the user’s location (inferred using the user’s IP address, or mobile GPS). These locations, however, are computed with inherent uncertainty. In such scenarios, it is desired to support search queries that take into account both text relevancy and location proximity. In this paper, we study the problem of text retrieval queries on probabilistic spatial data. We consider top- (c, k) queries to capture semantics of both textual relevance and probabilistic location proximity. A top- (c, k) query returns k tuples which have the highest probability of being in the top- c query results under the possible world semantics. We propose a framework to answer such queries. Our framework integrates two components: scoring textual similarity based on the query text; and the document text and calculating top- c confidence based on the probability of the document falling within the query region. We develop an IRTree-based Incremental Scoring Approach (ISA) that returns an iterator over tuples in decreasing order of text similarity. Our parameterized probabilistic ranking algorithm $PRank^c$, consumes the output of ISA interactively and calculates top- c confidence of these tuples in linear time. We also provide a heuristic optimization to terminate the $PRank^c$ algorithm earlier without compromising on result quality. We conduct experiments on real data to show the efficiency of this framework.

1 Introduction

As location-aware devices get widely used, many applications such as Twitter, allow users to produce data that are tagged with the geo-location of the user. Much work has been done on spatial keyword queries [2, 5, 7] for retrieving information from such spatial textual data. These works assume that the geolocations of the documents were computed with certainty. However, in many scenarios, the geographic location does not refer to a unique place, but an uncertain region. For example, when a user *tweets* on Twitter using a computer (which is not location aware), the user location is inferred using the IP address. However, this location cannot be computed with certainty resulting in a probabilistic spatial

Table 1. Twitter sample data

ID	Name	Tweet	Locations
t_1	Steve	Kobe Bryant, Obama of the NBA game	$\{l_1: .3, l_2: .5, l_3: .2\}$
t_2	Chris	Kobe game-typing 3 pointer in the 2004 NBA Finals	$\{l_4: .5, l_5: .3, l_7: .2\}$
t_3	Mark	Kobe signs some autographs after the game	$\{l_5: .9, l_6: .1\}$
t_4	Ben	Go to Chicago and watch NBA	$\{l_7: .7, l_8: .3\}$
t_5	Judy	Kobe 1st NBA game highlights from ESPN	$\{l_9: .2, l_{10}: .8\}$
t_6	Ricky	I'm here for NBA game	$\{l_{11}: .2, l_{12}: .8\}$

database. Similarly, automatic identification of geolocation [12, 18] tags a web page to a suitable region with a certain confidence, or a footprint [3] represented by a minimal bounding box.

In such applications, spatial keyword queries Q , consisting of a textual part (Q^t) and a spatial part (Q^s), are often natural and useful in retrieving information from such probabilistic spatial data, for example to provide *local search*. In such queries, users want to retrieve tuples that match their information needs (high text relevance, and high location proximity) and high confidence level (higher ranked).

Example 1. Consider a simple Twitter sample data shown in Table 1. Each tuple in this database has both textual data (Tweet column) and probabilistic spatial data (Location column), indicating possible places where the tweet was originated and the corresponding probabilities. For example, Mark produced t_3 from location l_5 or l_6 with probabilities 0.9 and 0.1 respectively. A spatial keyword query Q (“Kobe Bryant NBA game”, $MBR(\{l_3, l_5, l_7, l_8\})$) would return tweets which were originated in the given Minimal Bounding Box (MBR), in this case MBR containing $\{l_3, l_5, l_7, l_8\}$ and talks about NBA player Kobe’s game. Each probabilistic spatial tuple is represented by a Minimal Bounding Box (MBR) so that each possible location is inside the MBR. These tuples are indexed by an R-Tree, as shown in Fig. 1. A solid lined box represents the query region $Q^s = MBR(\{l_3, l_5, l_7, l_8\})$.

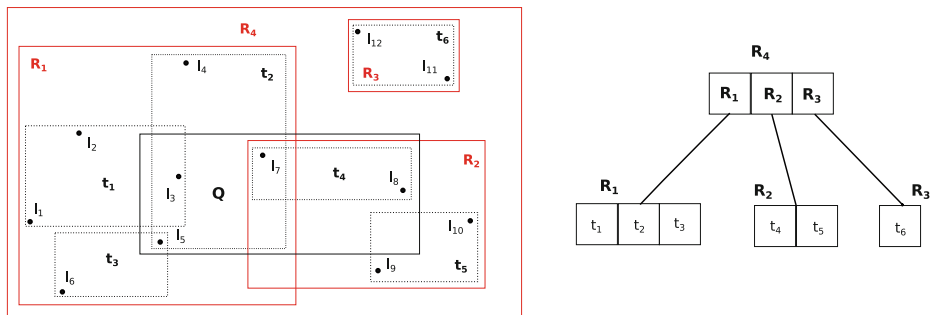


Fig. 1. R-Tree index for the Twitter sample in Table 1.

Example 1 demonstrates the type of probabilistic spatial data and the type of queries that will be studied in this paper. In order to answer such queries, we have to calculate two types of scores for each tuple (say t) (shown in Table 4), textual relevancy score $tScore(Q^t, t)$ (RawTF column in Table 4) and spatial confidence $cScore(Q^s, t)$ (the last column in Table 4) which computes the probability of tuple t being in the query region Q^s . For simplicity, we use RawTF scoring function in our running example to measure the textual similarity between query text (Q^t) and the tuple, although it could be any kind of scoring function such as OKAPI and TFIDF (see Sect. 5). To explain these scores, consider t_1 as an example. Since all four terms of Q^t are in the tweet, $tScore(Q^t, t_1) = 4$. Also, since t_1 's location falls in the query region Q^s when it takes value l_3 , the probability of t_1 satisfying Q^s is 0.2, i.e., $cScore(Q^s, t_1) = 0.2$. A spatial keyword query tries to report tuples that are both textually relevant (high $tScore$) and spatially reliable (high $cScore$).

In recent years, some studies have been conducted on indexing spatial text data, such as grid-based structures [3, 18], RTree-based structures [5, 10], answering particular types of spatial keyword queries including nearest neighbor [5], reverse nearest neighbor [10] and collective query [2]. However, none of them consider the uncertainty in the spatial text data which poses three challenges.

- *Semantics of keyword query on probabilistic spatial data:* The spatial attribute of a textual object is modeled as a distribution (or partial distribution) over multiple, mutually exclusive locations. The confidence that an object lies in the query range, $cScore(Q^s, t)$, or qualification probability, indicates reliability of an object for a query and should be carefully defined.
- *Tradeoff between textual similarity and spatial confidence:* Since the tuples can be ranked using two different scores, textual similarity and distance proximity, a suitable approach that integrates the two ranking factors is necessary in order to retrieve relevant tuples. The trade off between the two scores should be carefully defined.
- *Both efficiency and effectivity:* Although a spatial text index can speed up spatial keyword queries, queries over probabilistic data are typically time-consuming [6] and uncertainty in the data renders it difficult to achieve good efficiency. The tradeoff between the two scores can lead to bad precision and should be dealt with.

In this paper, we follow an interactive two-step framework for processing keyword queries on probabilistic spatial data. Our contributions are as follows.

- We introduce an Incremental Scoring Approach (ISA) based upon the IRTree index [5], which report tuples in the query region one by one in decreasing order of textual similarity (namely $tScore(Q^t, t)$).
- We propose a parameterized probabilistic ranking approach $PRank^c$ for evaluating top- (c, k) query. Based on possible world semantics, it ranks tuples by their confidence of being ranked in top- c answers (Top-C Confidence or TCC) and return k tuples with the highest TCC values.

Table 2. Notations

Notation	Description
D_p	Probabilistic spatial dataset
n	Size of dataset D_p
k	Number of tuples a query returns
c	Ranking parameter
$t_i.t$	Textual attribute of t_i
$t_i.s$	Probabilistic spatial attribute of t_i
$t_i.mbr$	Minimal bounding rectangle of t_i
Q^s	Spatial part of the query
Q^t	Textual part of the query
Q	Spatial keyword query
T_{Q^s}	Tuples that has non-zero $cScore(Q^s, t)$
T_{Q^t}	Tuples that has non-zero $tScore(Q^t, t)$
$tScore(Q^t, t_i)$	Textual score of Q^t and $t_i.t$
$cScore(Q^s, t_i)$	confidence of $t_i.s$ intersecting with Q^s
$TCC(t_i)$	Probability of t_i being part of top- c query result
$Pr(L_i, j)$	Probability that the size of $\{t_1 \dots t_i\}$ is j
$Pr(R_i, j)$	Probability that t_i is ranked j
$SPL(i, l)$	Probability that the size of $\{t_1 \dots t_i\}$ is in $[0, l]$

- We propose a heuristic optimization rule that helps the query terminate earlier.
- To demonstrate the efficiency of the proposed framework, we implement our framework and present an empirical study.

The rest of the paper is organized as follows. We formulate the problem of top- (c, k) query and give the general description of our framework of processing spatial text query in Sect. 2. In Sect. 3, we describe incremental scoring approach, which provides a mechanism to index tuples so that tuples with high textual relevance could be found. In Sect. 4, we introduce our algorithm to compute the top- c confidence of the tuples and optimization rules to improve the performance. We present an empirical study of our approach in Sect. 5. We summarize some related work in Sect. 6 and finally conclude the paper in Sect. 7.

2 Problem Statement

Let $D_p = \{t_1, t_2, \dots, t_n\}$ be a probabilistic spatial text dataset, where each tuple t_i has a textual attribute $t_i.t$ and probabilistic spatial attribute $t_i.s$ (a set of possible locations of t_i along with probabilities). A spatial keyword query Q , consisting of a set of keywords Q^t , a query region Q^s , and a number k , returns k tuples that are both textually relevant and spatially reliable. Notations used in this paper are listed in Table 2.

To answer a spatial keyword query, we need to first calculate two types of scores for all tuples and then rank them in terms of these two scores. It is natural to adopt a two-step framework as described in Table 3. In the first step,

Table 3. Two-step framework

Step 1	Scoring each tuple’s textual relevance $tScore(Q, t)$ and spatial confidence $cScore(Q, t)$
Step 2	Ranking tuples by combining above two factors

the textual relevance scores and the location scores are computed using various textual, spatial and hybrid indexes to speed up these two scoring processes, which could be done separately [19] or in combination [5]. In the second step, in order to report tuples that are both textually relevant and have spatial confidences, tradeoff must be made between the two factors. We examine two existing approaches to balance these two factors and discuss their drawbacks.

The simplest approach is Linear Combination (LC). It balances two factors using a parameter β and the final score is defined as follows.

$$LC(t, Q; \beta) = \beta \times tScore(Q^t, t) + (1 - \beta) \times cScore(Q^s, t) \quad (1)$$

When β is 1, it reduces to a traditional IR task and ignores spatial confidence. On the other extreme, when β is 0, it equals to ranking only based on spatial qualification. LC, even though straightforward, is semantically unreliable. For example, in Example 1 tuple t_5 should not be part of the query result as it has no chance of being within the query region ($cScore(Q^s, t_5) = 0$). However, it will be ranked higher than t_3 for large values of β even though t_3 is a better answer than t_5 . A similar problem will arise for high value of β . Choosing a reasonable value for parameter β to address this problem can be tricky.

The second approach is Confidence Threshold (CT). It ranks those tuples whose $cScore$ exceeds the predefined threshold Π . Ranking is done using their $tScore$. The final score is defined as follow.

$$CT(t, Q; \Pi) = \begin{cases} tScore(Q^t, t) & \text{if } cScore(Q^s, t) > \Pi \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Parameter Π controls the reliability of results. A large Π will prune most tuples even with high spatial confidence and will less more preference to the $tScore$, while a small Π will allow more spatial irrelevant tuples but very textually relevant tuples to become answers. The choice of a reasonable Π depends on the query Q . In particular, it is better to choose a large Π for a query Q if T_{Q^s} contains a large part of tuples with high qualification probability. A reasonable value of Π is vital to produce consistent good performance for any query.

Considering the drawbacks of the previous approaches, we choose top-(c, k) semantics to answer the uncertain spatial keyword queries. Considering the Possible World Semantics, a naive approach to evaluate a query over a probabilistic database D_p would be to first expand the database into a set of possible worlds $PW(D_p)$. The query would be executed over each possible world w in $PW(D_p)$ and then the set of results would be collapsed into a final result. We use

Table 4. Scores for sample data

ID	RawTF	cScore
t_1	4	0.2
t_2	3	0.5
t_3	2	0.9
t_4	1	1.0
t_5	3	0.0
t_6	2	0.0

Table 5. TCC based ranking

c	Tuple and TCC			
1	$(t_2, 0.4)$	$(t_3, 0.36)$	$(t_1, 0.2)$	$(t_4, 0.04)$
2	$(t_3, 0.81)$	$(t_2, 0.5)$	$(t_4, 0.45)$	$(t_1, 0.2)$
3	$(t_4, 0.91)$	$(t_3, 0.9)$	$(t_2, 0.5)$	$(t_1, 0.2)$
4	$(t_4, 1)$	$(t_3, 0.9)$	$(t_2, 0.5)$	$(t_1, 0.2)$

$IRRank(t_i, w)$ to denote t_i 's rank in the possible world w using the chosen textual similarity scoring function. The top- c confidence (TCC) of tuple t_i captures its probability of being in the top- c result based on possible world semantics.

Definition 1. For query Q and tuple t_i , top- c confidence of t_i , denoted by $TCC(t_i)$, is defined as the chance of t_i being in the top- c result, namely the sum of probabilities of possible worlds where t_i is in the top- c result. Formally,

$$TCC(t_i) = \sum_{w \in PW(D_p)} p(w) \times \xi(t_i, w) \quad (3)$$

where indicator $\xi(t_i, w)$ equals 1 when $IRRank(t_i, w) \leq c$, otherwise 0.

Proposition 1. Assume that tuple t_i is ranked at the v_i -th position using $tScore$, function $TCC(t_i)$ grows monotonically with value of c , and reaches its maximum value, namely $cScore(Q, t_i)$, at $c = v_i$.

A top- (c, k) query returns k tuples which have the highest probability of being in the top- c query results. Top- (c, k) queries consider both $tScore$ and $cScore$ in order to compute the top- c confidence. With a low $tScore$ value a tuple will not be in the top- c results in many possible worlds leading to a low top- c confidence. On the other hand, a low $cScore$ will reduce its probability of being in many possible worlds leading to a low top- c confidence.

In Example 1, four tuples $\{t_1, t_2, t_3, t_4\}$ will have a non-zero $tScore$. For a given value of c , these tuples will be ranked according to their TCC values. For now, let us assume $c = 1$. To compute $TCC(t_1)$, we need to find the probability with which t_1 will have the highest $tScore$. If we ignore the $cScore$, t_1 has the highest rank. However, with $(1 - cScore)$ probability, the tuple does not satisfy the spatial conditions. Thus, the top- c confidence of t_1 is its $cScore$, i.e. 0.2. The tuple with the next best $tScore$ (t_2) has a chance to be in the top- c results if t_1 doesn't exist and t_2 exists, i.e. $TCC(t_2) = (1 - cScore(Q, t_1)) * cScore(Q, t_2) = 0.4$. Table 5 shows the top- c confidence values for these tuples for different values of c . When $c = 1$ and $k = 1$, the query result is $\{t_2\}$.

For the step 1 of our framework, we propose a IRTree-based incremental scoring approach (ISA). ISA exploits the IRTree structure to index the tuples so that textually relevant tuples could be iterated over in the decreasing order of

Table 6. Inverted index of each node

Terms	Posting list			
	R_4	R_1	R_2	R_3
Kobe	$\{(R_1, 1), (R_2, 1)\}$	$\{(t_1, 1), (t_2, 1), (t_3, 1)\}$	$\{(t_5, 1)\}$	Φ
Bryant	$\{(R_1, 1), (R_2, 1)\}$	$\{(t_1, 1)\}$	Φ	Φ
NBA	$\{(R_1, 1), (R_2, 1), (R_3, 1)\}$	$\{(t_1, 1), (t_2, 1)\}$	$\{(t_4, 1), (t_5, 1)\}$	$\{(t_6, 1)\}$
Game	$\{(R_1, 1), (R_2, 1), (R_3, 1)\}$	$\{(t_1, 1), (t_2, 1), (t_3, 1)\}$	$\{(t_5, 1)\}$	$\{(t_6, 1)\}$

their *tScore*. Then, we propose a parameterized probabilistic ranking approach, named *PRank^c*. It (1) uses ISA to iterate over the tuples sorted in decreasing order of *tScore*, and (2) estimates the top-*c* confidence of the tuples using a statistical model (see Sect. 5). In the following sections, we describe these components, ISA and *PRank^c*, and propose optimizations to improve the performance (Table 6).

3 Incremental Scoring Approach

As the first step of the framework, we index the tuples using an IRTree hybrid index similar to the one proposed in [5]. Each node is augmented with a pseudo document. The weight of each term *t* in the pseudo document is the maximum weight of the term in the documents contained in its subtree. The concept of pseudo document enables us to estimate an upper bound for text relevancy between query text Q^t and any document contained in the subtree rooted at a specific node *N*. Thus, we have the following inequality:

$$\forall e \in N, tScore(Q^t, N) \geq tScore(Q^t, e) \tag{4}$$

See Theorem 3.1 in [5] for details.

For a given query, incremental scoring algorithm visits each node *N* (starting from the root) that shares locations with the query region in decreasing order of maximum possible textual similarity $tScore(Q^t, N)$, and prune non-intersecting nodes. The ISA algorithm is described below.

In Example 1, ISA starts with pushing root R_4 onto a priority queue *PriQue* and then executes steps as shown in Table 7. In the scoring process, all tuples that have overlapping locations with the query region would be visited, in decreasing order of *tScore*. The tuples will be reported in the order t_1, t_2, t_3 , and t_4 according to their *tScore* values. *ISA* ignores nodes that do not intersect with query region (for example R_3). Similarly, tuples that do not share any location with the query region are not reported either (for example t_5).

Using ISA, the query results can be iterated over in the decreasing order of their *tScore*. This is used later on by the Step 2 of the framework to finally calculate the TCC value of the tuples.

Algorithm 1. ISA(Q, root)

```

Input: Query Q and the IRTree root
Output: Tuples with non-zero cScore in decreasing order of tScore
Initialize a max priority queue PriQue
Push entry (root, tScore( $Q^t$ , root)) in PriQue
while PriQue not empty do
  entry  $\leftarrow$  PriQue.pop() //pop max tScore entry
  if entry is an object then
    Report entry
  else
    if entry is a leaf node then
      for each object in entry do
        if object shares points in  $Q^t$  then
          Push (object, tScore( $Q^t$ , object)) in PriQue
        end if
      end for
    else
      for each node in entry do
        if  $node.mbr \subset Q^s = \Phi$  then
          Push (node, tScore( $Q^t$ , node)) in PriQue
        end if
      end for
    end if
  end if
end while

```

Table 7. ISA processing steps

Step	Pop	Operation	PriQue
1	R_4	Push R_1, R_2	$(R_1, 4), (R_2, 3)$
2	R_1	Push t_1, t_2, t_3	$(t_1, 4), (t_2, 3), (R_2, 3), (t_3, 2)$
3	t_1	Report t_1	$(t_2, 3), (R_2, 3), (t_3, 2)$
4	t_2	Report t_2	$(R_2, 3), (t_3, 2)$
5	R_2	Push t_4, t_5	$(t_5, 3), (t_3, 2), (t_4, 1)$
6	t_5	Discard t_5	$(t_3, 2), (t_4, 1)$
7	t_3	Report t_3	$(t_4, 1)$
8	t_4	Report t_4	Φ

4 Probabilistic Ranking

For a query Q and dataset D_p , ISA returns results in the form (*tupleid*, *tScore*, *cScore*). These results are accessed from an iterator (*DocSeq*) in decreasing order of *tScore*. In the following sections, we describe a *PRank^c* algorithm for effectively ranking these relevant tuples based on their top-*c* confidences.

4.1 Basic Algorithm

In Example 1, *DocSeq* would return $\{(t_1, 4, 0.4), (t_2, 3, 0.5)\}$ as the first two tuples as their have the highest *tScore*. Since these tuples are probabilistic in nature, in a possible world, some of them may not exist. Lets $Pr(L_i, j)$ represent the probability that out of the i tuples from *DocSeq*, the actual size of the output is j . $Pr(L_i, j)$ can be computed as follows:

$$Pr(L_i, j) = \begin{cases} Pr(L_{i-1}, j - 1) \times p_i + Pr(L_{i-1}, j) \times (1 - p_i) & \text{if } i > 0, j \in [1, i] \\ \prod_{k \in [1, i]} (1 - p_k) & \text{if } i > 0, j = 0 \\ 1 & \text{if } i = 0, j = 0 \\ 0 & \text{if } j > i \end{cases} \quad (5)$$

Similarly, the rank of t_i is a random variable R_i ranging from 1 to i . For example, among the first two tuples, t_2 can have rank 2 when both t_1 and t_2 exist, while its rank would be 1 if t_1 does not exist. In general, the event of t_i being ranked at the j^{th} position (R_i, j) is equivalent to the event that the size of the preceding i tuples in *DocSeq* is $j - 1$, i.e.,

$$Pr(R_i, j) = Pr(L_{i-1}, j - 1) \times p_i \quad (6)$$

Using Proposition 1, top- c confidence (TCC) of a tuple can be computed as following:

$$TCC(t_i) = \sum_{j \in [1, c]} Pr(R_i, j) \quad (7)$$

Notice that $tScore(Q, t_{i-1}) > tScore(Q, t_i)$ as *DocSeq* returns tuples in the decreasing order of *tScore*.

PRank^c algorithm starts with $i = 0$ and processes each tuple from *DocSeq* sequentially. For tuple t_i , it calculates $Pr(L_i, j)$ and $Pr(R_i, j)$, and then its top- c confidence. After processing all tuples, k tuples with the highest TCC are reported. Algorithm 2 shows *PRank^c* in detail.

Algorithm 2. *PRank^c* Algorithm

Input: *DocSeq*, a sequence of tuples in decreasing order of their IR scores,

Output: *PriQue*, a priority queue of size k

Initialize $Pr(L_0, 0) = 1$ and $Pr(L_0, j) = 0$ for $j \in [1, c]$

while *DocSeq* not empty **do**

$t_i \leftarrow \text{DocSeq.pop}()$ //Pop the i^{th} doc

Calculate probabilities, $Pr(L_i, j), j \in [0, c]$

Calculate probabilities, $Pr(R_i, j), j \in [1, c]$

Calculate $TCC(t_i) = \sum_{j \in [1, c]} Pr(R_i, j)$

Push $(t_i, TCC(t_i))$ in priority queue *PriQue*

end while

return *PriQue*

For each of n tuples in *DocSeq*, $2c + 1$ values need to be calculated while sorting within priority queue requires $n \log(k)$ time, so time complexity is $O(n(c + \log(k)))$. We need $O(c)$ space for storing $Pr(L_i, j)$ for t_{i+1} in next loop and $O(k)$ space for the resulting PriQue, requiring in total $O(c + k)$ space.

4.2 Early Termination of *PRank^c*

PRank^c algorithm uses ISA to compute top- c confidence for each tuple that has a non-zero *tScore* and a non-zero *cScore*. In this subsection, we propose a mechanism to identify when processing further tuples is not required. By using this optimization, we do not have to process all tuples which will improve efficiency.

For a fixed value l , let $SPL(i, l) = \sum_{j \in [0, l]} Pr(L_i, j)$. Then, the following theorem gives us an upper bound on TCC in terms of SPL.

Theorem 1. *For any tuple t_i , its top- c confidence is upper bounded by $SPL(i - 1, c - 1)$.*

Proof.

$$\begin{aligned}
 TCC(t_{i+1}) &= \sum_{j=1}^c Pr(R_{i+1}, j) \\
 &= \sum_{j=1}^c Pr(L_i, j - 1) \times p_i \\
 &\leq \sum_{j=1}^c Pr(L_i, j - 1) \\
 &= \sum_{j=0}^{c-1} Pr(L_i, j) \\
 &= SPL(i, c - 1)
 \end{aligned}$$

□

Using Theorem 1, we can optimize our *PRank^c* algorithm so that it terminates early. Algorithm 3 shows the updated algorithm. For a top- (c, k) query, we keep track of the k^{th} largest TCC value and store it in variable τ . From the above analysis, we know that if $SPL(i, c - 1) < \tau$, all tuples after t_{i+1} will not be part of the top- (c, k) answers, so *PRank^c* algorithm can stop.

5 Experiments

To evaluate the efficiency of the proposed framework, we implement our solutions using Python. We now present the results of our experiments. The experiments were run on an Intel Xeon 2.4 GHz machine with 12 GB RAM and a 7200 RPM disk with a transfer rate of 3 Gb/s, running Linux.

Algorithm 3. $PRank^c$ -OPT Algorithm

Input: DocSeq, a sequence of tuples in decreasing order of their IR scores,
Output: PriQue, a priority queue of size k Initialize $Pr(L_0, 0) = 1$ and $Pr(L_0, j) = 0$ for $j \in [1, c]$ **while** DocSeq not empty **do** $t_i \leftarrow \text{DocSeq.pop()}$ //Pop the i^{th} docCalculate probabilities, $Pr(L_i, j), j \in [0, c]$ Calculate probabilities, $Pr(R_i, j), j \in [1, c]$ Calculate $TCC(t_i) = \sum_{j \in [1, c]} Pr(R_i, j)$ Push $(t_i, TCC(t_i))$ in priority queue PriQueCalculate $SPL(i - 1, c - 1)$ and threshold τ Stop, if $SPL(i - 1, c - 1)$ is less than τ **end while**

return PriQue

Dataset: We collected Twitter data using the Twitter API. The data consists of *tweets* which are geotagged. For each *tweet*, the location of the two previous and two future tweets of the user are collected as well. These locations are used to form a probability distribution function, which is assigned as the MBR for the *tweet*. For our experiments, we populate our database with 200,000 *tweets*.

Scoring Function: For the experiments, we use LogTFIDF (Eq. 9) as the scoring function to compute the textual relevance. However, other scoring functions such as OKAPI would work too.

$$RawTF(Q^t, d) = \sum_{t \in Q^t} TF(t, d) \times TF(t, Q^t) \quad (8)$$

$$LogTFIDF(Q^t, d) = \sum_{t \in Q^t} TF(t, d) \times \log\left(\frac{\#Docs}{\#Docs(t) + 1}\right) \times TF(t, Q^t) \quad (9)$$

Naive approach: As a base case for comparison, we compare the performance of our solutions with a naive approach described as follows. In the naive approach, for each tuple in the database, $tScore$ and $cScore$ are computed. The tuples are sorted using the $tScore$ values. After this, we run the $PRank^c$ algorithm to compute the top- c probabilities of these tuples and then returns the top- (c, k) results.

5.1 Results

We now present the results of our experiments. To reduce the error, all experiments were conducted three times. In each experiment, time is reported in seconds. In the graphs, “Basic” represents the naive approach. “ $PRank^c$ ” represents our solution which uses ISA and $PRank^c$ algorithms. “ $PRank^cOPT$ ” represents our solution with uses ISA, $PRank^c$ and the proposed optimization.

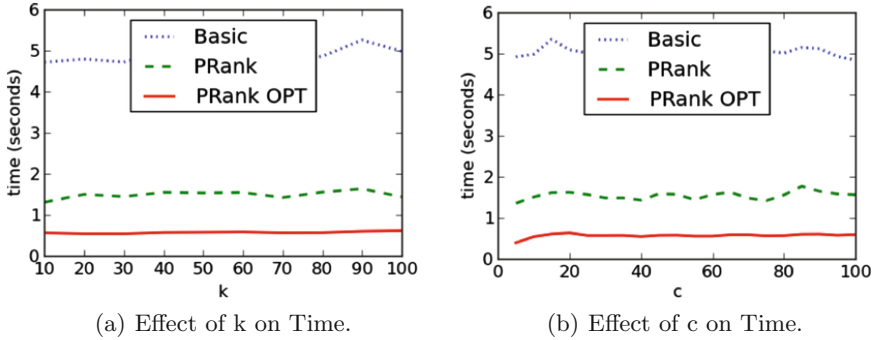


Fig. 2. Effect of query parameters (c, k)

To understand the efficiency of our solutions in terms of query processing time, we conduct two experiments. The experiments evaluate the performance of our solutions as the value of c and k changes. In the first experiment, we keep $c = k$, and change their values together. Figure 2(a) shows the results. Our solution using the ISA and $PRank^c$ algorithms perform multiple fold better than the base algorithm. The proposed optimization in $PRank^c$ improves the performance further by almost twice. Similar results are shown when c is varied without changing k . Figure 2(b) shows the results. In this experiment, we keep k at a constant value of 50 and vary the value of c . In this experiment as well, our optimized $PRank^c$ algorithm performs multiple fold better.

To understand the effect of optimization further, we conduct another set of experiments in which we vary the amount of qualified tuples, i.e., tuples which have non-zero $tScore$ and non-zero $cScore$. This is done by increasing the query range. Figure 3(a) shows the effect of increasing the qualified tuples on the execution time. As shown in the figure, our $PRank^c$ algorithm performs better than the base algorithm. Our optimized $PRank^c$ algorithm is not affected significantly by the increase in the qualified tuples. This is because of early termination. Even

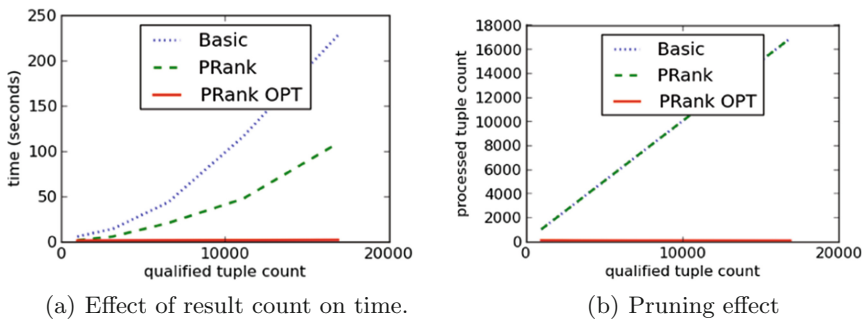


Fig. 3. Effect of optimization

though the number of qualified tuples is increased, our optimization keeps track of an upper bound on the top- c confidence, and once the upper bound is reached, the algorithm stops according to Theorem 1. To understand it further, we conduct another experiment to find how many qualified tuples are processed by the algorithms before returning the results. Figure 3(b) shows the results. Obviously, the tuples processed by the base algorithm and the $PRank^c$ algorithm are the same as they both process all tuples which has non-zero $tScore$ and non-zero $cScore$. However, our optimized $PRank^c$ algorithm terminates much earlier, leading to reduced processing time as well.

Overall, we observe that our proposed solutions perform much better than the base algorithm. Our optimized $PRank^c$ leads to very early termination improving the performance by multiple folds.

6 Related Work

Related work in this domain can be mainly categorized in three groups, geotagging documents, keyword search on spatial databases, and query processing in probabilistic databases. In this section, we present some work in these areas.

Geotagging documents. With the popularity of location based services, there is an increasing demand to compute geotags for web content. Several studies have been done to infer geotags from the textual information of the documents. McCurley et al. [12], Amitay et al. [1] and Markowetz et al. [11] investigate how to recognize geographical contexts, such as postal codes, addresses, telephone numbers and proper names of geographic entities, in the web using heuristic methods or simple scoring algorithm and relate a web page to a set of geographical locations. Zhou et al. [19] further map these geographical locations to unique coordinates and represent a web page location as an MBR and also develop a hybrid index where an R^* -tree and inverted file index are loosely coupled to speed up the spatial and textual retrieval process separately. With the rapid development of statistical machine learning, the geographical scope of a web page or geographical entities mentioned in a document tend to be recognized based on statistical approaches [15, 18] which typically associate multiple possible locations with a web page. These possible locations are represented as a probability distribution.

Keyword query on spatial databases. With the popularity of low-cost GPS chips and electronic maps, geo-tagged documents have become prevalent on the Internet. Various types of spatial keyword queries have been studied. In [5], Cong et al. propose an IR-Tree index where an inverted file index and an R-Tree are closely coupled for retrieving the most relevant web objects based on a scoring function that takes into account both text relevancy and location proximity. In [10], Lu et al. design a branch-and-bound algorithm to answer reverse spatial textual k nearest neighbor queries based on an intersection-union R-Tree index. In [2], Cao et al. study collective spatial keyword queries to retrieve a group of spatial objects that have the lowest interobject distance and cover all query keywords.

Probabilistic database. Much work has been done to provide database support for managing probabilistic data [6, 16]. Probabilistic queries are typically evaluated based on possible world semantics [6]. Due to the exponential number of possible worlds, and probabilistic query output, threshold queries [8, 13] and top- k queries are more relevant for information retrieval from probabilistic databases. Soliman et al. [17] investigate two types of uncertain ranking criteria, U-Topk and U-kRanks, to find most probable top- k tuple vectors as a whole and most probable tuples appearing at top- k ranks. Lian and Chen [9] developed the spatial and probabilistic pruning techniques for U-kRanks queries. Hua et al. [8] propose a probabilistic threshold approach that rank tuples by their confidence of being qualified results. Probabilistic spatial queries have also attracted much attention [4, 14], though these works do not consider textual retrieval.

7 Conclusions and Future Work

In this paper, we introduced the problem of spatial keyword queries on probabilistic spatial databases. We proposed a general framework to process these queries. First, we developed an incremental search algorithm that leverages the IR-Tree to report qualified tuples one by one in the decreasing order of textual similarity. Second, we proposed a parameterized probabilistic ranking algorithm that computes the top- c confidence of the qualified tuples in a linear time. An effective early stop method is also provided to optimize the algorithm. We evaluate our approach on a real dataset collected from Twitter. Experiments show that our solutions are efficient. An important part of top- (c, k) queries is to pick the parameter c so that the precision of the query results is high. We leave that for the future work.

Acknowledgements. The work in this paper was supported by National Science Foundation grants IIS-1017990 and IIS-09168724.

References

1. Amitay, E., Har'El, N., Sivan, R., Soffer, A.: Web-a-where: geotagging web content. In: Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (2004)
2. Cao, X., Cong, G., Jensen, C.S., Ooi, B.C.: Collective spatial keyword querying. In: Proceedings of ACM Special Interest Group on Management of Data (SIGMOD) (2011)
3. Chen, Y.-Y., Suel, T., Markowetz, A.: Efficient query processing in geographic web search engines. In: Proceedings of ACM Special Interest Group on Management of Data (SIGMOD) (2006)
4. Cheng, R., Chen, L., Chen, J., Xie, X.: Evaluating probability threshold k-nearest-neighbor queries over uncertain data. In: Proceedings of the International Conference on Extending Database Technology (EDBT) (2009)

5. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. In: Proceedings of the International Conference on Very Large Data Bases (VLDB) (2009)
6. Dalvi, N., Suciu, D.: Efficient query evaluation on probabilistic databases. In: Proceedings of the International Conference on Very Large Data Bases (VLDB) (2007)
7. De Felipe, I., Hristidis, V., Rish, N.: Keyword search on spatial databases. In: Proceedings of the International Conference on Data Engineering (ICDE) (2008)
8. Hua, M., Pei, J., Zhang, W., Lin, X.: Ranking queries on uncertain data: a probabilistic threshold approach. In: Proceedings of ACM Special Interest Group on Management Of Data (SIGMOD) (2008)
9. Lian, X., Chen, L.: Probabilistic ranked queries in uncertain databases. In: Proceedings of the International Conference on Extending Database Technology (EDBT) (2008)
10. Lu, J., Lu, Y., Cong, G.: Reverse spatial and textual k nearest neighbor search. In: Proceedings of ACM Special Interest Group on Management Of Data (SIGMOD) (2011)
11. Markowetz, A., Chen, Y.Y., Suel, T.: Design and implementation of a geographic search engine. In: International Workshop on the Web and Databases (WebDB) (2005)
12. McCurley, K.S.: Geospatial mapping and navigation of the web. In: Proceedings of the International Conference on World Wide Web (WWW) (2001)
13. Qi, Y., Jain, R., Singh, S., Prabhakar, S.: Threshold query optimization for uncertain data. In: Proceedings of ACM Special Interest Group on Management Of Data (SIGMOD) (2010)
14. Qi, Y., Singh, S., Shah, R., Prabhakar, S.: Indexing probabilistic nearest-neighbor threshold queries. In: Workshop on Management of Uncertain Data (2008)
15. Sarawagi, S.: Information extraction. *Found. Trends Databases* **1**(3), 261–377 (2008)
16. Singh, S., Mayfield, C., Shah, R., Prabhakar, S., Hambrusch, S.E., Neville, J., Cheng, R.: Database support for probabilistic attributes and tuples. In: Proceedings of the International Conference on Data Engineering (ICDE) (2008)
17. Soliman, M.A., Ilyas, I.F., Chang, K.C.-C.: Top-k query processing in uncertain databases. In: Proceedings of the International Conference on Data Engineering (ICDE), April 2007
18. Wing, B.P., Baldridge, J.: Simple supervised document geolocation with geodesic grids. In: Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (2011)
19. Zhou, Y., Xie, X., Wang, C., Gong, Y., Ma, W.-Y.: Hybrid index structures for location-based web search. In: ACM International Conference on Information and Knowledge Management (2005)

Towards Mobile Sensor-Aware Crowdsourcing: Architecture, Opportunities and Challenges

Jiyin He¹, Kai Kunze², Christoph Lofi³, Sanjay K. Madria⁴(✉),
and Stephan Sigg⁵

¹ Centrum Wiskunde and Informatica, Amsterdam, The Netherlands
jiyinhe@gmail.com

² Osaka Prefecture University, Osaka, Japan
kai.kunze@gmail.com

³ National Institute of Informatics, Tokyo, Japan
lofi@nii.ac.jp

⁴ Missouri University of Science and Technology, Rolla, USA
madrias@mst.edu

⁵ Georg-August University Goettingen, Goettingen, Germany
ssigg@gwdg.de

Abstract. The recent success of general purpose crowdsourcing platforms like Amazon Mechanical Turk paved the way for a plethora of crowd-enabled applications and workflows. However, the variety of tasks which can be approached via such crowdsourcing platforms is limited by constraints of the web-based interface. In this paper, we propose mobile user interface clients. Switching to mobile clients has the potential to radically change the way crowdsourcing is performed, and allows for a new breed of crowdsourcing tasks. Here, especially the ability to tap into the wealth of precision sensors embedded in modern mobile hardware is a game changer. In this paper, we will discuss opportunities and challenges resulting from such a platform, and discuss a reference architecture.

Keywords: Mobile platforms · Sensor-enabled crowdsourcing · Location-aware crowdsourcing

1 Introduction

Crowdsourcing has become a popular approach to many problems that cannot be easily addressed by automated methods and algorithms, or problems that explicitly require significant amount of human intelligence or human feedback. Crowdsourcing can often be found in knowledge processing tasks such as data or media classification [8], data acquisition tasks such as data completion [6] or information extraction [16], as well as in providing training data for machine-learning-based approaches [20]. Furthermore, crowdsourcing has proven to be useful to the research community for performing large-scale user studies for evaluating new prototype implementations [11], or performing surveys with a

large and diverse number of participants for investigating general human behavior or preferences [1]. Instead of laboriously growing own custom crowdsourcing platforms, these tasks mostly rely on general purpose crowdsourcing platforms such as Amazon Mechanical Turk, CrowdFlower, or ClickWorker. These platforms allow a complex task to be executed by dividing it into many smaller and simpler sub-tasks, i.e., HITs (Human Intelligence Tasks) – the smallest unit of crowdsourcable work, which are the distributed to a human worker pool. Workers are recruited and retained with payment. Hence, in theory such platforms can be used to perform any dividable tasks that require human intelligence. However, most of these services only offer a web-based interface for workers, and therefore tasks are limited to those that can be displayed and solved within a web browser.

In this paper, we propose an alternative architecture for a general-purpose crowd-sourcing platform based on mobile as well as PC devices to interact with the worker pool, referred to as “hybrid crowdsourcing platform”. This will increase not only the ease of use and acceptance of workers in an ever more mobile society, but also the utility and the range of possible crowdsourcing tasks for research applications as well as practical application. In particular, the access to GPS locations and mobile sensors will allow novel crowd-based applications that have not been possible before. Our contributions in this paper are as follows:

- We motivate and discuss need and benefits of mobile sensor-enabled crowdsourcing platforms.
- We highlight use cases of our platform, especially in the area of locality-sensitive services and ubiquitous computing.
- We present the design space and the generic architecture of such a platform, and discuss the impact of certain decisions on the system features and usability.

2 Background

Crowdsourcing can lead to significant cost savings [9,15,25], product quality improvements [2] and acceleration of time to market [3,4].

However, crowdsourcing also has the potential to mitigate regional differences in the distribution of labor and human resources. Therefore, most previous work on mobile crowdsourcing platforms focused on societal aspects of crowdsourcing [5,7,18]. These approaches have been tailored for developing countries as an alternative source of labor and income. In developing countries, the spread of personal computers and wired internet connectivity is low. However, still many may have access to mobile phones or even mobile internet service. Therefore, the core challenge discussed in these works is how crowdsourcing can be adapted to the low-end hardware commonly available in developing countries, and how gaps in internet connectivity could be covered using SMS or alternative messaging methods.

In contrast, mobile crowdsourcing as discussed in this paper especially focused on exploiting the capabilities of modern, powerful mobile hardware to offer new functionality to crowdsourcing services. Especially the ability to tap into the user's geo-location or access to high-quality sensor data allows for completely new applications.

3 General Design

We envision a crowdsourcing platform that can be used in a stationary as well as mobile setting. The various instances in the private devices of users are interconnected through a server in the cloud that takes care of the aggregation of responses, ranking, evaluation and source-selection for a given request. For this purpose, the server stores locations, end user profiles and source profiles among other information (cf. Fig. 1).

Manual labels and judgements can be harvested as well as sensor data on mobile devices. The requester is likewise part of the crowd as any user or service may issue a query for input of users, services or sensors.

By the combination of mobile, pervasive and crowdsourcing concepts, we will be able to provide crowdsourcing for the masses: A more democratic crowdsourcing usage pattern in which everybody can be crowdsourced or equally state own queries. Mobile crowdsourcing will be seamlessly integrated into daily life with constantly up-to-date, personalised queries that can be completed anytime,

Hybrid crowdsourcing

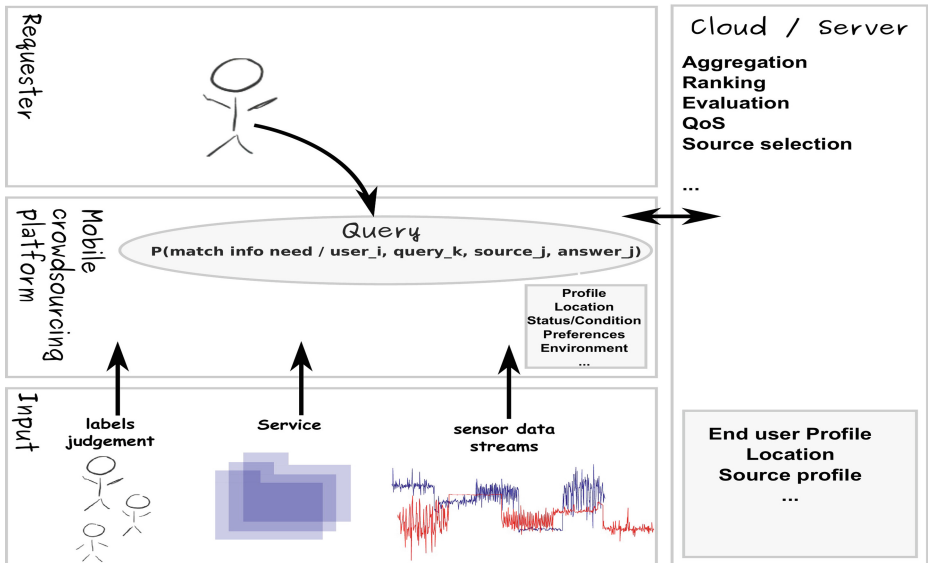


Fig. 1. A proposal of a mobile crowdsourcing platform

anywhere. Instead of playing pointless mobile games to bridge waiting times, people can instead solve interesting queries and even earn money by completing these tasks. Through the integration of context sources in addition to human sources for content provisioning, queries can be highly personalized (e.g. location, environment, condition [19]) and in addition be automatically evaluated for their quality (e.g. fatigue [14]). Such a platform can be exploited to collect huge amounts of labelled sensor data (by asking users to perform certain tasks while being recorded by sensors on the mobile device [23]) from a tightly controlled target population. In addition, it might change the nature of crowdsourcing by empowering ordinary people to set up simple queries that might even reach into their real world (ask people to buy/bring something somewhere). Furthermore, a such a platform might replace traditional data-bases in applications that rely on data which is changing at a high pace. For instance, imagine a dating service, in which a query for a potential partner is not stated to a database of registered users but instead towards the crowd.

4 Opportunities and Challenges

State-of-the-art crowdsourcing platforms are implemented through web-based services by international players such as Amazon. These platforms require explicit input and reach a maximally diverse population of possible content providers regardless of their location, gender, age, condition or further preferences. However, the result of a request is typically of medium or low quality and requires significant effort to filter out meaningful and quality responses [10]. The integration of crowdsourcing principles with mobile and Pervasive Computing has the potential to disruptively extend the possibilities underlying current crowdsourcing towards, among others, new applications, new classes of data and new possibilities to automatically evaluate quality of responses. We envision a platform with access to implicit information on, for instance, location, condition or further preferences that could restrict a given query to the most intended audience and also utilise sensor information (e.g. fatigue, crowd, loudness level) during the completion of a query in order to automatically estimate the quality of a response. In addition, with sufficient data at hand, prediction techniques might be applied in order to further boost the confidence on a result reached [22]. Figure 2 illustrates this concept.

Expected advantages of a hybrid mobile sensor-aware crowdsourcing paradigm include: (1) improved task performance and efficiency; (2) enabling new crowdsourcing process; and (3) enabling new types of applications. Below, we discuss these aspects, as well as the challenges involved in realising the proposed platform.

4.1 Improved Task Performance and Efficiency

Improved personalisation of request allocation and response aggregation. Hybrid mobile sensor-aware crowdsourcing would enable personalised

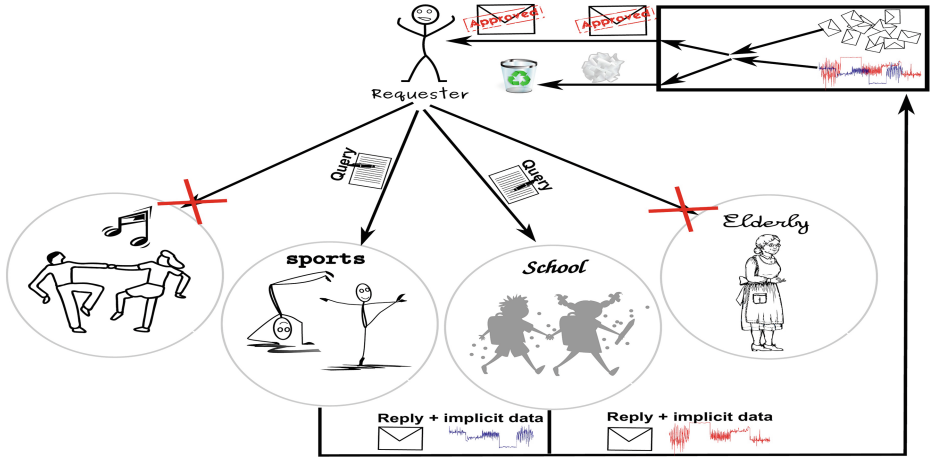


Fig. 2. Concept of mobile crowdsourcing

requests filtered by preferences set on mobile devices as well as by dynamic contextual parameters such as location, situation or condition.

Specifically, by maintaining worker profiles including a history of the tasks they have participated, their task performance, as well as the sensor inputs accompanied with this profile, e.g., their location and environment, the system can learn to predict the expertise of the worker, and under which conditions a task may be suitable for that particular worker. For instance, a worker may be able to accomplish a translation task with high quality in the morning when transiting from home to work, while this performance may decrease in the evening when he/she transits from work to home since she is exhausted after a long day already. Using such information, the system can (1) selectively recommend tasks to target workers, and (2) selectively return or aggregate worker response to the requester.

Crowdsourcing spontaneous feedback. With a mobile-based platform, both requesters and workers will have less constraints in terms of working locations. This may greatly reduce the time from stating a query to the reception of responses. As a result, responses can be very up-to-date and may include real-time assistance, for instance, in searching/recommendation for point-of-interest locations/navigation or spontaneous translation of foreign sentences (e.g. while ordering a menu at a restaurant).

New quality control mechanisms. With sensor data available alongside user input, this data may be utilised to estimate the quality of the provided input. For instance, by analysing the eye-gaze-movement, the platform can estimate fatigue or, reasoning from the loudness level [21] or amount of other people around [24], which can be used to judge whether the user is impaired in answering questions that require considerable concentration.

Information about situational impacts on cognitive performance. By utilising contextual information, a requester can gain knowledge about the performance of users in various environmental conditions. For instance, by stating a request to several groups of users in various contextual situations, the requester may learn about impacts on cognitive performance. Similarly, by controlling also the situational impacts for a series of queries to several sets of users, the requester can exclude side-effects on the result of a query.

4.2 New Crowdsourcing Processes

Crowdsourcing for the masses. A crowdsourcing platform on a mobile device, available anytime and anywhere at the convenience of users will change the principle nature of crowdsourcing. Constantly updated, up-to-date and personalised queries can be completed on-demand, interrupted and continued seamlessly. Another aspect is that mobile-based crowdsourcing mitigates hierarchies. Requester and source fall together to the same person as everybody is in the position to state a query. Consequently, quantity of queries will increase while their complexity will fall.

Weakening the strong correlation between labour and human resources. There is a strong relation between the physical location of labour and human resources. While crowdsourcing in general is capable of weakening this correlation, mobile crowdsourcing will further foster this development. In particular, since queries can be more personalised, companies are capable of stating more complex queries also for well-educated workers. This will open new possibilities for workers to offer their workforce without the necessity to relocate.

Participatory Sensing. The envisaged crowdsourcing platform provides access not only to manual input provided by users completing tasks, but also to sensors attached to the mobile platform (Gyroscope, Camera, GPS, Magnetometer, etc.) [17]. This might enable, for instance, quick requests for survey purposes even without manual user intervention. Devices and services might extend their contextual perception by harvesting (via automatically answered queries) for sensor information from devices in proximity. Similarly, a mobile crowdsourcing platform may be utilised to acquire labelled sensor data by requesting users to perform specific actions which are then recorded. For example, researchers can survey the relation between exercises and people's ability of solving math problems by requesting workers to solve a math problem before and after performing certain exercises. This process can be monitored and measures can be taken from both user input and sensor input from the mobile devices. This type of tasks are not feasible with traditional Web-based crowdsourcing platforms.

4.3 New Applications

Mobile crowdsourcing enables new applications for crowdsourcing. For instance, crowdsourcing can replace a database when sensor-based or non-time critical manual feedback is required. There are new challenges introduced by this

paradigm as data might then fluctuate in quality and quantity. In addition, crowdsourcing may partly leave the virtual space through a mobile platform. We envision, for instance, an event-hosting company that crowdsources actual manpower on demand. Also, crowdsourcing for educational purposes may serve the need of companies completing actual business-related tasks as well as the need of learners. For instance, a company active in language translations may provide users with text to be translated and later, after collection of all responses, with the corrected aggregated results for educational purposes.

Crowdsourcing as an anonymised customer information system. Mobile crowdsourcing can lower the burden and improve security and privacy in customer information systems. Instead of collecting and maintaining customer-related information for personalised interaction and product design, companies can reach a desired sub-set of customers on demand through mobile crowdsourcing platforms. This will significantly reduce cost and release companies from the burden to maintain huge databases of privacy-critical customer-related information.

Enabling technology for smart cities. A city is defined as smart when investments fuel sustainable economic growth in the respective aspects ‘economy’, ‘mobility’, ‘environment’, ‘people’, ‘living’ and ‘governance’ [12,13]. A hybrid crowdsourcing platform connects people, government, industry and the environment as all can state queries or provide input to requests stated. Mobile crowdsourcing can therefore serve as an interaction principle in such environments and constitute the backbone of a smart city, interconnecting all major entities.

Mobile crowdsourcing for energy management and smart buildings. Mobile crowdsourcing platform integrates environmental sensors and services. Humans and services acquire maintenance information from infrastructure and surrounding sensors via queries limited by proximity or belonging to a specific entity (building, room, etc.). In addition, services can serve as actuators, completing queries designed to control smart buildings and automation. In particular, the controlled entity might change relative to the location of the requester.

Mobile crowdsourcing as a chance for ubiquitous computing research. Currently, there is a trend to ubiquitous and smart computing, which especially includes wearable computing devices which transparently integrate into a user’s life. Often, these devices need to detect the user’s current activity and context in order to offer an appropriate service. However, detecting context and activity is an ongoing research challenge, and many approaches heavily rely on machine-learned models for this task. These, however, require exhaustive training data of users in clearly defined contexts performing a given activity in order to train those models. This training data is very hard to obtain efficiently and with the high diversity required for creating rich detection models (for example, sensor data of people riding a bicycle, or cooking in the kitchen, etc.). Therefore, mobile crowdsourcing can provide a significant boost to this line of research by allowing to acquire such training data quickly and cheaply.

4.4 Challenges

High performance data processing and analysis mechanism. With a mobile sensor-enabled crowdsourcing platform, we need to be able to process the vast amounts continuously generated explicit user inputs (requests and responses) as well as implicit sensor inputs in real time, e.g., in order to realise the above mentioned personalised request allocation and response aggregation. This requires high performance computational power as well as sophisticated data mining and machine learning algorithms that can scale to this type of data and give spontaneous responses. Further, sensor data as well as user inputs may be noisy. It is non-trivial to extract meaningful features from the raw sensor data as input for machine learning algorithms, or to derive human interpretable results.

Limitations of mobile devices. While mobile devices provide great flexibility for people to perform tasks, there are also limitations. These include: the small screen, the limitation of battery life, and the limited types of interactions allowed. For instance, it is less convenient for people to type long sentences in a mobile device compared to that on a PC. With these limitations in mind, dedicated user experience studies need to be carried out while designing and implementing mobile based HITs.

Data security and privacy issues. The proposed platform involves collecting data such as a user's location, activities, as well as other personal information measured by the sensors. A major concern is therefore data security and privacy issues. These personal information and mobile users' activities may be disclosed or abused by malicious users, which will threaten all other users. This problem has to be approached by several angles, including well-designed policies restricting the flow of data and information, and also by carefully considering which data needs to be anonymised, and which can be encrypted.

5 Conclusion

In this vision paper, we have discussed a mobile crowdsourcing paradigm which can augment the current web-based crowdsourcing platforms to provide real-time location based query response using mobile devices. Towards this goal, we have provided a hybrid-crowdsourcing architecture and discussed several facets to realize this vision. This mobile crowdsourcing approaches can filter and target workers who more closely matches not only the queries, but also the location and context requirements. In addition, some of the processing can be done in the centralized web-based part of the proposed architecture, which reduces the burden of processing queries on the mobile devices. This allows several new application of crowdsourcing, especially in the areas of crowd-sensing, ubiquitous computing, and smart cities.

References

1. Behrend, T.S., Sharek, D.J., Meade, A.W., Wiebe, E.N.: The viability of crowdsourcing for survey research. *Behav. Res. Meth.* **43**(3), 800–813 (2011)
2. Bonabeau, E.: Decisions 2.0: the power of collective intelligence. *MIT Sloan Manag. Rev.* **50**, 45–52 (2009)
3. Borst, I.: Understanding crowdsourcing - effects of motivation and rewards on participation and performance in vountary online activities. Ph.D. thesis, Erasmus research institute of Management, Rotterdam School of Management, Erasmus School of Economics, Erasmus University Rotterdam (2010)
4. den Ende, J.V., Villarroel, A., Tucci, C.: Strategic crowdsourcing, orchestrating innovation through the cream of the crowd. In: Panel Symposium, Academy of Management Conference (2009)
5. Eagle, N.: txt eagle: mobile crowdsourcing. In: Aykin, Nuray (ed.) IDGD 2009. LNCS, vol. 5623, pp. 447–456. Springer, Heidelberg (2009)
6. Franklin, M.J., Kossmann, D., Kraska, T., Ramesh, S., Xin, R.: Crowddb: answering queries with crowdsourcing. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp. 61–72. ACM (2011)
7. Gupta, A., Thies, W., Cutrell, E., Balakrishnan, R.: mclerk: Enabling mobile crowdsourcing in developing regions. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '12, pp. 1843–1852. ACM (2012)
8. He, J., van Ossensbruggen, J., de Vries, A.P.: Do you need experts in the crowd?: a case study in image annotation for marine biology. In: Proceedings of the 10th Conference on Open Research Areas in Information Retrieval, pp. 57–60 (2013)
9. Howe, J.: The rise of crowdsourcing **14**(6) (2009). <http://www.wired.com/wired/archive/14.06/crowds.html>
10. Jouret, G.: Inside cisco's search for the next big idea. *Harvard Bus. Rev.* **87**, 43–45 (2009)
11. Kittur, A., Chi, E.H., Suh, B.: Crowdsourcing user studies with mechanical turk. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 453–456. ACM (2008)
12. Komninos, N.: *Intelligent Cities: Innovation, Knowledge Systems and Digital Spaces*. Spon Press, London (2002)
13. Komninos, N.: Intelligent cities: towards interactive and global innovation environments. *Int. J. Innovation Reg. Dev.* **1**(4), 337–355 (2009)
14. Kunze, K., Kawaichi, H., Yoshimura, K., Kise, K.: Towards inferring language expertise using eye tracking. In: CHI'13 Extended Abstracts on Human Factors in Computing Systems, pp. 217–222. ACM (2013)
15. Lampel, J., Bhalla, A.: The role of status seeking in online communities: giving the gift of experience. *J. Comput. Mediated Commun.* **122** (2007). <http://jcmc.indiana.edu/vol12/issue2/lampel.html>
16. Lofi, C., Selke, J., Balke, W.-T.: Information extraction meets crowdsourcing: a promising couple. *Datenbank-Spektrum* **12**(2), 109–120 (2012)
17. Lukowicz, P., Pentland, A., Ferscha, A.: From context awareness to socially aware computing. *IEEE Pervasive Comput.* **11**(1), 32–41 (2012)
18. Narula, P., Gutheim, P., Rolnitzky, D., Kulkarni, A., Hartmann, B.: Mobileworks: a mobile crowdsourcing platform for workers at the bottom of the pyramid. In: *Human Computation* (2011)
19. Schuermann, D., Sigg, S.: Secure communication based on ambient audio. *IEEE Trans. Mob. Comput.* **12**(2), 358–370 (2013)

20. Selke, J., Lofi, C., Balke, W.-T.: Pushing the boundaries of crowd-enabled databases with query-driven schema expansion. *Proc. VLDB Endowment* **5**(6), 538–549 (2012)
21. Sigg, S., Schuermann, D., Ji, Y.: Pintext: a framework for secure communication based on context. In: *Proceedings of the Eighth Annual International ICST Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2011)* (2011)
22. Sigg, S., Gordon, D., von Zengen, G., Beigl, M., Haseloff, S., David, K.: Investigation of context prediction accuracy for different context abstraction levels. *IEEE Trans. Mob. Comput.* **11**(6), 1047–1059 (2012)
23. Sigg, S., Scholz, M., Shi, S., Ji, Y., Beigl, M.: Rf-sensing of activities from non-cooperative subjects in device-free recognition systems using ambient and local signals. *IEEE Trans. Mob. Comput.* **13**(4) (2013). doi <http://doi.ieeecomputersociety.org/10.1109/TMC.2013.28>
24. Sigg, S., Blanke, U., Troester, G.: The telepathic phone: frictionless activity recognition from wifi-rssi. In: *IEEE International Conference on Pervasive Computing and Communications (PerCom), PerCom '14* (2014)
25. Wu, C., Gerlach, J., Young, C.: An empirical analysis of open source software developers motivations and continuance intentions. *Inf. Manage.* **44**, 253–262 (2007)

Conditioning Probabilistic Relational Data with Referential Constraints

Ruiming Tang¹(✉), Dongxu Shao¹, M. Lamine Ba², and Huayu Wu³

¹ National University of Singapore, Singapore, Singapore
{tangruiming, dcsshao}@nus.edu.sg

² Institut Mines-Télécom, Télécom ParisTech; LTCI, Paris, France
mouhamadou.ba@telecom-paristech.fr

³ Institute for Infocomm Research, Singapore, Singapore
huwu@i2r.a-star.edu.sg

Abstract. A probabilistic relational database is a compact form of a set of deterministic relational databases (namely, *possible worlds*), each of which has a probability. In our framework, the existence of tuples is determined by associated Boolean formulae based on elementary events. An estimation, within such a setting, of the probabilities of possible worlds uses a prior probability distribution specified over the elementary events. Direct observations and general knowledge, in the form of constraints, help refining these probabilities, possibly ruling out some possible worlds. More precisely, new constraints can translate the observation of the existence or non-existence of a tuple, the knowledge of a well-defined rule, such as primary key constraint, foreign key constraint, referential constraint, etc. Informally, the process of enforcing knowledge on a probabilistic database, which consists of computing a new subset of valid possible worlds together with their new (conditional) probabilities, is called *conditioning*. In this paper, we are interested in finding a new probabilistic relational database after conditioning with referential constraints involved. In the most general case, conditioning is intractable. As a result, we restricted our study to probabilistic relational databases in which formulae of tuples are *independent events* in order to achieve some tractability results. We devise and present polynomial algorithms for conditioning probabilistic relational databases with referential constraints.

1 Introduction

Uncertainty of data naturally arises from such applications as information extraction [6], information integration [9, 18] and version control [3], for instance.

Probabilistic databases address the problem of the management and of the representation of uncertain data by means of probabilities. A good probabilistic database model offers a representation of uncertain data that is generally compact and easy to be managed. In a probabilistic database, an instance denotes a set of possible deterministic database instances called *possible worlds*, each of which has a probability that can be initially estimated according to limited

prior knowledge of data. Probabilistic relational databases [2, 4, 5, 8, 10–12, 15, 16] represent uncertainties at attribute or tuple level, while schema is constrained.

Direct observations and general knowledge, in the form of constraints (such as the existence or non-existence of a tuple, primary key constraint, foreign key constraint, referential constraint, etc.), can be enforced into the database during a data cleaning process, for instance. These constraints help refining the probabilities of the possible worlds, possibly ruling out some of them. Enforcing such constraints to the set of possible worlds with their probabilities is called *conditioning* the probabilistic database. The core problem in probabilistic database conditioning is to find a new probabilistic database instance that denotes the subset of possible worlds with their new probabilities corresponding to the conditional probabilities.

The conditioning problem in probabilistic relational databases has been studied in [14, 17], respectively. Koch and Olteanu [14] show that relational conditioning is NP-Hard. In our previous work [17], we identify tractable scenarios for which we devise polynomial time algorithms. In [17], we focus on the special case in which tuples are independent, and the kinds of constraints considered in the tractable cases studied are observation constraints (i.e. existence constraints) and X-tuple constraints (i.e. primary key constraints).

In this paper, we continue studying the problem of conditioning probabilistic relational data. We adopt the data model in [17] and also focus on the special class of probabilistic relational databases in which the formulae attached to the tuples are *independent events*. We investigate a different kind of constraints, namely referential constraints (also called inclusion dependency [1]), from the ones we studied in [17]. A *referential constraint* is in the form $R[r_1 \dots r_m] \subseteq S[s_1 \dots s_n]$, where R, S are (possibly identical) relation names, $r_1 \dots r_m$ is a subset of distinct attribute names of R and $s_1 \dots s_n$ is a subset of distinct attribute names of S [1]. This referential constraint is a *foreign key constraint* if $s_1 \dots s_n$ is the primary key of S . A referential constraint in a real-life example is presented in Example 1.

Example 1. There are two relations: Movies (Table 1) and Showings (Table 2). Movies stores a set of movies with movies' title, director and actors. Showings stores locations (theater and hall) where movies are showed. Each tuple is associated with an independent event, representing the probability of this tuple being actual. There is a referential constraint saying that all the showed movies must be produced first, i.e. $Showings[title] \subseteq Movies[title]$. \square

The rest of this paper is structured as follows. We start by reviewing state-of-the-art probabilistic relational models and conditioning probabilistic relational databases in Sect. 2. Then, we present in Sect. 3 the probabilistic relational data model we consider in this paper. We describe how a referential constraint can be transformed to independent implication constraints and present three classes of implication constraints in Sect. 4. Tractable algorithms are presented for individual classes of implication constraints in Sects. 5, 6 and 7 respectively. The class of constraints in Sect. 7 covers the first two classes in Sects. 5, 6 and it

Table 1. Movies

Title	Director	Actor	Event
Hobbit 2	Peter Jackson	Martin Freeman	$e_{(b,1)}$
Hobbit 2	Peter Jackson	Ian McKellen	$e_{(b,2)}$
Hunger Game 2	Francis Lawrence	Jennifer Lawrence	$e_{(b,3)}$
Hunger Game 2	Francis Lawrence	Liam Hemsworth	$e_{(b,4)}$
Fast & Furious 7	James Wan	Vin Diesel	$e_{(b,5)}$
Fast & Furious 7	James Wan	Paul Walker	$e_{(b,6)}$

Table 2. Showings

Theater	Hall	Title	Event
Golden Village	Hall 1	Hunger Game 2	$e_{(a,1)}$
Golden Village	Hall 2	Hunger Game 2	$e_{(a,2)}$
Golden Village	Hall 3	Hobbit 2	$e_{(a,3)}$
Golden Village	Hall 4	Hobbit 2	$e_{(a,4)}$

actually is the class of general referential constraints. We conclude and present future work in Sect. 8.

2 Related Work

We briefly review hereafter state-of-the-art probabilistic relational data models and conditioning algorithms.

2.1 Probabilistic Relational Models

Probabilistic relational data models can be separated into two groups: tuple-level models and attribute-level models. Granularity of uncertainty differs between the two groups of models.

One simple idea to define a probabilistic relational model is the tuple-mutually-exclusive model [5]. In this model, a probabilistic database is an ordinary database where each tuple is associated with a probability of being actual. Each tuple is mutually exclusive to any other tuple in the same probabilistic relation. The sum of probabilities of all the tuples in a probabilistic relation is 1. In the tuple-independent model [7], a probabilistic database is an ordinary database where each tuple is associated with a probability of being actual, independent from any other tuple. Mixing previous two models is the block-independent-disjoint model [2, 8, 15]: tuples within a block are mutually-exclusive, while tuples across different blocks are independent.

The expressiveness of block-independent-disjoint model is stronger than tuple-independent model and tuple-mutually-exclusive model. However, it is not expressive enough in some cases, when relationships between tuples, other than independence and mutually exclusion, are needed.

The most expressive model is probabilistic c -table (e.g. [11–13]). In this model, each tuple is associated with a formula constructed from Boolean event using operators \wedge, \vee, \neg . Boolean events are independent and associated with probabilities of being true. A tuple is actual if its associated formula is evaluated to be true.

We adopt the probabilistic relational data model from [17]. It is a tuple-level model with additional constraints integrated as first class citizens, i.e., extended probabilistic c -table. The model caters for constraints rather than treating them as add-ons, as other tuple-level models can be adapted to.

2.2 Conditioning

Koch and Olteanu [14] show that relational conditioning is NP-Hard, since the probability computation problem of an arbitrary Boolean expression is hard. They present a general but exponential time algorithm as well as efficient heuristics and decomposition methods.

In [17], we identify tractable scenarios for which we devise polynomial time algorithms. We focus on the special case in which tuples are independent and the tractable cases studied are observation constraints (i.e. existence constraints) and X-tuple constraints (i.e. primary key constraints).

In this paper, we also focus on the special case in which tuples are independent, but we study a different set of constraints, namely referential constraints, from the ones studied in [17].

3 Data Model

In [17], we introduced a framework for conditioning probabilistic relational data. In this section, we briefly review the conditioning framework in [17]. In the end of this section, we present two theorems which are not studied in [17].

Let E be a set of symbols called *events* (e). A *complex event* (ce) is a well formed formula of propositional logic in which events are propositions. $C(E)$ is the set of complex events formed with the events in E . An *interpretation* of a complex event is a function from E to $\{\text{true}, \text{false}\}$.

Definition 1. (*Probabilistic Relational Database*). A **probabilistic relational database** \mathcal{D} is a quintuple $\langle D, E, f, C, p \rangle$: D is a traditional database instance, E is a set of events, f is a function from D to $C(E)$, C is a subset of $C(E)$, and p is a function from E to $[0, 1]$.

The probability of a complex event c , noted $p(c)$, is

$$p(c) = \sum_{I \in \mathcal{M}(c)} \left(\prod_{I(e)=\text{true}} p(e) \times \left(\prod_{I(e)=\text{false}} (1 - p(e)) \right) \right)$$

where $\mathcal{M}(c)$ is the set of models of c .

Informally and in short, a possible world is a traditional relational database such that the complex events associated with the tuples in the possible world are **true**, the complex events associated with the tuples not in the possible world are **false**, and the constraints in C are **true**. The probability of a possible world is the probability to find such a model given the probabilities of individual events, under the condition that the constraints are held.

Definition 2. (*Possible Worlds*). Let $\mathcal{D} = \langle D, E, f, C, p \rangle$ be a probabilistic relational database. D' is a **possible world** of \mathcal{D} if and only if there exists a model of the following formula F with a non zero probability.

$$F = \left(\bigwedge_{t_i \in D'} f(t_i) \right) \wedge \left(\bigwedge_{t_i \notin D'} \neg f(t_i) \right) \wedge \left(\bigwedge_{c \in C} c \right) \quad \text{and} \quad p(F) \neq 0$$

We call $p_{\mathcal{D}}(D')$ the probability, $p(F|C)$, of the possible world D' in the probabilistic relational database \mathcal{D} . We call $\mathfrak{P}(\mathcal{D})$ the set of possible worlds of \mathcal{D} .

\mathcal{D} is **consistent** (resp. **inconsistent**) if and only if there exists at least one possible world (resp. there does not exist a possible world) of \mathcal{D} , i.e. $\mathfrak{P}(\mathcal{D}) \neq \emptyset$ (resp. $\mathfrak{P}(\mathcal{D}) = \emptyset$). \mathcal{D} is inconsistent iff C is always evaluated to be false. Note that we define the concept of “consistent” only for defining the conditioning problem.

We defined an equivalent relationship on probabilistic relational databases. Two probabilistic relational databases are *world-equivalent* if they have the same possible worlds with the same probabilities, i.e. $\mathcal{D}_1 \equiv_w \mathcal{D}_2$ if and only if:

$$\forall D' \subseteq D \left((D' \in \mathfrak{P}(\mathcal{D}_1)) \leftrightarrow (D' \in \mathfrak{P}(\mathcal{D}_2)) \right) \quad \text{and}$$

$$\forall D' \in \mathfrak{P}(\mathcal{D}_1) (p_{\mathcal{D}_1}(D') = p_{\mathcal{D}_2}(D'))$$

We showed in [17] that for any consistent probabilistic relational database \mathcal{D}_1 then there exists a probabilistic relational database $\mathcal{D}_2 = \langle D, E_2, f_2, \emptyset, p_2 \rangle$ such that $\mathcal{D}_2 \equiv_w \mathcal{D}_1$.

Conditioning a probabilistic relational database consists in adding constraints. The **conditioning problem** consists in finding a world-equivalent probabilistic relational database with no constraint, given one probabilistic relational database with constraints.

We prove below two propositions which are not studied in [17].

Proposition 1. Let $\mathcal{D}_1 = \langle D, E_1, f_1, C, p_1 \rangle$ be a consistent probabilistic relational database. Let $\mathcal{D}_2 = \langle D, E_2, f_2, \emptyset, p_2 \rangle$ be a probabilistic relational database such that $\mathcal{D}_1 \equiv_w \mathcal{D}_2$. For a tuple $t \in D$, if $f_1(t)$ is independent of C , it is possible to condition so that $p_1(f_1(t)) = p_2(f_2(t))$ and it is possible to have $f_2(t) = f_1(t)$.

Proof.

$$p_2(f_2(t)) = p_1(f_1(t)|C) = \frac{p_1(f_1(t) \wedge C)}{p_1(C)} = \frac{p_1(f_1(t)) \cdot p_1(C)}{p_1(C)} = p_1(f_1(t))$$

Since the formula of t is independent of C , it is possible to keep it unchanged.

For the other tuples in D (whose associated formulae are dependent on C), their formula have to be updated and the probabilities of new created events have to be defined. □

In this paper, we consider a special class of probabilistic relational databases, namely *probabilistic relational databases with independent events* as the input of the conditioning problem.

Proposition 2. *Let $\mathcal{D} = \langle D, E_1, f_1, C, p_1 \rangle$ be a consistent probabilistic relational database with independent events. For a tuple $t \in D$, if $f_1(t)$ is independent of C , it is possible to condition so that its formula $f_2(t)$ is also an unique independent event and $p_1(f_1(t)) = p_2(f_2(t))$.*

Proposition 2 is a direct consequence of Proposition 1.

4 Referential Constraints

In this section, we first present how a referential constraint can be transformed to a set of independent implication constraints and we introduce the three kinds of implication constraints for which we will then present algorithms in the following sections. We then introduce one particular technical tool which relates to the *relevant part* of a probabilistic relational database for a given constraint. We also present the notion of *possible worlds according to the relevant part* of a probabilistic relational database for a constraint.

4.1 Transforming Referential Constraints to Implication Constraints

We start this section by defining the implication constraints.

Definition 3. (Implication Constraints). *An **implication constraint** is a well formed formula of propositional logic whose format is*

$$\bigvee_{i=1}^m e_{(a,i)} \Rightarrow \bigvee_{i=1}^n e_{(b,i)}$$

A referential constraint can be transformed to a set of independent implication constraints. We illustrate this remark by using the example below.

Example 2. Consider the referential constraint in Example 1, i.e., all the showed movies must be produced first, i.e. $Showings[title] \subseteq Movies[title]$. In this example, the referential constraint is transformed to two independent implication constraints. If “Hobbit 2” is showed, then it must be produced, i.e. $(e_{(a,3)} \vee e_{(a,4)}) \Rightarrow (e_{(b,1)} \vee e_{(b,2)})$. If “Hunger Game 2” is showed, then it must be produced, i.e. $(e_{(a,1)} \vee e_{(a,2)}) \Rightarrow (e_{(b,3)} \vee e_{(b,4)})$. Note that this result in only under the probabilistic relational model with independent events. □

In the next section, we will prove that conditioning two independent constraints yields the same result as conditioning them separately (Theorem 2), therefore **we consider conditioning a single implication constraint in the rest of the paper**. Note that when there are multiple referential constraints or there are multiple *dependent* implication constraints, our algorithms (which will be presented later) cannot be applied to solve the conditioning problem, because tuples affected by the constraints may have complicated formulae to make our algorithm fail to work.

Given a referential constraint $R[r_1, r_2, \dots, r_m] \subseteq S[s_1, s_2, \dots, s_n]$, we present below three classes of implication constraints (in the form of $\bigvee_{i=1}^m f_1(t_{(a,i)}) \Rightarrow \bigvee_{i=1}^n f_1(t_{(b,i)})$). In the rest of this paper, we use A to represent the set of tuples $\{t_{(a,1)}, \dots, t_{(a,m)}\}$ and we use B to present the set of tuples $\{t_{(b,1)}, \dots, t_{(b,n)}\}$.

- when r_1, r_2, \dots, r_m is the primary key of R and s_1, s_2, \dots, s_n is the primary key of S , we have $A = \{t_{(a,1)}\}, B = \{t_{(b,1)}\}$. We refer to this class as FKPK constraints. This class of constraints is realistic when relation R is a subclass of relation S .
- when s_1, s_2, \dots, s_n is the primary key of S , we have $A = \{t_{(a,1)}, \dots, t_{(a,m)}\}, B = \{t_{(b,1)}\}$. This is the class of foreign key constraints, and we refer to this class as FK constraints.
- for a general referential constraint, we have $A = \{t_{(a,1)}, \dots, t_{(a,m)}\}, B = \{t_{(b,1)}, \dots, t_{(b,n)}\}$. This is the **most general class of referential constraints** and we refer to this class as REF constraints.

4.2 Local Database and Local Possible Worlds

Corollary 1. *Let $\mathcal{D} = \langle D, E_1, f_1, C, p_1 \rangle$ be a consistent probabilistic relational database with independent events and C be an implication constraint. After conditioning, (1) the formulae of all the tuples in A and B must be updated and the probabilities of new events must be defined; (2) for the other tuples, the formulae and probabilities are unmodified.*

This corollary can be deduced from Proposition 2, because if $t \notin A \cup B$, then $f_1(t)$ is independent of C .

Definition 4. (*Local Database*). *Let $\mathcal{D} = \langle D, E_1, f_1, C, p_1 \rangle$ be a consistent probabilistic relational database with independent events and C be an implication constraint over two sets of tuples¹ A, B . The **local database**, that we denote by $LD(C, D)$ of D with respect to C is $A \cup B$.*

Theorem 1. *Let $\mathcal{D}_1 = \langle D, E_1, f_1, C, p_1 \rangle$ be a consistent probabilistic relational database with independent events and $\mathcal{D}_2 = \langle D, E_2, f_2, \emptyset, p_2 \rangle$ be a probabilistic relational database. C is an implication constraint. We claim that $\mathcal{D}_1 \equiv_w \mathcal{D}_2$ iff $\langle LD(C, D), E_1, f_1, C, p_1 \rangle \equiv_w \langle LD(C, D), E_2, f_2, \emptyset, p_2 \rangle$.*

¹ We do not distinguish tuples and their associated events because we study the probabilistic relational model with independent events.

Theorem 2. Let $\mathcal{D}_1 = \langle D, E_1, f_1, \{C_1, C_2\}, p_1 \rangle$ be a consistent probabilistic relational database with independent events, where C_1, C_2 are two independent implication constraints. $\mathcal{D}_2 = \langle D, E_2, f_2, \emptyset, p_2 \rangle$ is a probabilistic relational database. We claim that $\mathcal{D}_2 \equiv_w \mathcal{D}_1$ iff $\langle LD(C_1, D), E_1, f_1, C_1, p_1 \rangle \equiv_w \langle LD(C_1, D), E_2, f_2, \emptyset, p_2 \rangle$ and $\langle LD(C_2, D), E_1, f_1, C_2, p_1 \rangle \equiv_w \langle LD(C_2, D), E_2, f_2, \emptyset, p_2 \rangle$.

Proof. Since C_1, C_2 are independent, and the events of tuples are also independent, we deduce that $LD(C_1, D) \cap LD(C_2, D) = \emptyset$. Moreover, due to Proposition 2, the formulae of tuples in $LD(C_1, D)$ are not updated by enforcing C_2 , and conversely the formulae of tuples in $LD(C_2, D)$ are not changed by enforcing C_1 . Therefore we can enforce C_1, C_2 separately without affecting each other. Based on Theorem 1, we can prove the desired claim. \square

Definition 5. (*Local Possible Worlds*). The **local possible worlds** of $\mathcal{D}_1 = \langle D, E_1, f_1, C, p_1 \rangle$ correspond to the possible worlds of $\langle LD(C, D), E_1, f_1, C, p_1 \rangle$.

If the number of local possible worlds is polynomial to the size of the local database, a P-TIME conditioning algorithm is trivial to devise by enumerating all the local possible worlds. While the number of local possible worlds is exponential to the size of the local database, finding a P-TIME conditioning algorithm is a challenge. Therefore, before going to study conditioning algorithms, we show the numbers of local possible worlds for different classes of implication constraints.

For an FKPK constraint, the number of local possible worlds is 3: (1) 1 local possible worlds when $t_{(a,1)}, t_{(b,1)}$ exist; (2) 1 local possible worlds when $t_{(a,1)}$ does not exist and $t_{(b,1)}$ exists; (3) 1 local possible worlds when neither of $t_{(a,1)}, t_{(b,1)}$ exist.

For an FK constraint, the number of local possible worlds is $2^m + 1$, which is exponential to the number of tuples in the local database: (1) when at least one of $\{t_{(a,1)}, \dots, t_{(a,m)}\}$ exists and $t_{(b,1)}$ exists, there are $2^m - 1$ local possible worlds; (2) when $\{t_{(a,1)}, \dots, t_{(a,m)}\}$ does not exist and $t_{(b,1)}$ exists, there is 1 local possible worlds; (3) when $\{t_{(a,1)}, \dots, t_{(a,m)}\}$ does not exist and $t_{(b,1)}$ does not exist, there is 1 local possible worlds.

For a REF constraint, the number of local possible worlds is $2^n + (2^m - 1)(2^n - 1)$, which is exponential to the number of tuples in the local database: (1) when at least one of $\{t_{(a,1)}, \dots, t_{(a,m)}\}$ exists and at least one of $\{t_{(b,1)}, \dots, t_{(b,n)}\}$ exists, there are $(2^m - 1)(2^n - 1)$ local possible worlds; (2) when $\{t_{(a,1)}, \dots, t_{(a,m)}\}$ does not exist and at least one of $\{t_{(b,1)}, \dots, t_{(b,n)}\}$ exists, there are $2^n - 1$ local possible worlds; (3) when $\{t_{(a,1)}, \dots, t_{(a,m)}\}$ does not exist and $\{t_{(b,1)}, \dots, t_{(b,n)}\}$ does not exist as well, there is 1 local possible world.

5 Conditioning Algorithm for FKPK Constraints

In this section, we present a conditioning algorithm for FKPK constraints. In an FKPK constraint, $A = \{t_i\}$ and $B = \{t_j\}$. Assume $f_1(t_i) = e_i$ and $f_1(t_j) = e_j$. The local database is $LD(C, D) = \{t_i, t_j\}$.

Algorithm 1. Conditioning algorithm for FKPK implication constraints**Data:** $\langle \text{LD}(C, D), E_1, f_1, C, p_1 \rangle$ **Result:** A world equivalent $\langle \text{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$ **1** $f_2(t_i) \leftarrow x_i, f_2(t_j) \leftarrow x_i \vee x_j;$ **2** $p_2(x_i) = \frac{p_1(e_i)p_1(e_j)}{p_1(C)}, p_2(x_j) = p_1(e_j);$

The implication constraint is formulated as $e_i \Rightarrow e_j$. Its probability is computed as

$$p_1(C) = p_1(\neg e_i \vee e_j) = p_1(e_j) + (1 - p_1(e_i))(1 - p_1(e_j))$$

The probability of such a constraint can be computed in linear time. Algorithm 1 presents the conditioning algorithm for FKPK constraints. It introduces 2 events for the tuples. The time complexity of Algorithm 1 is linear to the size of the local database.

Theorem 3. *Algorithm 1 is correct and performs in linear time.*

Proof. There are three possible worlds of $\langle \text{LD}(C, D), E_1, f_1, C, p_1 \rangle$: in K_1 , neither of t_i and t_j exists; in K_2 , only t_j exists; in K_3 , both of t_i and t_j exist. After conditioning, $\langle \text{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$ has the same three possible worlds. We are going to prove that for each possible world, its probability after conditioning is the same as before conditioning.

– For K_1 ,

$$\begin{aligned} p_2(K_1) &= p_2(\neg x_i \wedge \neg x_j) = (1 - p_2(x_i))(1 - p_2(x_j)) \\ &= \frac{p_1(C) - p_1(e_i)p_1(e_j)}{p_1(C)}(1 - p_1(e_j)) = \frac{(1 - p_1(e_i))(1 - p_1(e_j))}{p_1(C)} = p_1(K_1) \end{aligned}$$

– For K_2 ,

$$\begin{aligned} p_2(K_2) &= p_2(\neg x_i \wedge x_j) = (1 - p_2(x_i))p_2(x_j) \\ &= \frac{p_1(C) - p_1(e_i)p_1(e_j)}{p_1(C)}p_1(e_j) = \frac{(1 - p_1(e_i))p_1(e_j)}{p_1(C)} = p_1(K_2) \end{aligned}$$

– For K_3 ,

$$p_2(K_3) = p_2(x_i) = \frac{p_1(e_i)p_1(e_j)}{p_1(C)} = p_1(K_3)$$

Algorithm 1 performs in linear time since it introduces two events and their probabilities can be computed in constant time. \square

Algorithm 2. Conditioning algorithm for FK implication constraints

Data: $\langle \text{LD}(C, D), E_1, f_1, C, p_1 \rangle$ **Result:** A world equivalent $\langle \text{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$

```

1 foreach  $i \in [1, m]$  do
2   |  $f_2(t_{(a,i)}) \leftarrow \eta \wedge x_i$ ;
3  $f_2(t_{(b,1)}) \leftarrow (\eta \wedge \bigvee_{i=1}^m x_i) \vee \lambda$ 
4 foreach  $i \in [1, m]$  do
5   |  $p_2(x_i) = p_1(e_{(a,i)})$ ;
6  $p_2(\lambda) = p_1(e_{(b,1)}), p_2(\eta) = \frac{p_1(e_{(b,1)})}{p_1(C)}$ ;

```

6 Conditioning Algorithm for FK Constraints

In this section, we present a conditioning algorithm for FK constraints. In an FK constraint, $A = \{t_{(a,1)}, \dots, t_{(a,m)}\}$ and $B = \{t_{(b,1)}\}$. Assume $f_1(t_i) = e_i$. The local database is $\text{LD}(C, D) = \{t_{(a,1)}, \dots, t_{(a,m)}, t_{(b,1)}\}$.

The constraint is formulated as $\bigvee_{i=1}^m e_{(a,i)} \Rightarrow e_{(b,1)}$. Its probability is computed as

$$\begin{aligned}
 p_1(C) &= p_1(\neg \bigvee_{i=1}^m e_{(a,i)} \vee e_{(b,1)}) = p_1(\bigwedge_{i=1}^m \neg e_{(a,i)} \vee e_{(b,1)}) \\
 &= p_1(\bigwedge_{i=1}^m \neg e_{(a,i)}) + p_1(e_{(b,1)}) - p_1(\bigwedge_{i=1}^m \neg e_{(a,i)} \wedge e_{(b,1)}) \\
 &= \prod_{i=1}^m (1 - p_1(e_{(a,i)})) + p_1(e_{(b,1)}) - \prod_{i=1}^m (1 - p_1(e_{(a,i)})) p_1(e_{(b,1)}) \\
 &= p_1(e_{(b,1)}) + (1 - p_1(e_{(b,1)})) \prod_{i=1}^m (1 - p_1(e_{(a,i)}))
 \end{aligned}$$

The probability of such a constraint can be computed in linear time. Algorithm 2 presents the conditioning algorithm for FK constraints. It introduces $m + 2$ events. The time complexity of Algorithm 2 is linear to the size of the local database.

Theorem 4. *Algorithm 2 is correct and performs in linear time to size of the local database.*

Proof. There are $2^m + 1$ possible worlds of $\langle \text{LD}(C, D), E_1, f_1, C, p_1 \rangle$: in K_1 , there is no tuple; in K_2 , there is only $t_{(b,1)}$ exists; in the other $2^m - 1$ possible worlds, $t_{(b,1)}$ exists and at least one of $t_{(a,1)}, \dots, t_{(a,m)}$ exists. After conditioning, $\langle \text{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$ has the same possible worlds. We are going to prove that for each possible world, its probability after conditioning is the same as before conditioning.

– For K_1 ,

$$\begin{aligned}
 p_2(K_1) &= p_2(\neg\lambda \wedge (\neg\eta \vee \eta \wedge \bigwedge_{i=1}^m \neg x_i)) = (1 - p_2(\lambda))(1 - p_2(\eta) + p_2(\eta) \prod_{i=1}^m (1 - p_2(x_i))) \\
 &= (1 - p_1(e_{(b,1)}))(1 - \frac{p_1(e_{(b,1)})}{p_1(C)} + \frac{p_1(e_{(b,1)})}{p_1(C)} \prod_{i=1}^m (1 - p_1(e_{(a,i)}))) \\
 &= \frac{1 - p_1(e_{(b,1)})}{p_1(C)} (p_1(C) - p_1(e_{(b,1)}) + p_1(e_{(b,1)}) \prod_{i=1}^m (1 - p_1(e_{(a,i)}))) \\
 &= \frac{1 - p_1(e_{(b,1)})}{p_1(C)} \prod_{i=1}^m (1 - p_1(e_{(a,i)})) = p_1(K_1)
 \end{aligned}$$

- For K_2 , the proof is similar to K_1 , except that λ is true in K_2 .
 – For the other $2^m - 1$ possible worlds, their proofs are similar, therefore we only show one of them. Consider the possible world K that $t_{(b,1)}$ exists, $t_{(a,k)}$ exists and all the others do not.

$$\begin{aligned}
 p_2(K) &= p_2(\eta \wedge x_k \wedge \bigwedge_{i=1, i \neq k}^m \neg x_i) = p_2(\eta) p_2(x_k) \prod_{i=1, i \neq k}^m (1 - p_2(x_i)) \\
 &= \frac{p_1(e_{(b,1)})}{p_1(C)} p_1(e_{(a,k)}) \prod_{i=1, i \neq k}^m (1 - p_1(e_{(a,i)})) = p_1(K)
 \end{aligned}$$

Algorithm 2 performs in linear time since it introduces $m + 2$ events and their probabilities can be computed in constant time. \square

We illustrate Algorithm 2 by an example.

Example 3. $A = \{t_{(a,1)}, t_{(a,2)}, t_{(a,3)}\}$ and $B = \{t_{(b,1)}\}$. $f_1(t_{(a,1)}) = e_{(a,1)}$, $f_1(t_{(a,2)}) = e_{(a,2)}$, $f_1(t_{(a,3)}) = e_{(a,3)}$, $f_1(t_{(b,1)}) = e_{(b,1)}$. The FK implication constraint is $(e_{(a,1)} \vee e_{a,2} \vee e_{a,3}) \Rightarrow e_{(b,1)}$.

After conditioning, $f_2(t_{(a,1)}) = \eta \wedge x_1$, $f_2(t_{(a,2)}) = \eta \wedge x_2$, $f_2(t_{(a,3)}) = \eta \wedge x_3$, $f_2(t_{(b,1)}) = (\eta \wedge (x_1 \vee x_2 \vee x_3)) \vee \lambda$. The probabilities of new events are $p_2(x_1) = p_1(e_{(a,1)})$, $p_2(x_2) = p_1(e_{(a,2)})$, $p_2(x_3) = p_1(e_{(a,3)})$, $p_2(\lambda) = p_1(e_{(b,1)})$, $p_2(\eta) = \frac{p_1(e_{(b,1)})}{p_1(C)}$. \square

7 Conditioning Algorithm for REF Constraints

In this section, we present conditioning algorithms for REF constraints.

7.1 Simplified Case

To understand the conditioning algorithm for REF constraints better, we consider a simplified version of REF constraint first. In this case, $A = \{t_{(a,1)}\}$ and $B = \{t_{(b,1)}, \dots, t_{(b,n)}\}$. Assume $f_1(t_i) = e_i$. The local database is $\text{LD}(C, D) =$

Algorithm 3. Conditioning algorithm for simplified REF implication constraints

Data: $\langle \text{LD}(C, D), E_1, f_1, C, p_1 \rangle$

Result: A world equivalent $\langle \text{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$

```

1  $f_2(t_{(a,1)}) \leftarrow \lambda$ 
2  $f_2(t_{(b,1)}) \leftarrow (\lambda \wedge x_1) \vee y_1$ 
3 foreach  $i \in [2, n-1]$  do
4    $f_2(t_{(b,i)}) \leftarrow (\lambda \wedge \bigwedge_{j=1}^{i-1} (\neg x_j \wedge \neg y_j) \wedge x_i) \vee y_i;$ 
5  $f_2(t_{(b,n)}) \leftarrow (\lambda \wedge \bigwedge_{j=1}^{n-1} (\neg x_j \wedge \neg y_j)) \vee y_n;$ 
6 foreach  $i \in [1, n]$  do
7    $p_2(y_i) = p_1(e_{(b,i)});$ 
8 foreach  $i \in [1, n-1]$  do
9    $p_2(x_i \vee y_i) = \frac{p_1(e_{(b,i)})}{p_1(\bigvee_{j=i}^n e_{(b,j)})};$ 
10  $p_2(\lambda) = \frac{p_1(e_{(a,1)})p_1(\bigvee_{i=1}^n e_{(b,i)})}{p_1(C)};$ 

```

$\{t_{(a,1)}, t_{(b,1)}, \dots, t_{(b,n)}\}$. The constraint is formulated as $e_{(a,1)} \Rightarrow \bigvee_{i=1}^n e_{(b,i)}$. Its probability is computed as

$$p_1(C) = p_1(\neg e_{(a,1)} \vee \bigvee_{i=1}^n e_{(b,i)}) = 1 - p_1(e_{(a,1)}) + p_1(e_{(a,1)})p(\bigvee_{i=1}^n e_{(b,i)})$$

The probability of such a constraint can be computed in linear time, since the probability of a disjunction of independent events can be computed associatively. Algorithm 3 presents the conditioning algorithm for simplified REF constraints. It introduces $2n$ events. The time complexity of Algorithm 3 is linear to the size of the local database.

Theorem 5. *Algorithm 3 is correct and performs in linear time to size of the local database.*

We omit the proof since it is similar to the proof of previous theorems. The basic idea of the proof is to prove for every possible world, its probability is the same after conditioning as before. Moreover, we have to make sure all the probability values are valid, i.e.,

- $p_2(x_i \vee y_i) \in [0, 1]$.
It is true, because $p_1(e_{(b,i)}) \leq p_1(\bigvee_{j=i}^n e_{(b,j)})$ and from line 9 of Algorithm 3 we know that $p_2(x_i \vee y_i) \in [0, 1]$.
- $p_2(x_i \vee y_i) \geq p_2(y_i)$.
It is true, because $p_1(\bigvee_{j=i}^n e_{(b,j)}) \leq 1$, and from line 7, 9 of Algorithm 3 we know that $p_2(x_i \vee y_i) \geq p_2(y_i)$.
- $p_2(\lambda) \in [0, 1]$.
It is true, because $p_1(e_{(a,1)})p_1(\bigvee_{i=1}^n e_{(b,i)}) \leq p_1(C)$ (from the formulae of C) and from line 10 of Algorithm 3 we know that $p_2(\lambda) \in [0, 1]$.

We illustrate Algorithm 3 by the example below.

Example 4. $A = \{t_{(a,1)}\}$ and $B = \{t_{(b,1)}, t_{(b,2)}, t_{(b,3)}\}$. $f_1(t_{(a,1)}) = e_{(a,1)}$, $f_1(t_{(b,1)}) = e_{(b,1)}$, $f_1(t_{(b,2)}) = e_{(b,2)}$, $f_1(t_{(b,3)}) = e_{(b,3)}$. The simplified REF implication constraint is $e_{(a,1)} \Rightarrow (e_{(b,1)} \vee e_{b,2} \vee e_{b,3})$.

After conditioning, $f_2(t_{(a,1)}) = \lambda$, $f_2(t_{(b,1)}) = (\lambda \wedge x_1) \vee y_1$, $f_2(t_{(b,2)}) = (\lambda \wedge \neg x_1 \wedge \neg y_1 \wedge x_2) \vee y_2$, $f_2(t_{(b,3)}) = (\lambda \wedge \neg x_1 \wedge \neg y_1 \wedge \neg x_2 \wedge \neg y_2) \vee y_3$. The probabilities of new events are $p_2(y_1) = p_1(e_{(b,1)})$, $p_2(y_2) = p_1(e_{(b,2)})$, $p_2(y_3) = p_1(e_{(b,3)})$, $p_2(x_1 \vee y_1) = \frac{p_1(e_{(b,1)})}{p_1(e_{(b,1)} \vee e_{(b,2)} \vee e_{(b,3)})}$, $p_2(x_2 \vee y_2) = \frac{p_1(e_{(b,2)})}{p_1(e_{(b,2)} \vee e_{(b,3)})}$, $p_2(\lambda) = \frac{p_1(e_{(a,1)})p_1(e_{(b,1)} \vee e_{(b,2)} \vee e_{(b,3)})}{p_1(C)}$. \square

7.2 General Case

In a REF constraint, $A = \{t_{(a,1)}, \dots, t_{(a,m)}\}$ and $B = \{t_{(b,1)}, \dots, t_{(b,n)}\}$. Assume $f_1(t_i) = e_i$. The local database is $\text{LD}(C, D) = \{t_{(a,1)}, \dots, t_{(a,m)}, t_{(b,1)}, \dots, t_{(b,n)}\}$.

The constraint is formulated as $\bigvee_{i=1}^m e_{(a,i)} \Rightarrow \bigvee_{i=1}^n e_{(b,i)}$. Its probability is computed as

$$\begin{aligned} p_1(C) &= p_1(\neg \bigvee_{i=1}^m e_{(a,i)} \vee \bigvee_{i=1}^n e_{(b,i)}) = p_1(\bigwedge_{i=1}^m \neg e_{(a,i)} \vee \bigvee_{i=1}^n e_{(b,i)}) \\ &= \prod_{i=1}^m (1 - p_1(e_{(a,i)})) + p_1(\bigvee_{i=1}^n e_{(b,i)})p_1(\bigvee_{i=1}^m e_{(a,i)}) \end{aligned}$$

The probability of such a constraint can be computed in linear time, since the probability of a disjunction of independent events can be computed associatively. Algorithm 4 presents the conditioning algorithm for simplified REF constraints. It introduces $m + 2n$ events. The time complexity of Algorithm 4 is linear to the size of the local database.

Theorem 6. *Algorithm 4 is correct and performs in linear time to size of the local database.*

We omit the proof since it is similar to the proof of previous theorems. The basic idea of the proof is to prove for every possible world, its probability is the same after conditioning as before. Moreover, we have to make sure all the probability values are valid, i.e.,

$$- p_2(x_i \vee y_i) \in [0, 1].$$

It is true for the same reason as the previous section.

$$- p_2(x_i \vee y_i) \geq p_2(y_i).$$

It is true for the same reason as the previous section.

$$- p_2(\lambda) \in [0, 1].$$

It is true, because we can deduce $p_1(\bigvee_{i=1}^n e_{(b,i)}) \leq p_1(C)$ from

$$p_1(C) = p_1(\bigvee_{i=1}^n e_{(b,i)}) + \prod_{i=1}^m (1 - p_1(e_{(a,i)})) \prod_{i=1}^n (1 - p_1(e_{(b,i)}))$$

Algorithm 4. Conditioning algorithm for REF implication constraints

Data: $\langle \text{LD}(C, D), E_1, f_1, C, p_1 \rangle$ **Result:** A world equivalent $\langle \text{LD}(C, D), E_2, f_2, \emptyset, p_2 \rangle$

```

1 foreach  $i \in [1, m]$  do
2   |  $f_2(t_{(a,i)}) \leftarrow \lambda \wedge a_i$ 
3  $f_2(t_{(b,1)}) \leftarrow (\lambda \wedge \bigvee_{j=1}^m a_j \wedge x_1) \vee y_1$ 
4 foreach  $i \in [2, n-1]$  do
5   |  $f_2(t_{(b,i)}) \leftarrow (\lambda \wedge \bigvee_{j=1}^m a_j \wedge \bigwedge_{j=1}^{i-1} (\neg x_j \wedge \neg y_j) \wedge x_i) \vee y_i;$ 
6  $f_2(t_{(b,n)}) \leftarrow (\lambda \wedge \bigvee_{j=1}^m a_j \wedge \bigwedge_{j=1}^{n-1} (\neg x_j \wedge \neg y_j)) \vee y_n;$ 
7 foreach  $i \in [1, m]$  do
8   |  $p_2(a_i) = p_1(e_{(a,i)});$ 
9 foreach  $i \in [1, n]$  do
10  |  $p_2(y_i) = p_1(e_{(b,i)});$ 
11 foreach  $i \in [1, n-1]$  do
12  |  $p_2(x_i \vee y_i) = \frac{p_1(e_{(b,i)})}{p_1(\bigvee_{j=i}^n e_{(b,j)})};$ 
13  $p_2(\lambda) = \frac{p_1(\bigvee_{i=1}^n e_{(b,i)})}{p_1(C)};$ 

```

We illustrate Algorithm 4 using the example below.

Example 5. $A = \{t_{(a,1)}, t_{(a,2)}\}$ and $B = \{t_{(b,1)}, t_{(b,2)}\}$. $f_1(t_{(a,1)}) = e_{(a,1)}$, $f_1(t_{(a,2)}) = e_{(a,2)}$, $f_1(t_{(b,1)}) = e_{(b,1)}$, $f_1(t_{(b,2)}) = e_{(b,2)}$. The REF implication constraint is $(e_{(a,1)} \vee e_{(a,2)}) \Rightarrow (e_{(b,1)} \vee e_{(b,2)})$.

After conditioning, $f_2(t_{(a,1)}) = \lambda \wedge a_1$, $f_2(t_{(a,2)}) = \lambda \wedge a_2$, $f_2(t_{(b,1)}) = (\lambda \wedge (a_1 \vee a_2) \wedge x_1) \vee y_1$, $f_2(t_{(b,2)}) = (\lambda \wedge (a_1 \vee a_2) \wedge \neg x_1 \wedge \neg y_1) \vee y_2$. The probabilities of new events are $p_2(y_1) = p_1(e_{(b,1)})$, $p_2(y_2) = p_1(e_{(b,2)})$, $p_2(a_1) = p_1(e_{(a,1)})$, $p_2(a_2) = p_1(e_{(a,2)})$, $p_2(x_1 \vee y_1) = \frac{p_1(e_{(b,1)})}{p_1(e_{(b,1)} \vee e_{(b,2)})}$, $p_2(\lambda) = \frac{p_1(e_{(b,1)} \vee e_{(b,2)})}{p_1(C)}$. \square

8 Conclusion and Future Work

In this paper, we have studied the problem of conditioning probabilistic relational data with referential constraints. We focus on the special case of probabilistic relational databases with independent events. We present and devise tractable algorithms for three classes of implication constraints, namely FKPK, FK and REF. REF is the class of general referential constraints and covers FKPK and FK constraints.

A probabilistic relational model captures only the uncertainty of data value, while probabilistic XML, as a hierarchical data model, captures the uncertainty of both value and structure. The uncertainty of the structure of data introduces new challenges for the conditioning problem. The implication constraints are also practical in probabilistic XML data, i.e., existence of a set of nodes implies existence of another set of nodes. We are now studying conditioning probabilistic XML data with implication constraints.

Acknowledgments. This paper is partially supported by the French government under the STIC-Asia program, CCIPX project. The work by Huayu Wu is supported by the A*STAR SERC Grant No. 1224200004.

References

1. Abiteboul, S., Hull, R., Vianu, V.: *Foundations of Databases*, Addison-Wesley, Boston (1995)
2. Agarwal, P., Bemjelloun, O., Sarma, A.D., Hayworth, C., Nabar, S., Sugihara, T., Widom, J.: Trio: a system for data, uncertainty, and lineage. In: *VLDB (2006)*
3. Ba, M.L., Abdessalem, T., Senellart, P.: Uncertain version control in open collaborative editing of tree-structured documents. In: *Proceedings of Document Engineering*, Florence, Italy (2013)
4. Barbará, D., Garcia-Molina, H., Porter, D.: The management of probabilistic data. *IEEE Trans. Knowl. Data Eng.* **4**(5), 482–507 (1992)
5. Cavallo, R., Pittarelli, M.: The theory of probabilistic databases. In: *Proceedings of VLDB (1987)*
6. Chang, C.-H., Kayed, H., Girgis, M.R., Shaalan, K.F.: A survey of Web information extraction systems. *IEEE Trans. Knowl. Data Eng.* **18**(10), 1411–1428 (2006)
7. Dalvi, N.N., Suciu, D.: Efficient query evaluation on probabilistic databases. *VLDB J.* **16**(4), 523–544 (2007)
8. Dey, D., Sarkar, S.: Psql: a query language for probabilistic relational data. *Data Knowl. Eng.* **28**(1), 107–120 (1998)
9. Dong, X.L., Halevy, A., Yu, C.: Data integration with uncertainty. *VLDB J.* **18**(2), 469–500 (2009)
10. Eiter, T., Lukasiewicz, T., Walter, M.: A data model and algebra for probabilistic complex values. *Proc. Ann. Math. Artif. Intell.* **33**, 205–252 (2001)
11. Fink, R., Olteanu, D., Rath, S.: Providing support for full relational algebra in probabilistic databases. In: *ICDE*, pp. 315–326 (2011)
12. Fuhr, N., Rölleke, T.: A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.* **15**(1), 32–66 (1997)
13. Green, T.J., Tannen, V.: Models for incomplete and probabilistic information. In: Grust, T., Höpfner, H., Illarramendi, A., Jablonski, S., Fischer, F., Müller, S., Patranjan, P.-L., Sattler, K.-U., Spiliopoulou, M., Wijsen, J. (eds.) *EDBT 2006. LNCS*, vol. 4254, pp. 278–296. Springer, Heidelberg (2006)
14. Koch, C., Olteanu, D.: Conditioning probabilistic databases. In: *Proceedings of PVLDB (2008)*
15. Re, C., Suciu, D.: Materialized views in probabilistic databases for information exchange and query optimization. In: *Proceedings of VLDB*, pp. 51–62 (2007)
16. Suciu, D., Olteanu, D., Ré, C., Koch, C.: *Probabilistic Databases*. Morgan & Claypool, San Rafael (2011)
17. Tang, R., Cheng, R., Wu, H., Bressan, S.: A framework for conditioning uncertain relational data. In: Liddle, S.W., Schewe, K.-D., Tjoa, A.M., Zhou, X. (eds.) *DEXA 2012, Part II. LNCS*, vol. 7447, pp. 71–87. Springer, Heidelberg (2012)
18. van Keulen, M., de Keijzer, A., Alink, W.: A probabilistic XML approach to data integration. In: *Proceedings of ICDE (2005)*

Author Index

- Abdessalem, Talel 360
Ali, Mohammed Eunus 256
Amarilli, Antoine 351
Amsterdamer, Yael 351

Ba, M. Lamine 360, 413
Baba, Yukino 376
Balke, Wolf-Tilo 376
Basak, Madhusudan 256
Berning, Tim 132
Bouguila, Nizar 18

Chang, Dong 117
Cho, Hyunsouk 376
Cui, Ge 317

Dev, Himel 256
Dong, Tingting 271

Fan, Wentao 18
Faust, Martin 132
Fort, Marta 308

Gao, Feng 388
Gao, Yang 329
Grund, Martin 132
Guan, Jihong 283
Guo, Wan 179

Hacid, Mohand-Saïd 161
Hassani, Marwan 146
Hayashi, Arata 271
He, Jiyin 403
Hong, Jihye 75
Huang, Bei 329
Huang, Chih-Yuan 342
Hwang, Seung-won 376

Ishikawa, Yoshiharu 271

Jain, Rohit 388
Jeong, Byeong-Soo 75

Khalafbeigi, Tania 342
Kim, Hyunwook 75
Kim, Jinho 89
Kunze, Kai 403

Kusmawan Putu, Y. 61
Kwon, Dae-Won 317
Kwon, Joonho 61

Lee, Suan 89
Lee, Wookey 89
Lee, Young-Koo 75
Li, Chao 296
Li, Hao 192
Li, Huiming 192
Li, Wengen 283
Li, Zhanhuai 179, 243
Liang, Steve 342
Lofi, Christoph 403
Luo, Jun 296

Maarry, Kinda El 376
Madria, Sanjay K. 403
Mesmoudi, Amin 161
Milo, Tova 351
Möller, Paul 102
Montenez, Sebastien 360
Morimoto, Yasuhiko 33
Müller, Stephan 102

Nawaz, Waqas 75

Pan, Wei 243
Park, Kisung 75
Plattner, Hasso 102, 132
Prabhakar, Sunil 388

Qin, Xiao 179

Rasyidi, Mohammad Arif 46
Ryu, Kwang Ryel 46

Schwalb, David 132
Seidl, Thomas 146
Sellarès, J. Antoni 308
Sen, Tanmoy 256
Sha, Chaofeng 217
Shao, Dongxu 413
Si, Luo 388
Siddique, Md. Anisuzzaman 33
Sigg, Stephan 403

Sugiura, Kento 271

Sun, Jing 230

Sun, Yi 217

Suo, Bo 243

Tanaka, Yuzuru 3

Tang, Ruiming 360, 413

Tian, Hao 33

Wang, Bin 230

Wang, Li 204

Wang, Mea 342

Wang, Xin 317

Wang, Zhuo 243

Wu, Hao 192

Wu, Huayu 413

Yang, Xiaochun 230

Yin, Ling 296

Yu, Chengcheng 204

Yu, Ge 117

Yuan, Peisen 217

Zhang, Lei 204

Zhang, Xiao 179

Zhang, Yanfeng 117

Zhang, Zimu 192

Zhao, Xiaonan 179

Zhao, Zhongying 296

Zhou, Aoying 204

Zhou, Qiming 296

Zhou, Shuigeng 283

Zong, Chuanyu 230