SIGMOD Officers, Committees, and Awardees

Chair
Juliana Freire
Computer Science & Engineering
New York University
Brooklyn, New York
USA
+1 646 997 4128
juliana.freire <at> nyu.edu

Vice-Chair
Ihab Francis Ilyas
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario
CANADA
+1 519 888 4567 ext. 33145
ilyas <at> uwaterloo.ca

Secretary/Treasurer
Fatma Ozcan
IBM Research
Almaden Research Center
San Jose, California
USA
+1 408 927 2737
fozcan <at> us.ibm.com

SIGMOD Executive Committee:
Juliana Freire (Chair), Ihab Francis Ilyas (Vice-Chair), Fatma Ozcan (Treasurer), K. Selçuk Candan, Yanlei Diao, Curtis Dyreson, Yannis Ioannidis, Christian Jensen, and Jan Van den Bussche.

Advisory Board:
Yannis Ioannidis (Chair), Phil Bernstein, Surajit Chaudhuri, Rakesh Agrawal, Joe Hellerstein, Mike Franklin, Laura Haas, Renee Miller, John Wilkes, Chris Olsten, AnHai Doan, Tamer Özsu, Gerhard Weikum, Stefano Ceri, Beng Chin Ooi, Timos Sellis, Sunita Sarawagi, Stratos Idreos, Tim Kraska

SIGMOD Information Director:
Curtis Dyreson, Utah State University

Associate Information Directors:
Huiping Cao, Manfred Jeusfeld, Asterios Katsifodimos, Georgia Koutrika, Wim Martens

SIGMOD Record Editor-in-Chief:
Yanlei Diao, University of Massachusetts Amherst

SIGMOD Record Associate Editors:
Vanessa Braganholo, Marco Brambilla, Chee Yong Chan, Rada Chirkova, Zachary Ives, Anastasios Kementsietsidis, Jeffrey Naughton, Frank Neven, Olga Papaemmanouil, Aditya Parameswaran, Alkis Simitsis, Wang-Chiew Tan, Nesime Tatbul, Marianne Winslett, and Jun Yang

SIGMOD Conference Coordinator:
K. Selçuk Candan, Arizona State University

PODS Executive Committee:
Jan Van den Bussche (Chair), Tova Milo, Diego Calvanse, Wang-Chiew Tan, Rick Hull, Floris Geerts

Sister Society Liaisons:
Raghu Ramakhrishnan (SIGKDD), Yannis Ioannidis (EDBT Endowment), Christian Jensen (IEEE TKDE).

Awards Committee:
Surajit Chaudhuri (Chair), David Dewitt, Martin Kersten, Maurizio Lenzerini, Jennifer Widom

Jim Gray Doctoral Dissertation Award Committee:
Ashraf Aboulnaga (co-Chair), Chris Jermaine (co-Chair), Paris Koutris, Feifei Li, Qiong Luo, Ioana Manolescu, Lucian Popa, Renée Miller

SIGMOD Systems Award Committee:
Mike Stonebraker (Chair), Make Cafarella, Mike Carey, Yanlei Diao, Paul Larson

SIGMOD Officers, Committees, and Awardees (continued)

SIGMOD Edgar F. Codd Innovations Award

For innovative and highly significant contributions of enduring value to the development, understanding, or use of database systems and databases. Recipients of the award are the following:

Michael Stonebraker (1992)
David DeWitt (1995)
Serge Abiteboul (1998)
Rudolf Bayer (2001)
Ronald Fagin (2004)
Jennifer Widom (2007)
Umeshwar Dayal (2010)
Stefano Ceri (2013)
Gerhard Weikum (2016)
Jim Gray (1993)
C. Mohan (1996)
Hector Garcia-Molina (1999)
Patricia Selinger (2002)
Michael Carey (2005)
Moshe Y. Vardi (2008)
Surajit Chaudhuri (2011)
Martin Kersten (2014)
Goetz Graefe (2017)
Philip Bernstein (1994)
David Maier (1997)
Rakesh Agrawal (2000)
Don Chamberlin (2003)
Jeffrey D. Ullman (2006)
Masaru Kitsuregawa (2009)
Bruce Lindsay (2012)
Laura Haas (2015)

SIGMOD Systems Award

For technical contributions that have had significant impact on the theory or practice of large-scale data management systems.

Michael Stonebraker and Lawrence Rowe (2015)
Martin Kersten (2016)
Richard Hipp (2017)

SIGMOD Contributions Award

For significant contributions to the field of database systems through research funding, education, and professional services. Recipients of the award are the following:

Maria Zemankova (1992)
Jeffrey Ullman (1996)
Raghu Ramakrishnan (1999)
Daniel Rosenkrantz (2001)
Surajit Chaudhuri (2004)
Hans-Jörg Schek (2007)
David Lomet (2010)
H.V. Jagadish (2013)
Samuel Madden (2016)
Gio Wiederhold (1995)
Avi Silberschatz (1997)
Michael Carey (2000)
Richard Snodgrass (2002)
Hongjun Lu (2005)
Klaus R. Dittrich (2008)
Gerhard Weikum (2011)
Kyu-Young Whang (2014)
Yannis E. Ioannidis (2017)
Yahiko Kambayashi (1995)
Won Kim (1998)
Laura Haas (2000)
Michael Ley (2003)
Tamer Özsu (2006)
Beng Chin Ooi (2009)
Marianne Winslett (2012)
Curtis Dyreson (2015)

SIGMOD Jim Gray Doctoral Dissertation Award

SIGMOD has established the annual SIGMOD Jim Gray Doctoral Dissertation Award to *recognize excellent research by doctoral candidates in the database field.* Recipients of the award are the following:

- **2006** *Winner*: Gerome Miklau. *Honorable Mentions*: Marcelo Arenas and Yanlei Diao.
- **2007** *Winner*: Boon Thau Loo. *Honorable Mentions*: Xifeng Yan and Martin Theobald.
- **2008** *Winner*: Ariel Fuxman. *Honorable Mentions*: Cong Yu and Nilesh Dalvi.
- **2009** *Winner*: Daniel Abadi. *Honorable Mentions*: Bee-Chung Chen and Ashwin Machanavajjhala.
- **2010** *Winner:* Christopher Ré. *Honorable Mentions*: Soumyadeb Mitra and Fabian Suchanek.
- **2011** *Winner*: Stratos Idreos. *Honorable Mentions*: Todd Green and Karl Schnaitterz.
- **2012** *Winner*: Ryan Johnson. *Honorable Mention*: Bogdan Alexe.
- **2013** *Winner*: Sudipto Das, *Honorable Mention*: Herodotos Herodotou and Wenchao Zhou.
- **2014** *Winners*: Aditya Parameswaran and Andy Pavlo.
- **2015** *Winner*: Alexander Thomson. *Honorable Mentions*: Marina Drosou and Karthik Ramachandra
- **2016** *Winner*: Paris Koutris. *Honorable Mentions*: Pinar Tozun and Alvin Cheung
- **2017** *Winner*: Peter Bailis. *Honorable Mention*: Immanuel Trummer

A complete list of all SIGMOD Awards is available at: **https://sigmod.org/sigmod-awards/**

[Last updated : June 30, 2017]

Editor's Notes

Welcome to the June 2017 issue of the ACM SIGMOD Record!

First of all, we welcome the new officers of the SIGMOD Executive Committee, including Juliana Freire as the Chair, Ihab Francis Ilyas as the Vice-Chair, and Fatma Ozcan as the Treasurer. The new SIGMOD Executive Committee takes office in June 2017.

The first column of this issue is the Database Principles column, featuring an article by Barceló, Pieris, and Romero on semantic optimization of Conjunctive Queries (CQs). As one of the most fundamental classes of database queries, CQs correspond to select-project-join queries in relational algebra. The database theory community has identified the classes of CQs that can be evaluated in polynomial time, i.e., the CQs that admit a suitable tree decomposition of small width. Furthermore, semantic information about the data, in the form of constraints, can be used to enrich query optimization by guiding the query transformation process. This article considers such semantic query optimization for the tractable classes of CQs and answers questions including whether a set of constraints can be used to reformulate a CQ as one of small width, and if so, what is the cost of computing and evaluating such a reformulation. As such, the article provides us with a theoretical framework for reasoning about the efficiency of such query reformulations.

The Survey column features an article by Singh and Bawa on MapReduce-based spatial query processing approaches. Support of high performance queries on spatial data has been an important topic in database research. This article presents classification and analysis of recent spatial query processing approaches, implemented in the MapReduce framework, into two categories. The first category includes hierarchical index approaches and the second category includes key-value storage based index approaches, with the main differences lying in the way that a spatial index is implemented on the partitioned dataset.

The Systems and Prototypes column features an article on the ARCHIMEDES system for efficient query processing over probabilistic knowledge bases. Due to the uncertainty of information extraction algorithms and the limitations of human knowledge, current knowledge bases are still incomplete and uncertain. The ARCHIMEDES system addresses query processing in probabilistic knowledge bases with three key technical components: (a) knowledge expansion derives implicit knowledge from knowledge bases using large rule sets; (b) query-driven inference improves inference performance by focusing MCMC on the query variables; (c) the system further leverages unified data- and graph-parallel computation to improve performance.

The Distinguished Profiles column features Beng Chin Ooi, Distinguished Professor of Computer Science at the National University of Singapore (NUS). Beng Chin is the recipient of the 2009 SIGMOD Contributions Award, and he is an IEEE and ACM Fellow and Fellow of Singapore National Academy of Science. In this interview, Beng Chin talks about his research vision (in 2011 when this interview took place), how he built up a successful research group and advised students at NUS, how he interacted with the Chinese database research community, his entrepreneur experience, and finally his work ethic and favorite pastimes.

The Reports column features a report on the third workshop on Algorithms and Systems for MapReduce and Beyond (BeyondMR'16) held in conjunction with the 2016 SIGMOD conference in San Francisco, California, USA. The goal of the workshop was to bring together researchers and practitioners to explore algorithms, computational models, architectures, languages and interfaces for systems that need large-scale parallelization and systems designed to support efficient parallelization and fault tolerance. The workshop program featured two very well attended invited talks by Ion Stoica from AMPLab, University of California Berkeley and Carlos Guestrin from the University of Washington. The program also included 5 regular and 5 short papers on specialized programming and data management systems based on MapReduce and extensions, graph processing systems, and data-intensive workflow and dataflow systems.

On behalf of the SIGMOD Record Editorial board, I hope that you enjoy reading the June 2017 issue of the SIGMOD Record!

Your submissions to the SIGMOD Record are welcome via the submission site:
http://sigmod.hosting.acm.org/record

Prior to submission, please read the Editorial Policy on the SIGMOD Record's website:
https://sigmodrecord.org

Yanlei Diao

June 2017

Past SIGMOD Record Editors:

Ioana Manolescu (2009-2013)
Alexandros Labrinidis (2007–2009)
Mario Nascimento (2005–2007)
Ling Liu (2000–2004)
Michael Franklin (1996–2000)
Jennifer Widom (1995–1996)
Arie Segev (1989–1995)
Margaret H. Dunham (1986–1988)
Jon D. Clark (1984–1985)
Thomas J. Cook (1981–1983)
Douglas S. Kerr (1976-1978)
Randall Rustin (1974-1975)
Daniel O'Connell (1971–1973)
Harrison R. Morse (1969)

Semantic Optimization in Tractable Classes of Conjunctive Queries*

Pablo Barceló
Center for Semantic Web Research &
DCC, University of Chile
pbarcelo@dcc.uchile.cl

Andreas Pieris
School of Informatics
University of Edinburgh
apieris@inf.ed.ac.uk

Miguel Romero
Center for Semantic Web Research &
DCC, University of Chile
mromero@dcc.uchile.cl

ABSTRACT

This paper reports on recent advances in semantic query optimization. We focus on the core class of conjunctive queries (CQs). Since CQ evaluation is NP-complete, a long line of research has concentrated on identifying fragments of CQs that can be efficiently evaluated. One of the most general such restrictions corresponds to bounded generalized hypertreewidth, which extends the notion of acyclicity. Here we discuss the problem of reformulating a CQ into one of bounded generalized hypertreewidth. Furthermore, we study whether knowing that such a reformulation exists alleviates the cost of CQ evaluation. In case a CQ cannot be reformulated as one of bounded generalized hypertreewidth, we discuss how it can be approximated in an optimal way. All the above issues are examined both for the constraint-free case, and the case where constraints, in fact, tuple-generating and equality-generating dependencies, are present.

1. INTRODUCTION

Conjunctive queries (CQs) are one of the most fundamental classes of database queries (see, e.g., [2, 18, 34, 42, 50]). In particular, CQs correspond to select-project-join queries in relational algebra and to select-from-where queries in SQL. However, CQ evaluation is not an easy task, especially over large volumes of data. This has led to a flurry of activity for developing heuristics that alleviate CQ evaluation in practice.

One important method of this kind is CQ optimization. Recall that query optimization is a basic database task that amounts to transforming a query into one that is more efficient to evaluate. The database theory community has developed several principled methods for optimization of CQs, many of which are based on *static-analysis* tasks such as containment [2]. In a nutshell, such methods compute a *minimal* equivalent version of a CQ, where minimality refers to the number of atoms. As argued by Abiteboul, Hull, and Vianu [2], this provides a theoretical notion of "true optimality" for the reformulation of a CQ, as opposed to practical considerations based on heuristics. For each CQ q, the minimal equivalent CQ is its *core* q' [40]. Although the static analysis tasks that support CQ minimization are NP-complete [18], this is not a major problem for most real-life applications, as the input (the CQ) is small.

It is known, on the other hand, that semantic information about the data, in the form of constraints, can be used to enrich query optimization by guiding the query transformation process. This is often referred to as *semantic query optimization* [16]. In the aforementioned analysis of CQ minimization, however, constraints play no role, as CQ equivalence is defined over *all* databases. Adding constraints yields a refined notion of CQ equivalence, which holds over those databases that satisfy a given set of constraints only. Minimization of CQs under this refined notion thus provides a principled approach to semantic query optimization [23].

An important shortcoming of the results on CQ minimization (under constraints) mentioned above is that there is no theoretical guarantee that the minimized version of a CQ is in fact easier to evaluate (recall that, in general, CQ evaluation is NP-complete [18]). We know, on the other hand, quite a bit about classes of CQs which can be evaluated in polynomial time: These are the ones that admit a suitable *tree decomposition of small width* [19, 21, 35, 38]. Our proposal is to study the fundamental problem of semantic optimization in such tractable classes of CQs; i.e., whether a set of constraints can be used to reformulate a CQ as one of small width, and if so, what is the cost of computing and evaluating such a reformulation. Following Abiteboul et al., this would provide us with a theoretical guarantee of "true efficiency" for those reformulations.

Due to its relative importance in the database literature and mature theoretical status, we concentrate on the classes of CQs of bounded generalized hypertreewidth (see [33] for a recent survey). In particular,

*Part of this work was done while Romero was visiting the Simons Institute for the Theory of Computing. Barceló and Romero are funded by the Millennium Nucleus Center for Semantic Web Research under Grant NC120004. Pieris is supported by the EPSRC Programme Grant EP/M025268/.

CQs of generalized hypertreewidth one correspond to the oldest and most studied tractability condition for CQs: *acyclicity* [50]. In terms of constraints, we consider *tuple-generating dependecies* (tgds) and *equality-generating dependencies* (egds), which subsume all real-life database constraints; in particular, tgds extend *inclusion dependencies*, while egds extend *functional dependencies*. Tgds and egds are widely applied in data integration [43], data exchange [25], and ontology-based data access [13], as a tool for expressing rich semantic constraints among different relations in the database. Due to this fact, they can be used to enhance the semantic optimization process studied here.

In this survey, we present several recent results that serve as a theoretical framework for the problem of how to obtain maximal benefit from semantic optimization in classes of CQs of bounded generalized hypertreewidth. We focus on the following questions:

1. For which classes of constraints is the reformulation problem decidable? Also, in these decidable classes, what is the complexity of the problem?

2. How semantic optimization in tractable classes of CQs can be used to tackle the ultimate problem of evaluating CQs more efficiently?

Potential impact. While most of the CQs encountered in practical situations are of low hypertreewidth [33], the work presented here is relevant due to the following:

1. Evaluating a CQ q of generalized hypertreewidth k, for $k \geq 1$, over a database $\mathcal{D}$ takes time $O(|\mathcal{D}|^{k+1} \cdot |q|)$ [35]. Even if k is small, say $k = 3$, finding an equivalent CQ q' of smaller hypertreewidth might improve the complexity of evaluation (especially when the database $\mathcal{D}$ is large).

2. Assume that q is of hypertreewidth $k > 1$. In the case that an equivalent CQ of hypertreewidth $k' < k$ cannot be found, the tools presented here provide an *approximation* of q of hypertreewidth k'. Such an approximation is a CQ q' of hypertreewidth k' that is *maximally contained* in q. That is, q' returns sound (but not necessarily complete) answers to q over those databases that satisfy the constraints, and there is no CQ q'' of hypertreewidth k' that gets "closer" to q than q'. Evaluating such an approximation q' might be convenient for obtaining quick and sound answers to q when exact evaluation is infeasible or is taking too long.

CQs of bounded hypertreewidth require specialized algorithms for their implementation, and there is currently an important body of research integrating them into some optimizers [1, 3, 5, 30, 31]. This suggests that semantic optimization techniques based on hypertreewidth have the potential to provide new practical optimization tools. A nice example is the INSIDEOUT system from LogicBlox [6], which uses related techniques based on the notion of *fractional hypertreewidth* [38] to obtain efficient bounds for CQ evaluation.

Organization. Preliminaries are in Section 2. In Section 3, we study when a CQ can be reformulated as one of bounded generalized hypertreewidth in the absence of constraints, and how such a reformulation helps query evaluation. The extension of such an investigation to the case where constraints are available is considered in the next two sections: Section 4 deals with reformulation and Section 5 with evaluation. Approximations are studied in Section 6. Final remarks are in Section 7.

2. PRELIMINARIES

Databases. A *schema* is a finite set of relation symbols, each one of which has an associated arity $n > 0$. A *database* $\mathcal{D}$ over a schema σ is a finite set of atoms of the form $R(\bar{a})$, where R is a relation symbol in σ of arity $n > 0$ and $\bar{a}$ is an n-ary tuple of constants. We write $\mathcal{D}$ also for the set of elements mentioned in $\mathcal{D}$.

Conjunctive queries. A *conjunctive query* (CQ) q over a schema σ is a rule of the form:

$$\mathsf{Ans}(\bar{x}) \leftarrow R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m), \tag{1}$$

such that (a) each $R_i(\bar{x}_i)$ is an atom over σ, for $1 \leq i \leq m$, (b) $\bar{x}$ is a sequence of variables taken from the $\bar{x}_i$'s, and (c) Ans is a distinguished relation symbol that represents the answer of q. We write $q(\bar{x})$ to denote that $\bar{x}$ is the sequence of variables that appear in such an answer.

As usual, the evaluation of a CQ q of the form (1) over a database $\mathcal{D}$ is obtained by computing the join of the atoms in the set $\{R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)\}$, and then projecting only the variables $\bar{x}$ in the answer of q. For the purposes of this paper, it is convenient to formally define this in terms of the notion of *homomorphism* from q to $\mathcal{D}$. Recall that these are the mappings h from the set of variables in q to the elements of $\mathcal{D}$ such that $R_i(h(\bar{x}_i)) \in \mathcal{D}$ for each $1 \leq i \leq m$. The *evaluation of $q(\bar{x})$ over $\mathcal{D}$*, denoted $q(\mathcal{D})$, consists then of those tuples $h(\bar{x})$ such that h is a homomorphism from q to $\mathcal{D}$.

Example 1. We describe a social network using a schema σ with two binary relation symbols: Friends and Likes. The first one establishes when two persons are friends, while the second one establishes when a person likes a post. Suppose that we want to retrieve all the pairs of mutual friends that have some post they like in common. We can express this as a CQ $q(x, y)$ defined as $\mathsf{Ans}(x, y) \leftarrow \mathsf{Friends}(x, y), \mathsf{Likes}(x, z), \mathsf{Likes}(y, z)$. □

Tractable classes of CQs. The *evaluation problem* for CQs is defined as follows: Given a CQ q, a database $\mathcal{D}$, and a tuple $\bar{t}$ of constants in $\mathcal{D}$, check if $\bar{t} \in q(\mathcal{D})$. This problem is NP-complete [18], but becomes tractable for several syntactically defined subclasses of CQs. The oldest such tractability condition corresponds to *acyclicity* [11, 32]. Intuitively, a CQ q is acyclic if its atoms can be arranged in the form of a tree while preserving the connectivity of its variables. More formally, q is acyclic if it admits a *join tree*, that is, a tree T whose nodes are the atoms of q, and for each variable x that appears in q it is the case that the set of nodes in which x is mentioned defines a connected subtree of T. We denote by AC the class of acyclic CQs. Yannakakis's seminal work established that AC defines a tractable class of CQs; in fact, the queries in this class can be evaluated in linear time $O(|\mathcal{D}| \cdot |q|)$, where $|\mathcal{D}|$ and $|q|$ are the size of the database $\mathcal{D}$ and the CQ q, respectively [50].

Example 2. The CQ in Example 1 is not acyclic. In any way we arrange its atoms as the nodes of a tree, we will lose the connectivity for at least one variable. On the other hand, the CQ $\mathsf{Ans}(x, y) \leftarrow \mathsf{Friends}(x, y), \mathsf{Likes}(x, z), \mathsf{Likes}(y, z')$, which is obtained from q by "breaking" the join on variable z, is in AC. This is witnessed by the join tree whose root is the atom $\mathsf{Friends}(x, y)$, and the atoms $\mathsf{Likes}(x, z)$ and $\mathsf{Likes}(y, z')$ are its children. Such CQ retrieves pairs of mutual friends each one of which likes some post. □

It has been observed, however, that a significant proportion of the CQs that appear in practice are not acyclic, but are in some sense *mildly acyclic* (see, e.g., [33]). This motivated the search for notions that represent the degree of acyclicity of a CQ, and for efficient evaluation algorithms for CQs with low degree of acyclicity. The degree of acyclicity of a CQ in this context is traditionally known as its *width*. Such width can be defined by using the notion of *generalized hypertree decomposition*, which extends the notion of join tree by allowing each node of the tree to be associated with several atoms of the query. The formal definition follows.

A generalized hypertree decomposition of a CQ $q := \mathsf{Ans}(\bar{x}) \leftarrow R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)$ is a tuple (T, λ, χ), where T is a tree, λ is a mapping that assigns a subset of the variables in q to each node t of T, and χ is a mapping that assigns a subset of the atoms $\{R_1(\bar{x}_1), \ldots, R_m(\bar{x}_m)\}$ to each node t of T, such that:

1. For each $1 \leq i \leq m$, the variables in $\bar{x}_i$ are contained in $\lambda(t)$, for some $t \in T$.
2. For each variable x in q, the set of nodes t of T for which x occurs in $\lambda(t)$ is connected.
3. For each $t \in T$, the variables in $\lambda(t)$ are "covered" by the atoms in $\chi(t)$; i.e., $\lambda(t) \subseteq \bigcup_{R_i(\bar{x}_i) \in \chi(t)} \bar{x}_i$.

For a generalized hypertree decomposition (T, λ, χ), its *width* is defined as the maximal size of a set of the form $\chi(t)$ over all nodes t of T. The *generalized hypertreewidth* of a CQ q is the minimum width over all generalized hypertree decompositions of q. We denote by $\mathsf{GHW}(k)$, for $k \geq 1$, the set of CQs of generalized hypertreewidth bounded by k. The notion of bounded generalized hypertreewidth subsumes acyclicity; in particular, $\mathsf{AC} = \mathsf{GHW}(1)$ [35].

Example 3. Recall that the CQ in Example 1 is not acyclic, i.e., is not in $\mathsf{GHW}(1) = \mathsf{AC}$. It is, however, in $\mathsf{GHW}(2)$. The generalized hypertree decomposition of width two that witnesses this fact has only one node t such that $\lambda(t) = \{x, y, z\}$ and $\chi(t) = \{\mathsf{Likes}(x, z), \mathsf{Likes}(y, z)\}$. □

It can be shown, by using tools based on the *existential pebble game* [20], that CQs of bounded generalized hypertreewidth can be evaluated in polynomial time.

THEOREM 1. *Fix $k \geq 1$. The evaluation problem for the CQs in $\mathsf{GHW}(k)$ can be solved in polynomial time.*

At this point, it should be stressed that in the above theorem we assume that the input query already falls in $\mathsf{GHW}(k)$, for some fixed $k \geq 1$. However, one can claim that this is not a realistic assumption. In general, the input query is an arbitrary CQ for which we do not know a priori whether it falls in $\mathsf{GHW}(k)$. In this case, we should first check whether it belongs to $\mathsf{GHW}(k)$, and, if this is the case, then proceed with the actual evaluation. This brings us to the *recognizability problem for* $\mathsf{GHW}(k)$, that is, checking if a given CQ q is in $\mathsf{GHW}(k)$. It is known that for $k = 1$ (i.e., for acyclic queries), the above problem can be solved in linear time [49]. However, for $k > 1$, it becomes NP-complete [28]. This implies that, given a CQ q, we can check in time $2^{|q|^c}$, for some integer $c \geq 1$, whether q belongs to $\mathsf{GHW}(k)$, and, if this is the case, then a generalized hypertree decomposition of q of width k is constructed. Now, having such a hypertree decomposition in place, we can evaluate q over the input database $\mathcal{D}$ in time $O(|\mathcal{D}|^{k+1} \cdot |q|)$ [35].

Summing up, there is an integer $c \geq 1$ such that, given a CQ q, a database $\mathcal{D}$, and a tuple of constants $\bar{t}$, checking if q belongs to $\mathsf{GHW}(k)$, and, if this is the case, then check if $\bar{t} \in q(\mathcal{D})$, can be carried out in time:

$$2^{|q|^c} + O(|\mathcal{D}|^{k+1} \cdot |q|).$$

The fact that we spend exponential time in the size of the query for checking whether q belongs to $\mathsf{GHW}(k)$ is not a big practical drawback since this check corresponds to a static analysis task, i.e., it only depends

on the size of the "small" CQ. For such tasks, a single-exponential time procedure is considered to be acceptable, and it is actually the norm in many cases including database and verification problems; see, e.g., [2, 46, 48].

A restriction of the notion of generalized hypertreewidth, which ensures tractability of recognizition, known as *hypertreewidth*, has been also studied in the literature [33, 35]. However, due to the nature of the problems that we consider here, it is convenient to use the less restrictive notion of generalized hypertreewidth, even at the extra cost of recognition.

CQ containment and equivalence. Two notions that are crucial for query optimization purposes are CQ containment and equivalence. Let q and q' be CQs. Then, q is *contained* in q', denoted $q \subseteq q'$, if $q(\mathcal{D}) \subseteq q'(\mathcal{D})$, for every database $\mathcal{D}$. Further, q is *equivalent* to q', denoted $q \equiv q'$, if $q \subseteq q'$ and $q' \subseteq q$ (or, equivalently, if they return the same answers over every database, i.e., $q(\mathcal{D}) = q'(\mathcal{D})$ for each database $\mathcal{D}$). Interestingly, containment is polynomially equivalent to CQ evaluation. Given a CQ q, let $\mathcal{D}_q$ be the so-called *canonical database* of q obtained from q by replacing each variable x in q with a new constant $c(x)$. Then:

PROPOSITION 2. [18] *Let $q(\bar{x}), q'(\bar{x}')$ be CQs. It holds that $q \subseteq q'$ iff $c(\bar{x}) \in q'(\mathcal{D}_q)$.*

The above proposition implies that CQ containment and equivalence are NP-complete problems [18].

The core of a CQ. In CQ minimization one is interested in finding a minimal CQ (in terms of number of joins) that is equivalent to a given CQ q. Such a minimal equivalent CQ always corresponds to a *core* of q [39]. This is a CQ q' that is obtained by deleting atoms from q and the following hold: (a) $q \equiv q'$, and (b) q is not equivalent to any CQ q'' that is obtained by removing one or more atoms from q'. In other words, a core of q is a minimally equivalent CQ that is obtained by removing atoms from q. Clearly, a core of a CQ q always exists. Moreover, all cores of q are isomorphic [39], and, therefore, we can talk about *the* core of q.

3. REFORMULATION IN THE ABSENCE OF CONSTRAINTS

It is instructive to start our investigation by focussing on semantic optimization in classes of bounded generalized hypertreewidth in the absence of constraints. We concentrate on the following problems:

- **Reformulation:** Fix $k \geq 1$. Given a CQ q, is it the case that it can be reformulated as a CQ that falls in $\mathsf{GHW}(k)$ that yields the same answers over all databases? More formally, is there a CQ q' in $\mathsf{GHW}(k)$ such that $q \equiv q'$?
- **Evaluation:** In case the latter holds, can we efficiently perform query evaluation for q? Notice that this is not a priori obvious: Although we know that the reformulation q' of q can be efficiently evaluated (since it is in $\mathsf{GHW}(k)$), the cost of computing such a reformulation might be prohibitively high.

3.1 Reformulation of CQs in $\mathsf{GHW}(k)$

We denote by $\mathsf{Equiv}(\mathsf{GHW}(k))$ the class of CQs q that are equivalent to some CQ $q' \in \mathsf{GHW}(k)$. In particular, the CQs in $\mathsf{Equiv}(\mathsf{GHW}(1))$ are known as *semantically acyclic* [7, 10]. It is easy to show that semantic acyclicity is incomparable to the notion of bounded generalized hypertreewidth, i.e., for each $k \geq 1$, there is a semantically acyclic CQ that is not in $\mathsf{GHW}(k)$.

Here we study the following problem for $k \geq 1$:

PROBLEM :	$\mathsf{Reformulation}(\mathsf{GHW}(k))$
INPUT :	A CQ q.
QUESTION :	Is q in $\mathsf{Equiv}(\mathsf{GHW}(k))$?

The main tool that we exploit in the investigation of the above problem is a simple characterization of the classes $\mathsf{Equiv}(\mathsf{GHW}(k))$, for $k \geq 1$, in terms of the core:

PROPOSITION 3 (IMPLICIT IN [10]). *Fix $k \geq 1$. For each CQ q the following are equivalent:*

- $q \in \mathsf{Equiv}(\mathsf{GHW}(k))$.
- *The core of q is in* $\mathsf{GHW}(k)$.

The upward implication is trivial: if the core q' of q is in $\mathsf{GHW}(k)$, then $q \in \mathsf{Equiv}(\mathsf{GHW}(k))$ since $q \equiv q'$ by definition. The downward implication states that if a CQ q is equivalent to some CQ $q' \in \mathsf{GHW}(k)$, then q' can always be assumed to be the core of q. The proof of this fact for the case $k = 1$ can be found in [10], but the same ideas can be used to show that this holds for any $k \geq 1$. It is worth noticing that similar techniques have also been used to prove analogous results for the notion of *bounded treewidth* [21]. This is yet another tractability-ensuring condition for CQ evaluation, which is particularly well-suited for the case when the arity of the schema is fixed [37].

Since the core of a CQ q is obtained by removing atoms from q, and such a core is equivalent to q, Proposition 3 implies the following small query property:

PROPOSITION 4. *Fix $k \geq 1$ and let q be a CQ. If $q \in \mathsf{Equiv}(\mathsf{GHW}(k))$, then there exists a CQ $q' \in \mathsf{GHW}(k)$, where $|q'| \leq |q|$, such that $q \equiv q'$.*

The above small query property allows us not only to establish that $\mathsf{Reformulation}(\mathsf{GHW}(k))$ is decidable, but also to pinpoint its exact complexity. Given a CQ q, here is a simple procedure that decides whether $q \in$

$\mathsf{Equiv}(\mathsf{GHW}(k))$: Guess a CQ q' of size at most $|q|$, and verify that (a) $q' \in \mathsf{GHW}(k)$, and (b) $q \equiv q'$. Since, as mentioned before, both (a) and (b) can be carried out in NP, we conclude that the whole procedure can be performed in NP. A matching lower bound can be proved using more elaborate techniques [21]. From the above discussion we get that:

THEOREM 5. *For each fixed $k \geq 1$, the problem* $\mathsf{Reformulation}(\mathsf{GHW}(k))$ *is* NP*-complete.*

3.2 Evaluation of CQs in $\mathsf{Equiv}(\mathsf{GHW}(k))$

Does knowing that a CQ q can be reformulated as a CQ q' in $\mathsf{GHW}(k)$ alleviate the cost of query evaluation? As for CQs in $\mathsf{GHW}(k)$, it can be shown, by exploiting techniques based on the existential pebble game [20], that CQs in $\mathsf{Equiv}(\mathsf{GHW}(k))$, for some fixed $k \geq 1$, can be evaluated in polynomial time.

THEOREM 6. *Fix $k \geq 1$. The evaluation of CQs in* $\mathsf{Equiv}(\mathsf{GHW}(k))$ *can be solved in polynomial time.*

In the previous theorem, we assume that the input query falls in $\mathsf{Equiv}(\mathsf{GHW}(k))$, for a fixed $k \geq 1$. However, as already discussed in Section 2 for $\mathsf{GHW}(k)$, this is not a realistic assumption. In a practical context, we should first check whether the input query belongs to $\mathsf{Equiv}(\mathsf{GHW}(k))$, and, if this is the case, then proceed with the actual evaluation. From Theorem 5 (and the underlying algorithm) we know that, given a CQ q, we can check in time $2^{|q|^c}$, for some integer $c \geq 1$, whether q belongs to $\mathsf{Equiv}(\mathsf{GHW}(k))$. Now, if this is the case, then a CQ q' such that $q \equiv q'$ and $|q'| \leq |q|$ that belongs to $\mathsf{GHW}(k)$, and a generalized hypertree decomposition of q' of width k are constructed; actually, q' is the core of q. Having q' and its decomposition in place, we can evaluate it over the input database $\mathcal{D}$ in time $O(|\mathcal{D}|^{k+1} \cdot |q|)$ [35]. Summing up:

COROLLARY 7. *Fix $k \geq 1$. There is an integer $c \geq 1$ such that, given a CQ q, a database $\mathcal{D}$, and a tuple of constants $\bar{t}$, checking whether q belongs to* $\mathsf{Equiv}(\mathsf{GHW}(k))$*, and, if this is the case, then check whether $\bar{t} \in q(\mathcal{D})$, can be solved in time:*

$$2^{|q|^c} \; + \; O(|\mathcal{D}|^{k+1} \cdot |q|).$$

This bound simply states that, after a preprocessing step that takes single-exponential time in the size of the CQ q to check whether q belongs to $\mathsf{Equiv}(\mathsf{GHW}(k))$, an evaluation step that takes polynomial time is performed. While the running time of the procedure is not polynomial in the combined size of the database $\mathcal{D}$ and the CQ q, it can still be considered as efficient for the reasons already explained in Section 2.

4. REFORMULATION IN THE PRESENCE OF CONSTRAINTS

As said above, in the constraint-free case, a CQ q is equivalent to one in $\mathsf{GHW}(k)$ iff its core is in $\mathsf{GHW}(k)$. Hence, the only reason why q is not in $\mathsf{GHW}(k)$ in the first place is because it has not been minimized (since CQ minimization reduces to computing the core). Thus, semantic optimization in $\mathsf{GHW}(k)$ is not really different from usual CQ minimization. The presence of constraints, on the other hand, yields a more interesting notion of semantic optimization. This is because constraints can be applied on CQs to produce $\mathsf{GHW}(k)$ reformulations of them. Let us show this via an example.

Example 4. Consider a database that stores information about customers, records, and musical styles. The relation $\mathsf{Interest}$ contains pairs (c, s) such that the customer c is interested in the style s. The relation Class contains pairs (r, s) such that the record r is of style s. Finally, the relation Owns contains a pair (c, r) when the customer c owns the record r. Consider now a CQ $q(x, y)$ defined as follows:

$$\mathsf{Ans}(x, y) \leftarrow \mathsf{Owns}(x, y), \mathsf{Class}(y, z), \mathsf{Interest}(x, z).$$

The above query asks for pairs (c, r) such that the customer c owns the record r and has expressed interest in at least one of the styles with which r is associated. It can easily be proved that q is the core of itself but it is not acyclic. Therefore, Proposition 3 implies that q is not equivalent to an acyclic CQ (without constraints).

Assume now that the record store keeps a full list of interests for its customers, based on the styles of the records each customer has bought in the past. In other words, the database satisfies the constraint τ defined as:

$$\forall x \forall y \forall z \big(\mathsf{Owns}(x, y), \mathsf{Class}(y, z) \rightarrow \mathsf{Interest}(x, z)\big).$$

Having this information in place, we can reformulate $q(x, y)$ as the following acyclic CQ $q'(x, y)$:

$$\mathsf{Ans}(x, y) \leftarrow \mathsf{Owns}(x, y), \mathsf{Class}(y, z).$$

Notice that q and q' are in fact equivalent over every database that satisfies the constraint τ. □

Before we proceed further, let us introduce the classes of constraints that we consider in this paper, and some basics on CQ equivalence under constraints.

Constraints. We consider the two most important classes of database constraints; namely:

1. *Tuple-generating dependencies* (tgds), i.e., expressions of the form $\forall \bar{x} \forall \bar{y} \big(\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})\big)$, where ϕ and ψ are conjuntions of atoms. Notice that the constraint in Example 4 is a tgd. Tgds

subsume the central class of *inclusion dependencies* (IDs) [24]. For example, assuming that R and P are binary relations, the ID $R[1] \subseteq P[2]$, which simply states that the set of values occurring in the first attribute of R is a subset of the set of values in the second attribute of P, is expressed via the tgd $\forall x \forall y (R(x, y) \rightarrow \exists z\, P(z, x))$.

2. *Equality-generating dependencies* (egds), i.e., expressions of the form $\forall \bar{x}\big(\phi(\bar{x}) \rightarrow y = z\big)$, where ϕ is a conjunction of atoms and y, z are variables in $\bar{x}$. Egds subsume the important classes of *keys* and *functional dependencies* (FDs). For example, assuming that R is a ternary relation, the FD $R : \{1\} \rightarrow \{3\}$, i.e., the first attribute of R functionally determines the third attribute of R, is expressed via the egd $\forall x \forall y \forall z \forall y' \forall z' (R(x, y, z) \wedge R(x, y', z') \rightarrow z = z')$. Notice that FDs that have more than one attribute in the right-hand side, are expressed via a *set* of egds.

A database $\mathcal{D}$ *satisfies* a tgd of the above form if the following holds: for each tuple $(\bar{a}, \bar{b})$ of elements such that all atoms in $\phi(\bar{a}, \bar{b})$ are in $\mathcal{D}$, there is a tuple $\bar{c}$ of elements such that all atoms in $\psi(\bar{a}, \bar{c})$ are in $\mathcal{D}$. Analogously, $\mathcal{D}$ satisfies an egd $\forall \bar{x}(\phi(\bar{x}) \rightarrow y = z)$ if, for each tuple $\bar{a}$ of elements such that all atoms in $\phi(\bar{a})$ are in $\mathcal{D}$, the elements in $\bar{a}$ that correspond to variables y and z are the same. Finally, $\mathcal{D}$ satisfies a set Σ of constraints if it satisfies every tgd and egd in Σ.

CQ equivalence under constraints. In semantic optimization, one uses the information provided by the constraints to replace a given CQ q by an equivalent one that is easier to evaluate. However, in this context the right notion of equivalence corresponds to the one that is measured over those databases that satisfy the constraints only (as we know that our datasets satisfy such constraints). We formalize this as follows. Let q, q' be CQs and Σ a set of constraints. Then *q is equivalent to q' under Σ*, denoted $q \equiv_\Sigma q'$, if and only if $q(\mathcal{D}) = q'(\mathcal{D})$ for each database $\mathcal{D}$ that satisfies Σ. The notion of containment is defined analogously, and we write $q \subseteq_\Sigma q'$.

If Σ consists only of egds, then deciding $q \equiv_\Sigma q'$ remains NP-complete. This is obtained by applying the well-known *chase procedure* [45] on q and q', respectively, and then checking for equivalence of the resulting queries. When applied on a CQ q, the chase procedure *fires* each egd on the canonical database $\mathcal{D}_q$ of q, equating variables if needed in order to restore consistency. The procedure finishes when the resulting database satisfies all egds in Σ.

On the other hand, the situation is more difficult if Σ consists of tgds. Although a characterization based on the chase procedure exists, the result of the chase under a set of tgds is, in general, infinite. Hence, this tool no longer provides a decision procedure. This is not surprising since checking CQ containment/equivalence under arbitrary sets of tgds is undecidable [12]. This negative result has motivated a long search for practical restrictions of the class of tgds with decidable CQ containment (and, thus, equivalence) problems. Such restrictions are often classified into four main paradigms:

1. *Full tgds:* These are tgds without existentially quantified variables, i.e., of the form $\forall \bar{x}(\phi(\bar{x}) \rightarrow \psi(\bar{x}))$. In the database literature, such sets are particularly important as they correspond to queries expressible in the widely studied *Datalog* language [2]. CQ containment is decidable for sets of full tgds since the chase always terminates.

2. *Guardedness:* A tgd is *guarded* if its body $\phi(\bar{x}, \bar{y})$ contains an atom, called the *guard*, that contains all the variables in $(\bar{x} \cup \bar{y})$. Although the chase under guarded tgds does not necessarily terminate, CQ containment is decidable in this case since the result of such a chase has finite treewidth [13]. A crucial subclass of guarded tgds is the class of *linear* tgds [14], i.e., tgds with only one atom in the body, which subsume inclusion dependencies.

3. *Non-recursiveness:* A set Σ of tgds is *non-recursive* if its predicate graph contains no directed cycles. (Non-recursive sets of tgds are also known as *acyclic* [25, 44], but we reserve this term for CQs). This class ensures the termination of the chase, and thus decidability of CQ containment.

4. *Stickiness:* The goal of stickiness is to capture joins among variables that are not expressible via guarded tgds, but without forcing the chase to terminate. The definition is based on an inductive marking procedure that marks the variables that could violate a particular semantic property of the chase [15]. Decidability of CQ containment is obtained by applying *query rewriting* techniques.

The complexity of CQ containment and equivalence varies from class to class. It is EXPTIME-complete for full tgds [22] and sticky sets of tgds [22], 2EXPTIME-complete for guarded tgds [13], PSPACE-complete for linear tgds [14, 41], and NEXPTIME-complete for non-recursive sets of tgds [44]. Fixing the schema, or even its arity, yields better complexities in most of the cases.

4.1 Reformulation with tgds

One of the main tasks of our work is to study the problem of checking if a CQ q can be reformulated as a CQ in $\mathsf{GHW}(k)$, for a fixed $k \geq 1$, over those databases that satisfy a set Σ of tgds. We formally define such a reformulation problem below. Given a set Σ of constraints, we write $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$ for the set of CQs

q for which there exists a CQ q' in $\mathsf{GHW}(k)$ such that $q \equiv_\Sigma q'$. We assume in the following that $\mathbb{C}$ is a class of sets of tgds (e.g., guarded, non-recursive, sticky, etc.):

PROBLEM :	$\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$
INPUT :	A CQ q and a finite set $\Sigma \in \mathbb{C}$.
QUESTION :	Is q in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$?

One might be tempted to think, in view of the Example 4, that when a reformulation $q' \in \mathsf{GHW}(k)$ of q under a set Σ of tgds exists, then such a q' is not very "big", or at least its size is bounded by $|q| + |\Sigma|$. This would allow us to state a small query property for the reformulation problem, and thus establish its decidability. As explained next, this is not the case since the decidability of $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ depends not only on the decidability of CQ containment under sets of tgds in $\mathbb{C}$, but also on other considerations.

Undecidable cases. Under mild syntactic assumptions on its input, $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ is as hard as CQ containment under $\mathbb{C}$, i.e, we can obtain decidability of $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ only for those $\mathbb{C}$'s for which CQ containment is decidable [7]. At this point, one might think that some version of the converse also holds, i.e., $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ is reducible to CQ containment under sets of tgds in $\mathbb{C}$. This would imply the decidability of $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ for any class $\mathbb{C}$ of sets of tgds for which CQ containment is decidable (in particular, for the full, guarded, non-recursive, and sticky sets of tgds). The next result shows that the picture is more complicated than this, as the reformulation problem is undecidable even if we focus on the class $\mathsf{AC} = \mathsf{GHW}(1)$ of acyclic CQs and the class $\mathcal{F}$ of sets of full tgds:

THEOREM 8. [7] $\mathsf{Reformulation}(\mathsf{GHW}(1), \mathcal{F})$ *is undecidable.*

The question that comes up is whether the reformulation problem is decidable for the remaining classes of tgds, i.e., guarded, non-recursive and sticky, that have a decidable CQ containment problem, and if yes, what is the exact complexity of the problem.

Decidable cases. The next proposition is the main tool that we use to show the decidability of the reformulation problem under certain classes of tgds:

PROPOSITION 9. *Fix $k \geq 1$. Let q and q' be CQs over schema σ such that $q' \in \mathsf{GHW}(k)$ and $q' \subseteq q$. There is a CQ $q'' \in \mathsf{GHW}(k)$ such that $q' \subseteq q'' \subseteq q$ and $|q''| \leq |q| \cdot (2k+1) \cdot a_\sigma$, where a_σ is the maximum arity over all predicates of σ.*

PROOF (SKETCH). Since, by hypothesis, $q'(\bar{x}') \subseteq q(\bar{x})$, it is the case from Proposition 2 that there exists a homomorphism h from q to $\mathcal{D}_{q'}$ such that $h(\bar{x}) = c(\bar{x}')$. Also, since $q' \in \mathsf{GHW}(k)$, it admits a generalized hypertree decomposition (T, λ, χ) of width k. Assume that $\alpha_1, \ldots \alpha_n$ are the atoms of q. By definition, the atoms $h(\alpha_1), \ldots, h(\alpha_n)$ are covered by some nodes $v_1, \ldots, v_m$, where $m \leq n$, of T. In other words, for each $i \in \{1, \ldots, n\}$, there exists $j \in \{1, \ldots, m\}$ such that the elements occurring in $h(\alpha_i)$ form a subset of $\lambda(v_j)$. Consider now the subtree T_q of T consisting of $v_1, \ldots, v_m$ and their ancestors. From T_q we extract the tree $F = (V, E)$ defined as follows:

- V consists of all the root and leaf nodes of T_q, and all the inner nodes of T_q with at least two children.
- For $v, u \in V$, $(v, u) \in E$ iff u is a descendant of v in T_q, and the only nodes of V that occur on the unique simple path from v to u in T_q are v and u.

Our intention is to construct the desired CQ q'' by transforming the atoms occurring in F into a CQ. Let $\mathcal{J} = \bigcup_{v \in V} \chi(v)$. Notice that F is *not* necessarily a generalized hypetree decomposition of $\mathcal{J}$. Indeed, it may be the case that there exists an atom $R(\bar{t})$ in $\mathcal{J}$, but there is no node v in F such that $\bar{t} \subseteq \lambda(v)$. Interestingly, we can transform F into a generalized hypertree decomposition (F, λ', χ') of some instance $\mathcal{J}'$, by renaming some of the terms in $\mathcal{J}$, and then exploit $\mathcal{J}'$ for constructing q''. For example, a node v in F labeled by $\lambda(v) = \{t_1, t_2\}$ and $\chi(v) = \{R(t_1, t'), P(t_2, t', t'')\}$ will be transformed into a node labeled by $\lambda'(v) = \{t_1, t_2, \#_1, \#_2\}$ and $\chi'(v) = \{R(t_1, \#_1), P(t_2, \#_1, \#_2)\}$, where $\#_1$ and $\#_2$ are fresh constants. The set $\mathcal{J}'$ is then defined as $\{h(\alpha_1), \ldots, h(\alpha_n)\} \cup \bigcup_{v \in V} \chi'(v)$.

It is not difficult to verify that (F, λ', χ') is a generalized hypertree decomposition of $\mathcal{J}'$ of width k. Moreover, by construction, F has at most $2 \cdot |q|$ nodes, and each such node is labeled (via χ') with at most k atoms. Thus, $|\mathcal{J}'| \leq 2 \cdot |q| \cdot k + |q| = |q| \cdot (2k+1)$. Therefore, by transforming $\mathcal{J}'$ into a CQ, we obtain a query in $\mathsf{GHW}(k)$ of size at most $|\mathcal{J}'| \cdot a_\sigma \leq |q| \cdot (2k+1) \cdot a_\sigma$; let $q''(\bar{x}')$ be this query. Observe that $c(\bar{x}') \in q(\mathcal{D}_{q''})$ since the homomorphism h maps q to $\mathcal{D}_{q''}$. This fact allows us to conclude that $q'' \subseteq q$ by Proposition 2. Furthermore, there exists a homomorphism, obtained by reversing the renaming substitutions applied during the construction of (F, λ', χ'), that maps q'' to $\mathcal{D}_{q'}$. This allows us to show that $c(\bar{x}') \in q''(\mathcal{D}_{q'})$, and, therefore, that $q' \subseteq q''$ from Proposition 2. Consequently, q'' is the desired CQ, and Proposition 9 follows. □

We write $\mathcal{G}$, $\mathcal{L}$, $\mathcal{NR}$ and $\mathcal{S}$ for the classes of guarded, linear, non-recursive, and sticky sets of tgds, respectively. By exploiting Proposition 9, we can establish a small query property, analogous to Proposition 4 for the constraint-free case, for all the above classes. We de-

fine, for each $\mathbb{C} \in \{\mathcal{G}, \mathcal{L}, \mathcal{NR}, \mathcal{S}\}$, a function $f_{\mathbb{C}}$ from the set of pairs (Σ, q), where $\Sigma \in \mathbb{C}$ and q is a CQ, to the natural numbers, which will be used to bound the size of the "small" query. Given a set Σ of tgds and a CQ q, let (i) $p_{\Sigma,q}$ be the number of predicates in Σ and q, (ii) $a_{\Sigma,q}$ the maximum arity over all those predicates, and (iii) b_Σ the maximum number of atoms in the body of a tgd in Σ. Then:

$$f_{\mathbb{C}}(\Sigma, q) \quad = \quad \begin{cases} |q|, & \mathbb{C} = \mathcal{G}, \\ |q|, & \mathbb{C} = \mathcal{L}, \\ |q| \cdot (b_\Sigma)^{p_{\Sigma,q}}, & \mathbb{C} = \mathcal{NR}, \\ p_{\Sigma,q} \cdot (a_{\Sigma,q} \cdot |q| + 1)^{a_{\Sigma,q}}, & \mathbb{C} = \mathcal{S}. \end{cases}$$

We can now state the following small query property:

PROPOSITION 10. *Fix $k \geq 1$. Let Σ be a finite set of tgds in $\mathbb{C} \in \{\mathcal{G}, \mathcal{L}, \mathcal{NR}, \mathcal{S}\}$ and q a CQ . If $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, then there is a CQ $q' \in \mathsf{GHW}(k)$, where $|q'| \leq f_{\mathbb{C}}(\Sigma, q) \cdot (2k+1) \cdot a_{\Sigma,q}$, such that $q \equiv_\Sigma q'$.*

PROOF (SKETCH). We first focus on the class $\mathcal{G}$; the same proof applies for linear tgds since $\mathcal{L} \subseteq \mathcal{G}$. By hypothesis, $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, which means that there exists a CQ $q'(\bar{x}')$ in $\mathsf{GHW}(k)$ such that $q \equiv_\Sigma q'$. Let q_Σ and q'_Σ, respectively, be the CQs that are obtained by applying the chase procedure over the atoms of q and q', respectively, using the tgds of Σ. Notice that such queries might contain an *infinite* number of atoms. The notion of evaluation, as well as generalized hypertree decomposition and generalized hypetreewidth, naturally extend to such infinite queries. It is well-known that $q \subseteq_\Sigma q'$ iff $q_\Sigma \subseteq q'$; see, e.g., [13].

Guarded tgds enjoy a property called *generalized hypetreewidth preserving chase*. This means that if we chase a CQ in $\mathsf{GHW}(k)$ using guarded tgds, the resulting query also falls in $\mathsf{GHW}(k)$. Therefore, $q'_\Sigma \in \mathsf{GHW}(k)$. It can easily be checked that Proposition 9 holds even if the left-hand side query q' is infinite. Since $q'_\Sigma \subseteq q$, there exists a CQ $q'' \in \mathsf{GHW}(k)$ such that $q'_\Sigma \subseteq q'' \subseteq q$ and $|q''| \leq |q| \cdot (2k+1) \cdot a_{\Sigma,q}$. Since $q'' \subseteq q$, we have that $q'' \subseteq_\Sigma q$. Moreover, $q \subseteq_\Sigma q''$. This follows from the fact that $q \subseteq_\Sigma q'$ (by hypothesis) and $q' \subseteq_\Sigma q''$ (since $q'_\Sigma \subseteq q''$). We conclude that $q \equiv_\Sigma q''$.

We now focus on non-recursive sets of tgds. It is not difficult to verify that this class does not enjoy the generalized hypetreewidth preserving chase property, and, thus, we cannot apply the same argument as for guarded tgds. However, $\mathcal{NR}$ enjoys some other crucial property, which is very useful for our purposes. Given a CQ q, and a set $\Sigma \in \mathcal{NR}$, we can construct a *union of CQs* (UCQ) Q such that: for every $q'(\bar{x}')$, it holds that $q' \subseteq_\Sigma q$ iff $c(\bar{x}') \in Q(\mathcal{D}_{q'})$. Moreover, the *height* of such a rewriting Q, that is, the maximal size of its disjuncts, is at most $f_{\mathcal{NR}}(\Sigma, q)$; for more details see [36].[1] We can now explain how Proposition 10 is obtained for $\mathcal{NR}$.

[1]Let us clarify that the work [36] does not explicitly consider the class $\mathcal{NR}$. However, the rewriting algorithm in that paper works also for non-recursive sets of tgds.

Since $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, there is a CQ $q'(\bar{x}')$ in $\mathsf{GHW}(k)$ such that $q \equiv_\Sigma q'$. As $\mathcal{NR}$ is UCQ rewritable, there is a UCQ Q such that $c(\bar{x}') \in Q(\mathcal{D}_{q'})$, i.e., there exists a CQ q_r (one of the disjuncts of Q) such that $c(\bar{x}') \in q_r(\mathcal{D}_{q'})$. Hence, $q' \subseteq q_r$ by Proposition 2. From Proposition 9, there is a CQ $q'' \in \mathsf{GHW}(k)$ such that $q' \subseteq q'' \subseteq q_r$ and $|q''| \leq |q_r| \cdot (2k+1) \cdot a_{\Sigma,q}$. Since $|q_r| \leq f_{\mathcal{NR}}(\Sigma, q)$, we have $|q''| \leq f_{\mathcal{NR}}(\Sigma, q) \cdot (2k+1) \cdot a_{\Sigma,q}$. We claim now that $q \equiv_\Sigma q''$. In fact, $q \subseteq_\Sigma q'$ by hypothesis, and thus $q \subseteq_\Sigma q''$ (since $q' \subseteq q''$). On the other hand, $q_r \subseteq_\Sigma q$ (otherwise, Q would not be a UCQ rewriting), and since $q'' \subseteq q_r$, we get that $q'' \subseteq_\Sigma q$.

Finally, for the class of sticky sets of tgds, we follow the same approach as for non-recursive sets of tgds. The class $\mathcal{S}$ is UCQ rewritable, and, given a set $\Sigma \in \mathcal{S}$ and a CQ q, the height of each UCQ rewriting of q and Σ is at most $f_{\mathcal{S}}(\Sigma, q)$; for more details see [36]. □

Since CQ containment is decidable for any class $\mathbb{C} \in \{\mathcal{G}, \mathcal{L}, \mathcal{NR}, \mathcal{S}\}$, Proposition 10 provides a decision procedure for $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$. Given a CQ q and a finite set $\Sigma \in \mathbb{C}$, this procedure is as follows:

1. Guess a CQ q' that falls in $\mathsf{GHW}(k)$ of size at most $f_{\mathbb{C}}(\Sigma, q) \cdot (2k+1) \cdot a_{\Sigma,q}$; and
2. Verify that $q \equiv_\Sigma q'$.

By exploiting known results on the complexity of CQ containment for the classes of tgds under consideration (see above), and carefully analyzing the time and space complexity of the above procedure, we obtain worst-case optimal upper bounds, apart from the case of sticky sets of tgds for which the complexity remains open. The lower bounds are inherited from CQ containment. Then:

THEOREM 11. *Fix $k \geq 1$ and a class of tgds $\mathbb{C} \in \{\mathcal{G}, \mathcal{L}, \mathcal{NR}, \mathcal{S}\}$. $\mathsf{Reformulation}(\mathsf{GHW}(k), \mathbb{C})$ is*

- *2*EXPTIME*-complete for $\mathbb{C} = \mathcal{G}$.*
- *PSPACE*-complete for $\mathbb{C} = \mathcal{L}$.*
- *NEXPTIME*-complete for $\mathbb{C} = \mathcal{NR}$.*
- *in NEXPTIME and EXPTIME-hard for $\mathbb{C} = \mathcal{S}$.*

If we assume that the underlying schema is fixed, then in all cases the complexity becomes NP-complete. Better complexity results can be obtained in the case of $\mathcal{G}$, $\mathcal{L}$ and $\mathcal{S}$ if the arity of the schema is fixed, while for $\mathcal{NR}$ it remains NEXPTIME-hard; for details see [7].

4.2 Reformulation with egds

The reformulation problem under egds is quite challenging, and not very well-understood up to date. Although the CQ containment problem under egds can easily be shown to be decidable (as said before, it is

NP-complete), currently we do not even know the decidability status of the reformulation problem under the simple class of egds that correspond to keys.

A positive, yet very challenging result in this direction has been recently obtained by Figueira [26]. It states that the reformulation problem is decidable for the class of *unary* FDs, denoted $\mathbb{UFD}$, when restricted to schemas consisting of unary and binary relations. Recall that unary FDs are FDs of the form $R : A \rightarrow B$, where the cardinality of A is one. The following holds:

THEOREM 12. [26] *Fix $k \geq 1$. Given a finite set $\Sigma \in \mathbb{UFD}$ over a schema with unary and binary relations, and a CQ q, we can decide in double-exponential time if there exists a CQ $q' \in \mathsf{GHW}(k)$ such that $q \equiv_\Sigma q'$.*

Let us clarify that in [26] the above result is shown for CQs of bounded treewidth. However, the proof adapts to CQs of bounded generalized hypetreewidth [27].

5. THE EVALUATION PROBLEM

As we observed earlier, in the absence of constraints the property of being equivalent to a CQ in $\mathsf{GHW}(k)$, for $k \geq 1$, has a positive impact on query evaluation. We observe here that, at least partially, this good behavior extends to the notion of being equivalent to a CQ in $\mathsf{GHW}(k)$, for $k \geq 1$, under the decidable classes of constraints we identified in the previous section. In particular, evaluation of CQs in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for sets Σ of constraints in such classes can be solved by a *fixed-parameter tractable* (fpt) algorithm, assuming the parameter to be $|q| + |\Sigma|$. Recall that this means that the problem can be solved in time $O(|\mathcal{D}|^c \cdot f(|q| + |\Sigma|))$, for $c \geq 1$ and $f : \mathbb{N} \rightarrow \mathbb{N}$ a computable function. This is an improvement over general CQ evaluation for which no fpt algorithm is believed to exist; see, e.g., [29, 47].

Fix $k \geq 1$ and a class $\mathbb{C}$ of constraints. We study the following problem in this section:

PROBLEM :	$\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{C})$
INPUT :	$\Sigma \in \mathbb{C}$, a CQ $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, a database $\mathcal{D}$ such that $\mathcal{D}$ satisfies Σ, and a tuple $\bar{t}$ of elements in $\mathcal{D}$.
QUESTION :	Is $\bar{t} \in q(\mathcal{D})$?

5.1 Evaluation under tgds

Recall that Theorem 6 establishes that the evaluation problem for CQs that can be reformulated in the class $\mathsf{GHW}(k)$ in the absence of constraints is feasible in polynomial time. As stated next, this good behavior extends to the class $\mathcal{G}$ of sets of guarded tgds:

PROPOSITION 13. $\mathsf{Evaluation}(\mathsf{GHW}(k), \mathcal{G})$ *is feasible in polynomial time, for each fixed $k \geq 1$.*

The proof of Proposition 13 for the case $k = 1$ can be found in [7]. A slight modification of this proof yields the result for any $k \geq 1$. We do not know if this good behavior extends to the classes $\mathcal{NR}$ and $\mathcal{S}$. We can prove, nevertheless, that the problem in question retains some good properties; in fact, it is fixed-parameter tractable under such classes:

PROPOSITION 14. *Fix $k \geq 1$ and $\mathbb{C} \in \{\mathcal{NR}, \mathcal{S}\}$.* $\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{C})$ *is fixed-parameter tractable.*

$\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{C})$, as in the constraint-free case, makes the unrealistic assumption that we know in advance that the CQ q is in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for a given set Σ of tgds in $\mathbb{C}$. To study the more realistic scenario in which we want to first check if this is the case, and then, if so, check whether $\bar{t} \in q(\mathcal{D})$, we have to return to the guess-and-check procedure from Section 4.1. This procedure checks in double-exponential time if a CQ q is in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for any set of tgds $\Sigma \in \mathbb{C}$. More importantly, in case that $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$ it also yields an equivalent CQ q' in $\mathsf{GHW}(k)$ of at most exponential size in $|q| + |\Sigma|$. We can then compute and evaluate such a query q' on $\mathcal{D}$, and return $q(\mathcal{D}) = q'(\mathcal{D})$. We know that the latter can be done in time $O(|\mathcal{D}|^{k+1} \cdot |q'|)$, which is $|\mathcal{D}|^{k+1} \cdot 2^{O(|q|+|\Sigma|)}$. Summing up:

COROLLARY 15. *Fix $k \geq 1$ and $\mathbb{C} \in \{\mathcal{G}, \mathcal{NR}, \mathcal{S}\}$. Given a CQ q, a set Σ of tgds in $\mathbb{C}$, a database $\mathcal{D}$ satisfying Σ, and a tuple $\bar{t}$ in $\mathcal{D}$, the problem of checking if q is in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, and, if this is the case, then check whether $\bar{t} \in q(\mathcal{D})$, can be solved in time:*

$$2^{2^{O(|q|+|\Sigma|)}} \; + \; |\mathcal{D}|^{k+1} \cdot 2^{O(|q|+|\Sigma|)}.$$

Notice that Proposition 14 follows directly from this result. The algorithm presented above, however, can hardly be claimed to be practical. In fact, it requires a preprocessing step for computing an equivalent reformulation of q under Σ that takes double-exponential time. Although this is a static analysis task, a double-exponential time procedure is too costly in practice even for small q and Σ. Thus, it would be useful to develop heuristics that lower the complexity of this task to at least single-exponential time. A notable exception is the class of linear tgds since, in this case, the guess-and-check algorithm from Section 4.1 takes exponential time to check if a CQ q is in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for $\Sigma \in \mathcal{L}$.

5.2 Evaluation under egds

Following the same approach as above, we can prove that $\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{UFD})$ is fixed-parameter tractable, when restricted to schemas with unary and binary relations. This is because, again, the procedure that checks reformulation for a CQ q under a set $\Sigma \in \mathbb{UFD}$,

used in the proof of Theorem 12, yields an equivalent CQ q' in $\mathsf{GHW}(k)$ in case that such a q' exists. Importantly enough, this fixed-parameter tractable algorithm works without the unrealistic assumption that q belongs to $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for the given set Σ.

Notably, it follows from techniques in [7] that fixed-parameter tractability of evaluation extends to the whole class $\mathbb{EGD}$ of sets of egds. Moreover, for the class $\mathbb{FD}$ of FDs it is even possible to obtain tractability:

PROPOSITION 16. *Fix $k \geq 1$. It holds that:*

1. $\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{EGD})$ *is fixed-parameter tractable.*
2. $\mathsf{Evaluation}(\mathsf{GHW}(k), \mathbb{FD})$ *can be solved in polynomial time.*

In contrast to the case of $\mathbb{UFD}$, though, the evaluation algorithms underlying Proposition 16 require knowing in advance that $q \in \mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, for the given set Σ of egds. However, checking whether such a promise holds for q might be an undecidable problem.

6. APPROXIMATIONS

Let $\mathbb{C}$ be any of the decidable classes of finite sets of tgds we study in this paper (i.e., $\mathcal{G}$, $\mathcal{NR}$, or $\mathcal{S}$). Then, for any CQ q and set Σ of constraints in $\mathbb{C}$, our techniques yield the *maximally contained* CQs q' in $\mathsf{GHW}(k)$ under Σ.[2] Following the recent database literature, such q's correspond to the $\mathsf{GHW}(k)$*-approximations of q under* Σ; see, e.g., [8, 9, 10]. Computing and evaluating the $\mathsf{GHW}(k)$-approximations of q might help finding "quick" (i.e., fixed-parameter tractable) answers to it when exact evaluation is infeasible.

We define the notion of $\mathsf{GHW}(k)$-approximation of q under Σ below, following the idea that such approximations correspond to its maximally contained CQs in $\mathsf{GHW}(k)$ under Σ:

Definition 1. ($\mathsf{GHW}(k)$-approximations) Fix $k \geq 1$. Let q be a CQ and Σ a finite set of tgds. A $\mathsf{GHW}(k)$*-approximation of q under* Σ is a CQ $q' \in \mathsf{GHW}(k)$ that satisfies the following two conditions:

- **Soundness:** q' only retrieves sound answers with respect to q; in other words, $q' \subseteq_\Sigma q$.
- **Maximality:** There is no CQ q'' in $\mathsf{GHW}(k)$ that approximates q better in terms of containment; i.e., for every $q'' \in \mathsf{GHW}(k)$ it is the case that:

$$q' \subseteq_\Sigma q'' \subseteq_\Sigma q \implies q' \equiv_\Sigma q''.$$

Notice that whenever q is in $\mathsf{Equiv}(\mathsf{GHW}(k))_\Sigma$, i.e., there is a CQ $q' \in \mathsf{GHW}(k)$ such that $q \equiv_\Sigma q'$, then the unique $\mathsf{GHW}(k)$-approximation of q under Σ is q' itself. That is, the notion of $\mathsf{GHW}(k)$-approximation provides a suitable extension of the notion of $\mathsf{GHW}(k)$-reformulation. We show in the following example that computing an approximation might be useful when exact reformulation is impossible.

Example 5. Recall the database given in Example 4 whose schema is $\{\mathsf{Interest}, \mathsf{Class}, \mathsf{Owns}\}$. Suppose we additionally have a relation $\mathsf{Incompatible}$ that contains pairs (s_1, s_2) whenever style s_1 is incompatible with style s_2. Consider now the query that retrieves all the customers c that own a record r from a style s in which he/she is interested, and also c has shown interest in at least two incompatible styles. This query can be expressed by the following CQ $q(x)$:

$$\begin{aligned}\mathsf{Ans}(x) \leftarrow\ & \mathsf{Owns}(x,y), \mathsf{Class}(y,z),\\ & \mathsf{Interest}(x,z), \mathsf{Interest}(x,z_1), \mathsf{Interest}(x,z_2),\\ & \mathsf{Incompatible}(z_1,z_2).\end{aligned}$$

As in Example 4, suppose that the database satisfies the constraint $\tau := \forall x \forall y \forall z \big(\mathsf{Owns}(x,y), \mathsf{Class}(y,z) \rightarrow \mathsf{Interest}(x,z)\big)$. As it turns out, q cannot be reformulated in $\mathsf{GHW}(1)$. The intuition is that, although we can remove atom $\mathsf{Interest}(x,z)$ as in Example 4, the cycle over variables $\{x, z_1, z_2\}$ is still present. Nevertheless, we can approximate q in $\mathsf{GHW}(1)$ via the CQ $q'(x)$:

$$\begin{aligned}\mathsf{Ans}(x) \leftarrow\ & \mathsf{Owns}(x,y), \mathsf{Class}(y,z),\\ & \mathsf{Interest}(x,w), \mathsf{Incompatible}(w,w).\end{aligned}$$

Note that q' is obtained by removing $\mathsf{Interest}(x,z)$ from q, and identifying the variables z_1 and z_2 with w. Interestingly, this example also shows that approximations can improve in the presence of constraints. Indeed, a possible approximation of q, ignoring τ, is $q''(x)$:

$$\begin{aligned}\mathsf{Ans}(x) \leftarrow\ & \mathsf{Owns}(x,t), \mathsf{Class}(t,t), \mathsf{Interest}(x,t),\\ & \mathsf{Interest}(x,w), \mathsf{Incompatible}(w,w).\end{aligned}$$

It is easy to verify that $q'' \subsetneq_\Sigma q'$. □

6.1 Approximations in the absence of constraints

As in the case of the reformulation problem, it is instructive to start by studying approximations in the absence of constraints. We call $\mathsf{GHW}(k)$-approximations of q under $\Sigma = \varnothing$ simply $\mathsf{GHW}(k)$-approximations of q. As shown in [8], $\mathsf{GHW}(k)$-approximations have good properties in this context that justify its application. In particular, they always exist, are of polynomial size, and can be computed in single-exponential time in the size of the CQ q. For brevity, we write $\mathsf{Approx}(q, \mathsf{GHW}(k))$ for the set of all the $\mathsf{GHW}(k)$-approximations of q (up to equivalence). The following holds:

[2] As said, the decidability of reformulation under egds is not well-understood. Thus, we concentrate on tgds in this section.

THEOREM 17. *Fix $k \geq 1$. Then:*

1. *Every CQ q has a $\mathsf{GHW}(k)$-approximation.*
2. *Given a CQ q, there is an exponential time algorithm that computes the set $\mathsf{Approx}(q, \mathsf{GHW}(k))$.*
3. *For each CQ q, each CQ in $\mathsf{Approx}(q, \mathsf{GHW}(k))$ is of polynomial size.*

The proof of the above result relies on Proposition 9. Recall that the latter proposition states that for every CQ q and CQ $q' \in \mathsf{GHW}(k)$ such that $q' \subseteq q$, we can find a CQ $q'' \in \mathsf{GHW}(k)$ that approximates q at least as well as q', i.e., $q' \subseteq q'' \subseteq q$, and its size is polynomially bounded by that of q. Let us now explain how Theorem 17 follows from Proposition 9.

First, observe that for every CQ q there is at least one q' in $\mathsf{GHW}(k)$ of polynomial size such that $q' \subseteq q$. Simply take a single variable x and add a tuple $R(x, \ldots, x)$ for each symbol R in the underlying schema σ. The resulting CQ q' is in $\mathsf{GHW}(1)$, and thus in $\mathsf{GHW}(k)$ for each $k \geq 1$. Moreover, there is a homomorphism from q to the canonical database $\mathcal{D}_{q'}$ of q': just map each variable of q to $c(x)$. Thus, $(c(x), \ldots, c(x)) \in q(\mathcal{D}_{q'})$, and hence $q' \subseteq q$ from Proposition 2. It is clear that q' is of polynomial size. Let $t_\sigma : \mathbb{N} \to \mathbb{N}$ be the polynomial such that $t_\sigma(n) = \max\{|q'|, n \cdot (2k+1) \cdot a_\sigma\}$. (Recall that a_σ is the maximum arity of a relation in σ).

Consider now the set $\mathsf{Cont}(q, \mathsf{GHW}(k))$ of CQs q' in $\mathsf{GHW}(k)$ over σ of size at most $t_\sigma(|q|)$ such that $q' \subseteq q$. From the above discussion, this set is nonempty. Let us consider the set $\mathsf{Maximal}(q, \mathsf{GHW}(k))$ consisting of the $\subseteq$-maximal elements of $\mathsf{Cont}(q, \mathsf{GHW}(k))$. We claim that $\mathsf{Maximal}(q, \mathsf{GHW}(k))$ consists of all the $\mathsf{GHW}(k)$-approximations of q (up to equivalence). We first show that each $\mathsf{GHW}(k)$-approximation q' of q is equivalent to some CQ $q'' \in \mathsf{Maximal}(q, \mathsf{GHW}(k))$. Consider such a $\mathsf{GHW}(k)$-approximation q' of q. By definition, $q' \in \mathsf{GHW}(k)$ and $q' \subseteq q$, and, thus, from Proposition 9 there is a CQ $q^* \in \mathsf{GHW}(k)$ such that $q' \subseteq q^* \subseteq q$ and the size of q^* is at most $t_\sigma(|q|)$. Therefore, $q^* \in \mathsf{Cont}(q, \mathsf{GHW}(k))$, and there is a CQ $q'' \in \mathsf{Maximal}(q, \mathsf{GHW}(k))$ such that $q' \subseteq q^* \subseteq q'' \subseteq q$. By definition, $q'' \in \mathsf{GHW}(k)$ and, thus, $q' \equiv q''$ since q' is a $\mathsf{GHW}(k)$-approximation of q. The proof that each CQ q' in $\mathsf{Maximal}(q, \mathsf{GHW}(k))$ is, in fact, a $\mathsf{GHW}(k)$-approximation of q follows a similar reasoning.

Notice that $\mathsf{Maximal}(q, \mathsf{GHW}(k))$ contains at least one CQ (since $\mathsf{Cont}(q, \mathsf{GHW}(k))$ is nonempty). Thus, each CQ q has at least one $\mathsf{GHW}(k)$-approximation. This yields item (1) of Theorem 17. For item (2), it is sufficient to observe that the set $\mathsf{Maximal}(q, \mathsf{GHW}(k))$ can be computed in single-exponential time. This is done by simply enumerating all CQs q' of size at most $t_\sigma(|q|)$, and for each one of them checking the following: (a) $q' \in \mathsf{GHW}(k)$, (b) $q' \subseteq q$, and (c) there is no $q'' \in \mathsf{GHW}(k)$ such that $q' \subsetneq q'' \subseteq q$ and the size of q'' is at most $t_\sigma(|q|)$. Each one of these steps can be carried out in single-exponential time.

Evaluation of approximations. Let us look at the problem of evaluating the $\mathsf{GHW}(k)$-approximations of q, i.e., given a CQ q, a database $\mathcal{D}$, and a tuple $\bar{t}$ in $\mathcal{D}$, checking whether $\bar{t} \in q'(\mathcal{D})$ for some $\mathsf{GHW}(k)$-approximation q' of q. Since each such a q' is contained in q, we can then be sure that $\bar{t}$ also belongs to $q(\mathcal{D})$.

As explained above, the set $\mathsf{Approx}(q, \mathsf{GHW}(k))$ of $\mathsf{GHW}(k)$-approximations of q can be computed in single-exponential time. Hence, checking if $\bar{t} \in q'(\mathcal{D})$ for some $\mathsf{GHW}(k)$-approximation q' of q can be carried out by a fixed-parameter tractable algorithm in time:

$$2^{r(|q|)} \; + \; |\mathcal{D}|^{k+1} \cdot 2^{r'(|q|)},$$

for polynomials $r, r' : \mathbb{N} \to \mathbb{N}$. Notably, unless P = NP this problem cannot be solved in polynomial time:

PROPOSITION 18. *Fix $k \geq 1$. Given a CQ q, a database $\mathcal{D}$, and a tuple $\bar{t}$ in $\mathcal{D}$, checking if $\bar{t} \in q'(\mathcal{D})$ for some $\mathsf{GHW}(k)$-approximation q' of q is* NP*-complete.*

Let us end up by explaining more in detail why it might be convenient, in some cases, to evaluate the approximations of a CQ q as as way to obtain quick answers when exact evaluation is infeasible or is taking too long. Suppose, in particular, that q cannot be reformulated as a CQ in $\mathsf{GHW}(k)$. Hence, it must be the case that $q \in \mathsf{GHW}(k')$ for some $k' > k$. Let us assume that a generalized hypertree decomposition of q of width k' is available to us. We can then use this decomposition to solve the exact evaluation problem for q over $\mathcal{D}$ in time $O(|\mathcal{D}|^{k'+1} \cdot |q|)$. Still, in the realistic case in which $\mathcal{D}$ is too large – in particular, when $2^{r(|q|)} + 2^{r'(|q|)} \ll |\mathcal{D}|$ – we have that evaluating the $\mathsf{GHW}(k)$-approximations of q over $\mathcal{D}$ in time $2^{r(|q|)} \; + \; |\mathcal{D}|^{k+1} \cdot 2^{r'(|q|)}$ can be considerably faster than evaluating q itself in time $O(|\mathcal{D}|^{k'+1} \cdot |q|)$.

Number of approximations. Theorem 17 establishes a single-exponential upper bound on the number of $\mathsf{GHW}(k)$-approximations that a CQ can have. As established next, this is optimal even for the case $k = 1$.

PROPOSITION 19. [8] *There is a family $\{q_n\}_{n \geq 1}$ of CQs such that each CQ q_n is of size at most $O(n)$ and has $\Omega(2^n)$ non-equivalent $\mathsf{GHW}(1)$-approximations.*

6.2 Approximations with tgds

Let us now study $\mathsf{GHW}(k)$-approximations under sets Σ of tgds. Our main result establishes that if Σ comes from one of the well-behaved classes of sets of tgds we study in the paper (i.e., $\mathcal{G}$, $\mathcal{NR}$, or $\mathcal{S}$), then the $\mathsf{GHW}(k)$-approximations under Σ continue to have good properties in terms of existence and computation. For brevity, we write $\mathsf{Approx}(q, \mathsf{GHW}(k), \Sigma)$ for the set

of all the $\mathsf{GHW}(k)$-approximations of q under Σ (up to equivalence). The following holds:

THEOREM 20. *Fix $k \geq 1$ and $\mathbb{C} \in \{\mathcal{G}, \mathcal{NR}, \mathcal{S}\}$:*

1. *Every CQ q has a $\mathsf{GHW}(k)$-approximation under Σ, where $\Sigma \in \mathbb{C}$.*
2. *Given a CQ q and a set $\Sigma \in \mathbb{C}$, there is a double-exponential time algorithm that computes the set $\mathsf{Approx}(q, \mathsf{GHW}(k), \Sigma)$.*
3. *For each CQ q and set $\Sigma \in \mathbb{C}$, each CQ in $\mathsf{Approx}(q, \mathsf{GHW}(k), \Sigma)$ is of exponential size.*

As for the case of Theorem 17, we prove Theorem 20 by exploiting a small query property:

PROPOSITION 21. *Fix $k \geq 1$ and assume that $\mathbb{C} \in \{\mathcal{G}, \mathcal{NR}, \mathcal{S}\}$. There is a polynomial $t : \mathbb{N} \to \mathbb{N}$ such that for each CQs q, q' and set $\Sigma \in \mathbb{C}$, if $q' \in \mathsf{GHW}(k)$ and $q' \subseteq_\Sigma q$, then there is a CQ $q'' \in \mathsf{GHW}(k)$ such that $q' \subseteq_\Sigma q'' \subseteq_\Sigma q$ and $|q''| \leq 2^{t(|q|+|\Sigma|)}$.*

The explanation of how Theorem 20 follows from Proposition 21 mimics the explanation of how Theorem 17 follows from Proposition 9. Let us note that Proposition 21 follows from the proof of Proposition 10, and moreover, we can refine the upper bound for $|q''|$ to be $f_{\mathbb{C}}(\Sigma, q) \cdot (2k+1) \cdot a_{\Sigma,q}$. Since $f_{\mathcal{G}}(\Sigma, q)$ is polynomial, we can obtain an improved version of Theorem 20, for the case of guarded tgds, stating that the approximations are of polynomial size.

The comparison of Theorem 17 and Theorem 20 shows that the addition of constraints does not come for free: (1) Computing the set of approximations under sets of tgds in $\mathbb{C}$ takes double-exponential time, as opposed to the single-exponential time procedure obtained in the absence of them. (2) Approximations in the presence of non-recursive and sticky sets of tgds can be of exponential size, while they are polynomial in their absence.

Evaluation of approximations. From Theorem 20, we obtain that evaluating $\mathsf{GHW}(k)$-approximations under Σ, where Σ is a set of tgds in $\mathbb{C}$, can be solved in time:

$$2^{2^{r(|q|+|\Sigma|)}} \; + \; |\mathcal{D}|^{k+1} \cdot 2^{2^{r'(|q|+|\Sigma|)}},$$

for suitable polynomials $r, r' : \mathbb{N} \to \mathbb{N}$. That is, this problem is fixed-parameter tractable. On the other hand, the double-exponential dependence on $|q| + |\Sigma|$ is impractical. It would be important then to develop heuristics that find at least some of these approximations in at most single-exponential time on the size of q and Σ.

7. FINAL REMARKS

We have not only surveyed, but also provided a common framework for recent results about semantic optimization in the classes $\mathsf{GHW}(k)$ – of CQs of bounded generalized hypertreewidth – under tgds or egds. Surprisingly, there are cases where CQ containment is decidable, while reformulation is undecidable. Such cases include the class of full tgds. We have then focussed on the main classes of tgds for which CQ containment is decidable, and do not subsume full tgds, i.e., guarded, non-recursive and sticky sets of tgds. For all these classes we have explained why the reformulation problem is decidable, and provided several complexity results. Regarding egds, we have presented a deep result that establishes the decidability of the reformulation problem under unary FDs over binary schemas.

We have also considered the problem of evaluating a query that can be reformulated in $\mathsf{GHW}(k)$ over a database that satisfies certain constraints. In all cases, when the refomulation problem is decidable such an evaluation problem can be solved by a fixed-parameter tractable procedure. This procedure is "realistic", as it also checks whether the query satisfies the reformulation requirements. By lifting this condition, one can further show that the aforementioned evaluation problem remains fixed-parameter tractable under any sets of egds, and even tractable for sets of guarded tgds or FDs.

Finally, we explained how the techniques developed for studying the reformulation problem also yield the $\mathsf{GHW}(k)$-approximations of a query when an exact reformulation in $\mathsf{GHW}(k)$ cannot be found. Such approximations can be used to "quickly" find sound answers to the query when its exact evaluation is infeasible.

Interestingly, all the complexity results on reformulation under tgds presented in the paper continue to hold for a more *liberal* version of reformulation under constraints that is based on unions of CQs. In such case we are given a UCQ Q and a finite set Σ of tgds, and the question is whether there is a union Q' of CQs in $\mathsf{GHW}(k)$ that is equivalent to Q under Σ. Moreover, when such a reformulation exists we obtain that evaluation, as above, is fixed-parameter tractable.

Many challenging problems remain open, the most noticeable being the decidability status of reformulation under egds/FDs. For egds, we have some indications that the problem is undecidable; in fact, that the existing proof of undecidability for the reformulation problem under full tgds can be recast in terms of egds. For FDs we have no understanding whatsoever at this stage.

So far, decidability results for reformulation have been obtained separately for tgds, on the one hand, and egds, on the other. But in practice tgds and egds often appear together. The decidability boundary for CQ containment in the presence of both types of constraints is delicate [17], but some restricted decidable instances of the problem have been identified [4]. It deserves to be explored whether such restrictions also yield decidability for the reformulation problem studied here.

8. REFERENCES

[1] Christopher R. Aberger, Susan Tu, Kunle Olukotun, and Christopher Ré. Emptyheaded: A relational engine for graph processing. In *SIGMOD*, pages 431–446, 2016.

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] F. Afrati, M. Joglekar, C. Ré, S. Salihoglu, and J. D. Ullman. GYM: A multiround join algorithm in mapreduce. CoRR, abs/1410.4156, 2014.

[4] Antoine Amarilli and Michael Benedikt. Finite open-world query answering with number restrictions. In *LICS*, pages 305–316, 2015.

[5] K. Amroun, Z. Habbas, and W. Aggoune-Mtalaa. DBToaster: Higher-order delta processing for dynamic, frequently fresh views. *VLDB*, 5(10):968–979, 2012.

[6] M. Aref, B. ten Cate, T. J. Green, B. Kimelfeld, D. Olteanu, E-Pasalic, T. L. Veldhuizen, and G. Washburn. Design and Implementation of the LogicBlox System. *SIGMOD*, pages 1371–1382, 2015.

[7] Pablo Barceló, Georg Gottlob, and Andreas Pieris. Semantic acyclicity under constraints. In *PODS*, pages 343–354, 2016.

[8] Pablo Barceló, Leonid Libkin, and Miguel Romero. Efficient approximations of conjunctive queries. *SIAM J. Comput.*, 43(3):1085–1130, 2014.

[9] Pablo Barceló, Reinhard Pichler, and Sebastian Skritek. Efficient evaluation and approximation of well-designed pattern trees. In *PODS*, pages 131–144, 2015.

[10] Pablo Barceló, Miguel Romero, and Moshe Y. Vardi. Semantic acyclicity on graph databases. In *SIAM J. Comput.*, 2016.

[11] Catriel Beeri, Ronald Fagin, David Maier, Alberto O. Mendelzon, Jeffrey D. Ullman, and Mihalis Yannakakis. Properties of acyclic database schemes. In *STOC*, pages 355–362, 1981.

[12] Catriel Beeri and Moshe Y. Vardi. The implication problem for data dependencies. In *ICALP*, pages 73–85, 1981.

[13] Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.

[14] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.

[15] Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.

[16] U. S. Chakravarthy, J. Grant, and J. Minker. Logic-based approach to semantic query optimization. *ACM Trans. Database Syst.*, 15(2):162–207, 1990.

[17] A. K. Chandra and M. Y. Vardi. The implication problem for functional and inclusion dependencies. *SIAM J. of Comput.*, 14:671–677, 1985.

[18] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, pages 77–90, 1977.

[19] Chandra Chekuri and Anand Rajaraman. Conjunctive query containment revisited. *TCS*, 239(2):211–229, 2000.

[20] Hubie Chen and Víctor Dalmau. Beyond hypertree width: Decomposition methods without decompositions. In *CP*, pages 167–181, 2005.

[21] Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *CP*, pages 310–326, 2002.

[22] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.

[23] Alin Deutsch, Lucian Popa, and Val Tannen. Query reformulation with constraints. *SIGMOD Record*, 35(1):65–73, 2006.

[24] Ronald Fagin. A normal form for relational databases that is based on domains and keys. *ACM Trans. Database Syst.*, 6(3):387–415, 1981.

[25] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[26] Diego Figueira. Semantically acyclic conjunctive queries under functional dependencies. In *LICS*, pages 847–856, 2016.

[27] Diego Figueira, 2017. Personal communication.

[28] Wolfgang Fischl, Georg Gottlob, and Reinhard Pichler. General and fractional hypertree decompositions: Hard and easy cases. *CoRR*, abs/1611.01090, 2016.

[29] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.

[30] L. Ghionna, L. Granata, G. Greco, and F. Scarcello. Hypertree decompositions for query optimization. In *ICDE*, pages 36–45, 2007.

[31] L. Ghionna, L., G. Greco, and F. Scarcello. H-DB: A hybrid quantitative-structural SQL optimizer. In *CIKM*, pages 2573–2576, 2011.

[32] Nathan Goodman and Oded Shmueli. Tree queries: A simple class of relational queries. *ACM Trans. Database Syst.*, 7(4):653–677, 1982.

[33] Georg Gottlob, Gianluigi Greco, Nicola Leone, and Francesco Scarcello. Hypertree decompositions: Questions and answers. In *PODS*, pages 57–74, 2016.

[34] Georg Gottlob, Nicola Leone, and Francesco Scarcello. The complexity of acyclic conjunctive queries. *J. ACM*, 48(3):431–498, 2001.

[35] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *J. Comput. Syst. Sci.*, 64(3):579–627, 2002.

[36] Georg Gottlob, Giorgio Orsi, and Andreas Pieris. Query rewriting and optimization for ontological databases. *ACM Trans. Database Syst.*, 2014.

[37] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1), 2007.

[38] Martin Grohe and Dániel Marx. Constraint solving via fractional edge covers. *ACM Trans. Algorithms*, 11(1):4:1–4:20, 2014.

[39] Pavol Hell and Jaroslav Nešetřil. The core of a graph. *Discrete Mathematics*, 109:117–126, 1992.

[40] Pavol Hell and Jaroslav Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[41] David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189, 1984.

[42] Paris C. Kanellakis. Elements of relational database theory. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 1073–1156. 1990.

[43] Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[44] Thomas Lukasiewicz, Maria Vanina Martinez, Andreas Pieris, and Gerardo I. Simari. From classical to consistent query answering under existential rules. In *AAAI*, pages 1546–1552, 2015.

[45] David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, 1979.

[46] S. Malik and L. Zhang. Boolean satisfiability: from theoretical hardness to practical success. *CACM*, 52(68):76–82, 2009.

[47] Christos H. Papadimitriou and Mihalis Yannakakis. On the complexity of database queries. *J. Comput. Syst. Sci.*, 58(3):407–427, 1999.

[48] A. Robinson and A. Voronkov. *Handbook of Automated Reasoning*. The MIT Press, 2001.

[49] Robert Endre Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3):566–579, 1984.

[50] Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, pages 82–94, 1981.

A Survey of Traditional and MapReduce-Based Spatial Query Processing Approaches

Hari Singh*
Computer Science and Engineering Department
N.C. College of Engineering
Israna, Panipat, Haryana, India
harirawat@rediffmail.com

Seema Bawa
Computer Science and Engineering Department
Thapar University
Patiala, Punjab, India
seema@thapar.edu

ABSTRACT

Various indexing methods of spatial data have come out after rigorous efforts put by many researchers for fast processing of spatial queries. Parallelizing spatial index building and query processing have become very popular for improving efficiency. The MapReduce framework provides a modern way of parallel processing. A MapReduce-based works for spatial queries consider the existing traditional spatial indexing for building spatial indexes in parallel. The majority of the spatial indexes implemented in MapReduce use R-Tree and its variants. Therefore, R-Tree and its variant-based traditional spatial indexes are thoroughly surveyed in the paper. The objective is to search for still less explored spatial indexing approaches, having the potential for parallelism in MapReduce. The review work also provides a detailed survey of MapReduce-based spatial query processing approaches - hierarchical indexed and packed key-value storage based spatial dataset. Both approaches use different data partitioning strategies for distributing data among cluster nodes and managing the partitioned dataset through different indexing. Finally, a number of parameters are selected for comparison and analysis of all the existing approaches in the literature.

Keywords

Spatial, Index, MapReduce, R-Tree

1. INTRODUCTION AND MOTIVATION

Support of high performance queries on spatial data has become important due to the large volume, high computational complexity of spatial data, and considerable time taken by complex spatial queries [70]. The representation of semi-structured spatial data in Well-Known-Text (WKT) and Well-Known-Binary (WKB) spatial data storage format, specified by the Open Geospatial Consortium (OGC) [4], makes it interoperable. Distributed spatial database systems offer a variety of spatial query functions and indexes for fast data retrieval but lacks in scalability [28]. Limitation of spatial databases for fixed schema and strict database norms does not make these suitable for handling big spatial data [63].

The distributed computing technology has witnessed high scalability and an excellent performance through its computational power. It has encouraged the evolution of modern parallel processing frameworks, such as the MapReduce-based Hadoop [1], HBase [2, 71], Cassandra [43] and BigTable [19]. The pros and cons of these frameworks are discussed in [9, 32, 37, 45]. These frameworks provide an excellent scope for scalability and high performance computational power over traditional stand-alone systems for processing a large amount of data [3, 30, 31, 56]. Recently, MapReduce parallel frameworks have been extensively used for dealing with semi-structured spatial data and related queries efficiently [8, 66, 72]. CG_Hadoop contains a suite of MapReduce algorithms for various computational geometry problems for dealing with large scale spatial data [21]. However, the key-value storage based techniques do not process spatial queries efficiently, as these require exhaustive searching. These techniques are able to scale, but cannot handle multi-dimensional spatial data [68]. Due to limitations of spatial databases and the key-value storage based distributed systems, integration of well known spatial indexing methods on MapReduce has evolved for improving the data access.

The research in the field of spatial index construction and spatial query has always been inspired by minimizing index construction and query execution time. Top-down or bottom-up approaches for well-known datasets, also known as batch-oriented methods [12, 39, 40, 44, 46, 59], over the slow and incremental methods [11, 14, 15, 29, 35, 41, 62] are the result of such motivation. Parallel processing techniques for the bulk loading spatial index and spatial query execution has continued this research trend. A single processor-multiple disk system [35], multiple processors-multiple disk system [55] and, now, a MapReduce-based systems [7, 8, 17, 18, 27, 47, 49, 65, 66, 68, 69, 70, 72] are the results of such researches.

Recently, a lot of work has been done for indexing spatial data and implementing spatial queries for fast data retrieval. Many algorithms, discussed in Section 2, are available for the same. The MapReduce framework for parallel processing is proven handy for operations requiring intense computing and improved execution time to a considerable extent. Through this survey, it has been found that in the last few years the MapReduce framework has been exploited in the field of spatial data. The Section 3 discusses research works, for parallelizing existing traditional spatial indexes, for speeding up spatial query execution. Probably, the most relevant review of the present survey work

*Hari Singh has been presently working as a faculty in Computer Science & Engineering Department at Panipat Institute of Engineering & Technology, Panipat, Haryana, India.

is the survey of a large-scale analytical query processing in MapReduce [20]. It has provided a very good classification of existing approaches for optimizing the performance of MapReduce and analyzed join queries in MapReduce. However, in the present paper, spatial data-oriented data access methods have been surveyed to 1) analyze the existing non-disjoint decomposition methods for bulk-loading spatial indexes, which forms an integral part for processing spatial queries based on spatial index methodology, and 2) analyze the work done so far in the domain of spatial query processing on MapReduce.

The rest of the survey is organized as follows. Section 2 reviews the existing traditional indexing approaches for spatial data. Section 3 presents classification and details of recent spatial query processing approaches, implemented in MapReduce, into two categories. The first category discusses a hierarchical indexed approaches on spatial dataset and the second category discusses key-value storage based indexed approaches. The two categories differ in the way spatial index is implemented on the partitioned dataset. The approaches in both categories mainly differ in the spatial data partitioning strategies on cluster nodes. Section 4 presents summary of the paper.

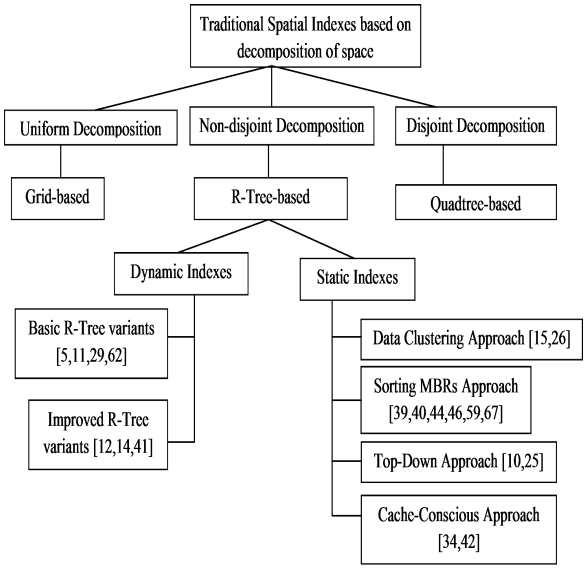


Figure 1: Traditional Spatial Indexing Approaches

2. TRADITIONAL SPATIAL INDEXING APPROACHES

In this section, existing spatial indexes, for non-disjoint decomposition, in the serial programming environment, are discussed. These are categorized according to the approach used for building an R-Tree and its variants, as shown in Fig. 1. The intent is to identify a good spatial index, having the potential for parallelization. Various dynamic and static indexing techniques, in a serial programmed environment are discussed in Section 2.1 and 2.2, respectively. The R-Tree and its variant indexes have been explored thoroughly with regard to parameters, such as space utilization, insertion cost, spatial query performance for uniformly and non-uniformly distributed data, number of nodes to be searched for spatial query, applicability in high dimensions and worst-case performance. A summary of traditional spatial indexes describing support for various functionalities is presented in Table 1.

2.1 Dynamic Indexes

The dynamic index is built at run-time for dynamic data. The techniques mentioned in this section for spatial index structures, mainly work on one or a combination of more factors, such as coverage, margin and overlap, for creating index structure. Firstly, we discuss the basic R-Tree [5, 29], R*-Tree [11], and R+-Tree[62] and secondly, various improvements over these basic tree structures for optimizing the parameters and its effect on query execution [12, 14, 41].

2.1.1 The Basic R-Tree Variants

The index of a dynamic R-Tree provides a high load time, sub-optimal space utilization, and a poor R-Tree structure [5, 29]. The index takes a large search time due to high overlapping of rectangles. The R-Tree optimization metrics require enclosing rectangle to be of larger size and contains the maximum number of data rectangles as per the node capacity. This causes assignment of a large number of entries to a node, and consequently, a high overlap among nodes. The query performance of R-Tree deteriorates with skewed data, as it causes increased overlapping. The directory rectangles from the early-inserted data rectangles may not efficiently represent the current data. Various node splitting and re-insertion methods provide solutions to the problem that distinguishes dynamic variants of the R-Tree.

R*-Tree optimizes coverage, margin and overlap of enclosing rectangles in internal nodes for data insertion and node splitting [11]. It is due to the split and forced reinsert algorithms of the R*-Tree that preserves the proximity of smaller rectangles in a node. The R*-Tree has a better retrieval performance due to a better tree structure. A better structure of R*-Tree than the R-Tree causes the insertion cost of R*-Tree comparable to R-Tree for uniformly distributed data, but much better for skewed data. The execution time of the spatial-join queries improves for R*-Tree on processor time and Input/Output (I/O) [16]. The R+-Tree provides a zero overlap among intermediate nodes through a disjoint decomposition [62]. The search performance of R+-Tree, in terms of disk accesses, is more than 50% than R-Tree for point queries, but space consumption of R+-Tree structure is more than the R*-Tree due to disjoint search space.

2.1.2 The Improved R-Tree Variants

The improved R-Tree variants work towards a better node-split for minimizing overlap among partitioned MBRs. Node splitting algorithm in RR*-Tree considers, a degree of the balance of a split and the perimeter based strategy, apart from the criteria, coverage, margin and overlap considered in R*-Tree [12]. The RR*-Tree shows a better performance than R-Tree variants due to its overlap optimization at all directory levels. It becomes better for high dimensional space due to a good balance maintaining splitting algorithm and perimeter based optimization. However, the overlap and perimeter based optimization is more compute-intensive for insertions. The WeR-Tree achieves better space utilization

Table 1: A Summary of Traditional Spatial Indexes in Serial Programming Environment

Approach	Types	Index	A	B	C1	C2	C3	C4	C5	D	E	F	G
Dynamic indexes	Basic R-Tree variants	R-Tree [5, 29]	✓	✓	✓	✓	✓	✓	X	✓	✓	✓	X
		R*-Tree [11]	✓	✓	✓	✓	✓	✓	X	✓	✓	✓	X
		R+-Tree [62]	✓	✓	✓	✓	X	X	X	✓	✓	✓	X
	Improved R-Tree variants	Revised R*-Tree [12]	✓	✓	✓	X	X	X	X	✓	✓	✓	X
		WeR-Tree [14]	✓	✓	✓	✓	X	X	✓	✓	✓	✓	X
		X-Tree [41]	✓	X	✓	✓	X	X	X	X	✓	✓	X
Static indexes	Data clustering approach	cR-Tree using k-means clustering [15]	✓	✓	✓	X	X	✓	✓	✓	✓	✓	X
		R-Tree through iterative optimization [26]	✓	✓	✓	X	X	X	X	✓	✓	✓	X
	Sorting MBRs approach	Hilbert R-Tree [39, 40]	✓	✓	✓	✓	X	X	X	✓	✓	✓	X
		STR R-Tree [46]	✓	X	✓	✓	X	X	X	✓	✓	✓	X
		Lowx R-Tree [59]	✓	X	✓	✓	X	X	X	✓	✓	✓	X
		Hilbert-curve on a tree structure [44]	X	✓	X	✓	X	X	X	X	✓	✓	X
		MR-Tree [67]	✓	X	✓	✓	X	X	X	✓	✓	✓	X
	Top-down approach	TGS R-Tree [25]	✓	✓	✓	✓	X	X	X	✓	✓	✓	✓
		Priority R-Tree [10]	✓	✓	✓	✓	X	X	X	✓	✓	✓	✓
	Cache-conscious approach	CR-Tree [42]	✓	✓	✓	✓	X	X	X	✓	✓	✓	X
		CR-Tree variant [34]	✓	✓	✓	✓	X	X	X	✓	✓	✓	X

✓- Support for functionality exists and X - Support for functionality does not exist
A-Efficient storage utilization, B-Reducing insertion cost, Spatial query performance for uniformly distributed data: C1-Query Rectangle/ Enclosure query, C2-Point query, C3-Intersection query, C4-Spatial-join query, C5-Nearest-neighbor query, D-Effect of data skewness-a kind of non-uniform data distribution, E-Number of nodes to be searched for spatial query, F-Applicability in high dimension, and G–Worst-case performance

and search performance than the R*-Tree [14]. It uses a packing technique to organize its structure better than R*-Tree, however, it takes a significant amount of time to bulk-load and reconstructing a sub-tree of unbalanced node.

The packing causes data points to be stored uniformly in leaf nodes that lead to fewer activated paths for queries. The insertion strategy searches for an unbalanced node location to insert a new entry in the existing R-Tree and partially builds a sub-tree there by keeping nodes in balance. The R*-Tree splits nodes for minimizing the volume of the resulting MBRs and thus causes more overlap in high dimensions and reduces the efficiency of an index structure. The X-Tree introduced overlap-free split policy and high page capacity nodes, named Super-nodes [41]. The overlap-free node split along a particular axis uses split history for data insertion. The Super-nodes handle unbalancing of node-fill caused due to overlap-free split and store more entries as compared to simple nodes to provide more storage utilization.

2.2 Static Indexes

Index building with dynamic insertion algorithms provide a significant dead space in nodes and results in bad performance. R-Tree variants do not exploit known dataset during insertion. However, if R-Tree is built statically, then, space utilization improves, as heuristic pack the input data space. This section categorizes different static indexes for spatial data on the basis of heuristic packing used for building index and discusses the effect of packing spatial datasets for constructing R-Tree and variants [10, 15, 25, 26, 39, 40, 44, 46, 59, 67], as well as spatial query performance.

2.2.1 Data Clustering Approach

The clustering technique splits spatial objects in the nodes on the basis of spatial proximity according to some parameter to minimize data access time. The K-means clustering technique is used for constructing the cR-Tree [15]. The k-means algorithm is order independent, unlike linear split heuristic of R-Tree and, time and space complexity of K-means is analogous to linear split algorithm of R-Tree. It uses multi-way split procedure than the traditional two-way split procedure for realizing an efficient R-Tree. The low index building time of cR-Tree is due to significant time saving on following a simple insertion algorithm as compared to the one used in R*-Tree. In another D-dimensional and batch oriented packing, the dimensional sort curve builds R-Tree by partitioning the D-dimensional data space into K partitions such that the volume of all the enclosing rectangles is minimized [26]. Though the linear packing methods are fast, but the D-dimensional approach better packs the data. It takes into account positions and spatial extents of objects in all dimensions that are achieved by a linear method. The batch oriented methods follow a bottom-up approach level by level, and consequently, achieves a high degree of parallelism. The method is poor, as it uses a large number of disk accesses and the efficiency deteriorates with data skew and dimensionality. The clustering method for constructing R-Tree in high dimensions is compute intensive as compared to R*-Tree and Hilbert R-Tree, but it performs better than the two on query execution time. It is because the latter two assign rectangles from different clusters to the same R-Tree node [15, 26].

2.2.2 Sorting MBRs Approach

One class of R-Tree indexes is bulk-loaded by sorting the MBRs either along one dimension or both dimensions in a two dimensional space. The tree leaves are filled-up first and, then, the rest of the index is built step-by-step in a bottom-up manner [39, 40, 44, 46, 59, 67]. In one such approach, the correspondence between points on space-filling curve, Hilbert-curve, and their sequence numbers are expressed as a tree structure [44]. It provides an overlap free tree node structure for the Hilbert-curve of a particular order. However, it is impractical to store the mapping of space filling curves to a tree representation explicitly and the traversal from the root to a leaf takes excessive node accesses.

In another approach, the Lowx R-Tree, a packed R-Tree for static environment, provides a simple method of packing spatial data using a dimension sort curve [59]. It sorts the rectangles with their x or y coordinates of one of the corners of the rectangle. It provides thin, long bounding rectangles along one dimension that results in nodes having less area but large perimeter. It performs well for point queries, but not so well for larger queries, such as region queries. The performance of queries decreases for skewed data. The solution to the problem was obtained by applying sorting and partitioning step for each of the dimensions [46]. The authors presented a Sort-Tile-Recursive (STR) packing algorithm to improve load time, space utilization and data retrieval efficiency of R-Tree. In another two-tier index MR-Tree, a combination of grid index and STR R-Tree index, two disk accesses take the search to a local STR R-Tree [67]. It reduces the search space and the number of node accesses in the MR index. However, the MR index is inferior to STR index in terms of spatial efficiency of the index. It is due to the low spatial efficiency of the grid index. The I/O cost of MR-Tree is lower as compared to STR-Tree for similar reason.

The Hilbert-curve based packing shows higher performance for uniformly and skewed data by minimizing area and perimeter of R-Tree leaf nodes [39, 40]. A slight variation of it sort MBRs on the basis of the Hilbert value of the center of rectangles for constructing R-Tree [40]. The nodes of the tree, put similar MBRs together and minimize the area and perimeter of MBRs under one node and achieve high space utilization. This approach brings proximity to the data objects in R-Tree nodes, and consequently, provides more space utilization by reducing the perimeter and area of the nodes. The Hilbert R-Tree performs better for all types of data than R*-Tree in terms of the number of node accesses. It achieves a high space utilization but the insertion time is comparable to R*-Tree due to ordering of data according to Hilbert-curve. The STR and Lowx R-Tree are better than Hilbert-curve based R-Tree for uniformly distributed points and region data [46, 59]. It is because the indexing methods based on space-filling curves (SFC) for R-Tree construction do not preserve spatial locality well and produce approximate results. For the same reason, STR-based R-Tree is much better than Hilbert-curve based R-Tree for skewed data for point and region queries. However, Lowx-based R-Tree performs poorly because of poor packing of data.

2.2.3 Top-Down Approach

One class of R-Tree indexes is bulk-loaded in a top-down manner. The R-Tree index is constructed in two steps: firstly, a good partition of the data is generated recursively, and secondly, the index is built from root to leaf. A Top-down Greedy Split (TGS) algorithm divides input dataset into two subsets through a recursive split procedure and constructs R-Tree in a top-down manner [25]. The split applies heuristic, such that the cost of some objective function on MBRs of each split subset is minimized and each subset has sufficient number of rectangles, so that resulting sub-trees are packed. The bulk-loading in TGS R-Tree requires more I/Os as compared to other R-Tree variants, since it scans all the rectangles to make the partition decision. However, TGS R-Tree performs comparable to Hilbert R-Tree and STR R-Tree on uniformly distributed data, and outperforms the latter two for large rectangles and skewed data, for point and range query. In another top-down approach, a Priority R-Tree is built from the priority leaves, that contain extreme rectangles along each dimension of the dataset, and the rest of the rectangle is further divided into two subsets of approximately equal size, and pseudo PR-Tree is constructed recursively [10]. The PR-Tree bulk-loading algorithm executes a window query in the optimal number of $O((N/B)^{1-1/d}+T/B)$ I/Os in the worst case as compared to other R-Tree bulk loading methods, where N is the total number of dataset rectangles in the R-Tree, B is the block size of the disk, and T is the output size. The PR-Tree outperforms all the others for window query on highly skewed data. The bulk-loading time of PR-Tree is more than the TGS R-Tree. However, the window query performance of PR-Tree is slightly better than the TGS R-Tree.

2.2.4 Cache-Conscious Approach

One class of R-Tree index has focused on cache-conscious indexes, similar to the cache-conscious B+-Tree [58], to optimize R-Trees [34, 42]. A cache-conscious version of R-Tree, CR-Tree, uses compressed MBR keys as indexed keys to obtain a wider and smaller R-Tree [42]. The compression is done with a Quantized Relative Minimum Bounding Rectangle (QRMBR) technique and the output is quantized. The QRMBR compresses the MBR keys by representing a child MBR relatively to its parent MBR. The compression and quantization technique used has the drawback that the false hits increase. However, selecting a proper quantization level, false hits are reduced. The authors found that in two, three and four dimensions, the number of node accesses for CR-Tree is smaller than R-Tree and the performance of CR-Tree improves with increasing node size. The number of cache misses is also smaller for CR-Tree in comparison to R-Tree in all dimensions. However, the cache miss graph initially decreases for a rise in node size in certain node size, and thereafter, the graph declines with a rise in node size. The cause of such a shape is the increased overhead due to a large node size that costs more than the gain obtained due to wider and smaller R-Tree. The solution to the problem was proposed by reducing the amount of L2 cache misses in the cache-conscious QRMBR R-Tree variants for better memory utilization and improved query performance [34]. The authors introduced Optimistic Latch Free Index Traversal (OLFIT) technique to overcome the cache miss problem of conventional index concurrency control by using a version and a latch in each node.

3. SPATIAL QUERY PROCESSING APPROACHES IN MAPREDUCE

In the past, parallelization of bulk-loading spatial indexes and spatial querying is achieved through one processor in communication with multiple disk architecture [35] and a shared-nothing architecture [55]. The MapReduce programming model offers a new distributed environment for parallel processing that provides high efficiency for executing tasks [8, 66, 72]. However, the MapReduce framework incurs high data transfer overhead, which need to be dealt carefully [8, 47, 49, 53]. A lot of work is found in literature that considers different methods of spatial query processing from the serial programming model and revising these in distributed environments, especially in MapReduce environment, for parallelization. The performance of spatial queries is found better over the indexed dataset as compared to the default hashing-based key-value storage in the Hadoop [2, 16, 50, 60]. In this section, spatial query processing approaches in MapReduce, based on a hierarchically indexed (Section 3.1) and packed key-value storage based (Section 3.2) approaches, have been surveyed. Both approaches use different spatial data partitioning methods, as shown in Figure 2, and then organizing spatial index on the partitioned dataset. The hierarchically indexed dataset uses uniform data partitioning [6, 22, 23, 24], random-sampling-based data partitioning [6, 22, 23, 24], clustering-based data partitioning [17], space-filling-curve based data partitioning such as Hilbert-curve based [47, 49, 66, 68], STR packing based [47], Z-curve based [17, 18] and X-mean algorithm based [17], Quadtree-based spatial data partitioning, such as Quadtree-based recursive tile partitioning for R*-Tree indexing [7, 8], a quadtree partitioning and Hilbert-curve based local indexing [72], quadtree-based data partitioning for implementing the PR-Quadtree based local index in MapReduce[38]. The key-value storage basde approaches are based on uniform data partitioning [27], SFC-based space partitioning [65, 69, 70] and spatio-temporal partitioning [52].

A survey of the two approaches for various functionalities is presented in Table 2. It describes whether the research works under the approaches provides support for functionalities. The functionalities considered under the survey are spatial proximity of distributed spatial data on cluster nodes, index build-time, efficiency of query execution, load balancing, data transmission overhead through network, applicability in high dimensions, latency for random access for large number of concurrent reads, latency for sequential access for large number of concurrent reads, effect of cluster scaling on query execution, effect of index-node size, effect of packet-data size and performance for non-uniformly distributes dataset.

3.1 Spatial Query Processing on Hierarchically Indexed Spatial Dataset

MapReduce speeds-up bulk-loading of spatial index and query execution. The benefits of building spatial index, such as R-Tree in MapReduce framework is that MapReduce abstracts data load-balancing, process scheduling and fault tolerance from the application logic, and manages transparently [18]. Otherwise, a lot of difficulty was involved in managing these distributed computing aspects earlier [55, 61].

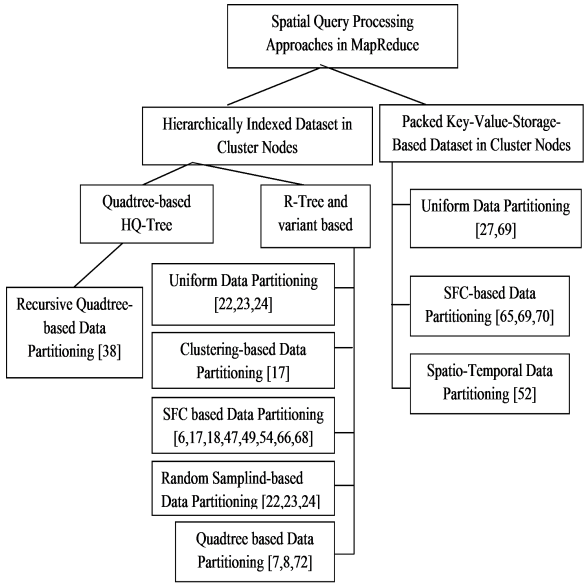


Figure 2: Spatial Query Processing Approaches in MapReduce

The MapReduce framework is also enhanced due to use of spatial indexes on MapReduce, as these improve latency for random reads [72].

This section discusses spatial query processing on the basis of various hierarchical indexes on spatial dataset. The hierarchical indexes mainly differ in data partitioning strategies and building spatial indexes on the partitioned dataset.

3.1.1 Uniform Data Partitioning

The method divides input spatial space into equal sized rectangles depending on number of mappers in MapReduce. The spatial data is partitioned into n rectangles, and the data that overlap in rectangles is redundantly assigned to overlapping rectangles. The number and size of rectangles are decided by the number of partitions required for input data. Each partitioned data are taken by a cluster node and processed there. For a grid index, the input space is partitioned in $\sqrt{n}$ x $\sqrt{n}$ rectangles of uniform size and the number of partitions (n) is calculated by dividing the input data size with HDFS block size [22, 23, 24].

Then, each slave node builds a local index in memory and writes it to disk. Lastly, a global index is built by master node. The index building time is very small due to the simple process and computations involved. The uniform or rectilinear space partitioning approach is easy to implement, but it causes non-uniform data distribution among cluster nodes for processing in MapReduce, especially for non-uniformly distributed and skewed dataset. This affects load balancing and hence efficiency of queries. Some of the nodes complete their task early and sit idle, waiting for other heavily loaded nodes to complete their tasks. The efficiency of spatial queries is not very good, as the data are unorganized and it takes a lot of time to search query data. The method works fine for uniformly distributed data, but performs poorly for non-uniformly distributed and skewed data.

Table 2: A Summary of Hierarchically Indexed and Packed Key-Value Storage Based Spatial Dataset in MapReduce

Approach	Data partitioning	Index realized on MapReduce	A	B	C1	C2	C3	C4	D	E	F	G	H	I	J	K	L
Hierarchically indexed	Uniform and Random-sampling-based [22, 23, 24]	Grid index	X	X	X	X	✓	X	X	✓	X	X	X	✓	X	X	X
	Clustering-based (x-mean) [17]	R-Tree	✓	✓	X	X	X	X	X	X	X	X	X	X	X	X	✓
	SFC-based (Z-curve) [17]	R-Tree	✓	✓	X	X	✓	X	X	X	X	X	X	X	X	X	X
	SFC-based (Z-curve) [18]	R-Tree	X	✓	X	X	X	X	X	X	X	X	X	✓	X	X	X
	SFC-based (Hilbert-curve) [47]	R-Tree on STR packing	✓	X	✓	X	✓	X	X	✓	✓	✓	✓	X	✓	X	X
	SFC-based (Hilbert-curve) [49]	R-Tree	✓	X	✓	X	X	X	✓	✓	X	X	X	X	X	X	X
	SFC-based (Hilbert-curve) [66]	R-Tree	✓	✓	✓	X	X	X	X	X	X	X	X	X	X	X	X
	SFC-based (Hilbert-curve) [68]	R-Tree	✓	✓	X	X	X	X	X	X	X	X	X	X	X	X	X
	Quadtree-based recursive tile partitioning [7, 8]	R*-Tree	X	✓	✓	✓	✓	X	✓	✓	X	✓	X	✓	X	X	X
	Quadtree-based [72]	R-Tree	✓	✓	✓	✓	✓	X	X	X	✓	✓	✓	✓	X	X	X
	Quadtree-based [38]	PR-Quadtree	X	✓	✓	X	X	X	✓	✓	X	✓	X	✓	✓	X	✓
Based on packed key-value storage	Uniform [27] (uses Controlled-Replicate approach)	Default key-value pair	X	X	X	X	✓	X	✓	✓	X	✓	X	X	X	X	X
	Uniform [69] (uses H-BRJ and H-BNLJ) and SFC-based (uses H-zkNNJ)	Default key-value pair	X	X	X	X	✓	X	X	✓	X	✓	X	✓	X	X	X
	SFC-based [65] (uses PBSM)	Default key-value pair	✓	X	X	X	✓	X	✓	✓	X	✓	X	✓	X	X	X
	SFC-based [70] (SJMR)	Default key-value pair	X	X	X	X	✓	X	✓	X	X	✓	X	✓	X	X	X
	Hybrid: spatio-temporal [52]	PMI- and OMI-based key-value pair	✓	✓	✓	✓	✓	✓	✓	X	X	X	X	✓	X	X	X

✓- Support for functionality exists and X - Support for functionality does not exist
A-Evaluating spatial proximity of distributed spatial data on cluster nodes, B-Index build-time, Efficiency of query execution: C1-Spatial Selection(Point, Line, Window and Range search query), C2-Spatial Aggregation, C3-Spatial-join, C4-Spatio-temporal, D-Load balancing: data distribution among cluster nodes, E-Data transmission overhead through network, F-Applicability in high dimensions, G-Latency of random access for large number of concurrent reads, H-Latency of sequential access for large number of concurrent reads, I-Effect of cluster scaling on query execution, J-Effect of index-node size, K-Effect of packet-data size, and L-Performance for non-uniformly distributed dataset

3.1.2 *Clustering-Based Data Partitioning*

The method partitions spatial objects into groups according to their spatial clustering. A comparison between the Z-curve based data partitioning and the X-means clustering-based data partitioning has been done for parallel R-Tree construction [17]. It is observed that the Z-curve has a linear complexity of the mappers input and generates almost equal sized partitions, but the spatial locality is not always well preserved. The X-means based iterative clustering algorithm uses Bayesian Information Criteria (BIC) score to rank clusters according to the Gaussian distribution. In this scheme, though the number of partitions is estimated but the size of partition varies considerably and iterations cause expensive computations. For nearest-neighbor queries, the X-means better approximates spatial data distribution and reduces overlapping as compared to the Z-curve which directly relates to data retrieval efficiency. But, the X-means takes almost double time for R-Tree index creation than the Z-order and significant time is elapsed in the clustering phase.

3.1.3 *Random-Sampling-Based Data Partitioning*

In this approach, random sampling of spatial data is distributed among clustered nodes. The SpatialHadoop frame-

work uses it for implementing R-Tree and R+-Tree [23, 24]. The bulk-loading of indexes is done using an STR packing technique. Partitioning the data of input file is guided by the boundaries of the leaf node. The index building time of R-Tree and R+-Tree is more as compared to the grid index due to the complexity of index building process and computations involved [22]. However, the efficiency of spatial query is very good, as data is indexed and query data search time is low. The method works fine for uniformly distributed data, but does not work that well for non-uniformly distributed and skewed data.

3.1.4 SFC-Based Data Partitioning

The space-filling curve is used to transform multi-dimensional location information into one-dimensional space. A Z-curve based uniform data partitioning is used for data partitioning during the map-phase [17, 18, 49]. In MD-HBase, a scalable multi-dimensional data store on HBase, a similar approach is used to distribute the data on cluster nodes. Multidimensional index structures, K-d Tree and Quadtree, are implemented on the partitioned dataset for demonstrating the scalability and efficiency of range and kNN queries [54].

A similar SFC-based approach for bulk-loading R-Tree on MapReduce uses the Hilbert-curve [66, 68]. The partitioning function puts objects in same partition to keep spatial proximity by using the sorted MBR values of object nodes from the Hilbert-curve and transforms to a standard and proven multi-dimensional index structure, R-Tree, through parallelizarion in MapReduce. In another SFC-based approach, parallel-gopt (p-gopt), a parallel R-Tree is built on the SFC and gopt-partitioned dataset [6]. The leaf nodes of R-Tree are filled-up in order to minimize a cost function named gopt-loading, rather than filling-up nodes to the maximum. Initially, the input dataset is sorted, in parallel, according to a space-filling curve. The sorted sequence is partitioned into sub-sequences according to gopt-partitioning method. The method makes sub-sequences of sizes between b (lower limit) and B (upper limit) according to a cost function, where each sub-sequence corresponds to a leaf node. The bulk-loading time of R-Tree using the p-gopt partitioning is more than the other parallel R-Tree construction approaches in MapReduce. However, the method outperforms other parallel R-Trees for average spatial queries in terms of node accesses.

The packing algorithms, such as STR and Hilbert packing guarantee the proximity of spatial data in R-Tree leaf nodes to reduce query response time and data transfer overhead, through network [47]. The buffer management and R-Tree node size further improves query efficiency. The buffer management speeds-up data access by keeping less space occupying internal nodes in the buffer to 1) minimize the disk access 2) avoid the bottleneck caused in case of concurrent access. A limited number of leaf nodes are permitted in the buffer, depending on space availability to further reduce the disk access time. A large index node size reduces data transfer overhead for two reasons. 1) High I/O costs of loading data from HDFS than local storage device 2) High cost of random reads than sequential reads in HDFS. The cost paid for low data transfer overhead and improved I/O is increased CPU effort for filtering more data objects. However, the space-filling curve based data partitioning approaches lose on preserving spatial locality due to a mapping from the higher dimensions to one-dimensional space, but it works better in high dimensions.

3.1.5 Quadtree-Based Data Partitioning

Quadtree-based data partitioning preserves spatial locality of objects and provides a uniform recursive decomposition of space into partitions until the number of objects in a partition are not more than a defined limit. The approach is highly suitable for parallel processing, but it is difficult to apply in high dimensions. Though, the performance of Quadtree-based indexes for index building and query processing is well established [13, 33, 36, 48]. It is due to the regular disjoint decomposition approach of Quadtree-based indexes which takes less index building and query processing time, as compared to non-disjoint and irregular disjoint decomposition approach, in R-Tree and variants. However, Quadtree-based approaches incur high data transfer and I/O costs [64].

One class of MapReduce-based approaches, for constructing R-Tree, uses a Quadtree-based space partitioning [7, 8, 72]. The VegaGiStore consists of a Quadtree-based global index and Hilbert-curve local index [72]. The former index finds data blocks and the latter locates spatial objects for efficient data retrieval with low latency access. It was observed that Quadtree-based regular disjoint decomposition technique, for spatial data partitioning, gives a stable performance for increasing k in kNN queries as compared to the other key-value storage systems, such as Hadoop, Cassandra, HBase, and the traditional spatial databases such as PostGIS, Oracle Spatial, etc. In another two-tier indexed approach, a global partition indexing for regions and local spatial indexing for objects in tiles, is used [7, 8]. The bulk-loading of spatial index is performed on each dataset by using the R*-Tree. It uses a recursive partitioning approach and multiple-assignment approach for load-balancing and boundary object problems, respectively. The indexing improves latency time of random read queries. The performance of spatial queries improve with the scalability of cluster, but it causes a high intermediate data transfer overhead. However, the proposed approaches have not considered the effect of index-node size and data-packet size.

A different technique, HQ-Tree, uses a recursive regular quadtree partitioning for handling point data [38]. It is a MapReduce implementation of PR-Quadtree index. It is free from order of data insertion and space overlap due to disjoint decomposition spatial occupancy approach. The efficiency of index creation in MapReduce environment is found better for both uniform and non-uniform data than over the standalone machine. It is good at dealing with skewed data of point objects for search queries. However, the storage of index is high due to disjoint storage of objects. The HQ-Tree approach has not been compared with other MapReduce-based non-disjoint decomposition approaches, such as R-Tree and variants. The approach is limited to spatial point objects and can be extended to other spatial data, such as lines, rectangles, and polygons in spatial data. The authors found an increase in read-time with increasing index-node size due to the rise of communication overhead with increasing node size. The read-time increases drasti-

cally when the size of index node becomes greater than the size of HDFS data packet i.e. 64 KB.

3.2 Spatial Query Processing on Packed Key-Value Storage Based Spatial Dataset

Packed key-value storage based indexes in MapReduce do not build a hierarchical index on partitioned dataset. These uses key-value pair on a partitioned dataset in the MapReduce framework as an index for spatial query processing. The hierarchical tree structures, such as R-Tree and its variants, are good for queries that access only a particular part of the dataset, such as range and region search. However, for complex spatial queries that require reading the dataset in a linear fashion, the packed key-value storage data performs better under conditions, such as the characteristic of data distribution. Various approaches that deal with complex spatial queries use different clustering methods, such as uniform data partitioning [27, 69], space-filling curve [65, 69, 70] and spatio-temporal [52], for packing input spatial objects.

3.2.1 Uniform Data Partitioning

Similar to the uniform data partitioning approach of hierarchical indexed spatial dataset, there are many techniques in the domain of packed key-value storage index which are based on uniform data partitioning. The Hadoop-Block R-Tree Join (H-BRJ) builds a parallel R-Tree index on one of the dataset for executing kNN query. It uses a uniform sized partitioning for distributing input data over the cluster nodes and builds an R-Tree index there. Similarly, a Hadoop-Block Nested Loop Join (H-BNLJ) approach does not use indexing on any dataset and use a nested loop for kNN join. Here also, the uniform data partitioning shows similar characteristics, such as ease of implementation and non-uniform distribution among cluster nodes that subsequently leads to poor load balancing and efficiency of queries [69].

In an advancement over the uniform data partitioning, the efficiency of spatial-join query is observed to improve drastically when undesired data, duplicate data and data that does not form a part of the query space, are eliminated [27]. It uses a Controlled-Replicate framework for running multi-way spatial-join, that controls the replication of rectangles and, avoids unnecessary replication and processing, and hence, reduces both communication I/O costs.

3.2.2 SFC-Based Data Partitioning

The space-filling curve based data partitioning approach in MapReduce arranges original input spatial dataset according to a SFC and partitions input spatial dataset into blocks of uniform size. A key-value storage index is applied to the packed partitioned dataset for spatial queries. A Z-value based SFC is used in MapReduce for handling kNN query (H-zkNNJ) [69]. The method reduces excessive communication and computation cost incurred by H-BNLJ and H-BRJ. The Z-curve based partitioning approximates the solution and requires only linear number of reducers. The advantage of this method is a linear communication and computation cost as compared to quadratic costs involved with baseline methods H-BNLJ and H-BRJ that use a quadratic number of reducers for kNN-join. The cost paid for lower computation and communication in H-zkNNJ is in terms of accuracy of query results, as Z-order based SFCs do not well preserve the spatial locality. The Z-order based H-zkNNJ performs better than R-Tree based H-BRJ for index building and querying with increasing number of reducers. It is because the size of data blocks decreases and a large number of smaller R-Trees are constructed in parallel that consequently increase building costs.

A double-transformation technique, PBSM [56], is implemented in a parallel programming environment in MapReduce [65, 70]. A pending file structure and redundant partition method are used to reduce communication overhead and to deal with the boundary objects problem, in MapReduce [65]. The authors observed that the quantity of buckets and tiles, tile coding method, and tile-to-bucket mapping strategy affect performance. Therefore, two SFCs, Z-curve and Hilbert-curve were used. The Z-curve used for tile coding provides weak position consistency, but the convenience of implementation. The Hilbert-curve provides a better position consistency, but needed intense computation. The two-dimensional plane sweeping technique lowers computation cost in the absence of an index, to accelerate computations. The approach performs better for the ANN query for parallel spatial databases, such as Oracle Spatial, and query performance improves with scalability.

The SJMR technique partitions dataset with disjoint partitions evenly at map function with a Z-curve tile coding method and a round-robin tile to partition mapping method [70]. The SJMR approach uses a duplication avoidance strategy, named reference tile method, to avoid replication overhead increased by spatial objects. It is present in tiles from multiple partitions by replicating these in all partitions. The Z-curve tile coding method in combination with a round-robin mapping scheme works as a spatial partitioning function. The reference tile method returns result pair for common smallest tile falling inside current partition and strip of two records. A strip-based plane sweeping method produces a superset of spatial-join result through overlapped MBRs. The performance of SJMR increases with an increase in the number of strips in the plane sweeping algorithm and with a number of nodes in the cluster. The SJMR performs better than the Parallel PBSM [65]. The SJMR carries out partitioning of the dataset and, then, elimination of duplicates in the map phase before a spatial-join is performed by a reduce task. A single MapReduce task carries out spatial-join, while the Parallel-PBSM uses two MapReduce tasks for executing spatial-join. Firstly, a map task performs data partitioning and a reduce task computes spatial-join. Secondly, the next MapReduce task eliminates duplication. The performance of both increases with increase in reduce task number up to a level, however, beyond it, the performance of both methods deteriorates as reduce task is not able to complete in one cycle.

3.2.3 Spatio-Temporal Data Partitioning

The spatio-temporal data represent spatial objects with respect to time, such as the trajectory of a moving object. It is represented as (x,y,t), where x and y are coordinates of an object and t represents a timestamp of an object at a specified position. A framework is described for query processing in sequential trajectory data of moving objects based

on MapReduce [52]. The MapReduce framework is not suitable for handling continuously changing trajectory data as frequent updates are inefficient and costs too much in a cluster. A data partitioning strategy is also not applicable for maintaining continuity of trajectories.

The main problems are management of frequent updates to a trajectory data due to mobility of objects, data partitioning of skewed data and online query processing. The first problem is solved by maintaining new updated data in main memory at each node and writing to disk in batches when a particular size of the data is accumulated. The second problem of data partitioning is solved with a hybrid partitioning method. Some static and dynamic spatio-temporal space partitioning strategies are suitable for uniformly distributed and skewed data respectively, generated by a small number of moving objects. However, for massive moving objects, a highly skewed trajectory data consists of historic- static data and the new updated data. A hybrid method provides a solution by using individual static partitioning strategy for each time period. The key-value store in MapReduce is rearranged over partitioned dataset on cluster nodes to optimize query processing of range queries and trajectory based queries through Partition based Multilevel Index (PMI) and Object Inverted Index (OII). A good load balance, scalability of data importing, an index creation and query processing are achieved with a partitioning strategy with increasing number of computing nodes. However, the hierarchical tree structures avoid the exhaustive search over a provided input dataset for point query, range query and nearest-neighbor query.

4. SUMMARY

After comparing the surveyed approaches by means of classification criteria, some peculiar issues have been revealed which are thought to be relevant with respect to efficient spatial query processing. In the paper, these issues have been discussed with an intention to reflect its potential for further research. Many R-Tree variant spatial indexes for efficient spatial data handling exist, but not all have been used in the MapReduce framework. Basic R-Tree and its variant spatial indexes have been implemented extensively in MapReduce, as can be seen from the Table 2. However, many other existing spatial indexing techniques in a sequential programming environment, surveyed in Section 2 and summarized in Table 1 which perform better than the basic R-Tree variants, have not been implemented in MapReduce. It is learnt from Section 2 that approaches such as improved R-Tree variants, data clustering, sorting MBRs, top-down and cache-conscious approaches are superior to basic R-Tree variants. However, a lot of research work in implementing spatial indexes in MapReduce, presented in Section 3, has implemented basic R-Tree variants. Spatial indexes from other approaches have been rarely implemented. Presenting all spatial indexing approaches in detail in Section 2, motivates for their implementation in MapReduce.

The improved R-Tree variants are better than basic R-Tree variants [12, 14, 41]. The storage utilization, insertion cost and window query time of Revised R*-Tree is better than R*-Tree [12]. The applicability of Revised R*-Tree in high dimensions is more than the basic R-Tree variants [12]. Similarly, the storage utilization, insertion cost, performance of spatial queries such as window query, point query and nearest-neighbor query, of WeR-Tree is better than R*-Tree [14]. The applicability in high dimensions and spatial query performance for skewed data is also better than R*-Tree [14]. In a similar way, the X-Tree has proven better than R*-Tree for storage utilization, spatial queries such as window query and point query, and applicability in high dimensions [41].

The data clustering approach are better than basic R-Tree variants [15, 26]. The storage utilization, insertion cost, spatial query performance of queries, such as window query, point query and nearest-neighbor query of the cR-Tree, and performance of spatial queries for skewed data are better than R*-Tree [15]. However, the applicability of the index is low and comparable to the basic R-Tree variants [15]. The insertion cost and window query efficiency of another clustering approach, R-Tree using iterative optimization, is higher than R*-Tree [26]. However, the performance of the R-Tree using iterative optimization for window query is low and comparable to basic R-Tree variants when spatial dataset is skewed or is in high dimensions [26].

The sorting MBRs approach is better than basic R-Tree variants [39, 40, 44, 46, 59, 67]. The insertion cost, point query efficiency and applicability of expressing Hilbert-curve and their sequence numbers in a tree structure is better as compared to the R-Tree [44]. The storage utilization and, efficiency of the window and point query of uniformly distributed and skewed data, of Hilbert R-Tree is significantly higher as compared to R*-Tree. However, the insertion cost is comparable to R*-Tree and the applicability in high dimensions is low as compared to R*-Tree [39, 40]. The storage utilization, query efficiency and applicability in high dimensions, of Lowx R-Tree is better than R*-Tree but low as compared to Hilbert R-Tree [59]. The storage utilization and, performance of window and point query, of STR R-Tree is even better than Hilbert R-Tree. However, the query performance of skewed data and high dimensional data is comparable and lower than Hilbert R-Tree, respectively [46]. The storage utilization and insertion cost of MR-Tree is better than Hilbert R-Tree, but lower than STR R-Tree. The window and point query efficiency of MR R-Tree is better than STR R-Tree, however, the query performance of MR R-Tree for skewed data remains comparable to Hilbert R-Tree and the applicability in high dimensions is even lower than Hilbert R-Tree [67].

The performance of top-down approach, TGS R-Tree and Priority R-Tree, is better than all other approaches for all parameters [10, 25]. The insertion cost of the TGS R-Tree is even better than Priority R-Tree, however, the query performance of Priority R-Tree becomes better for skewed data. Both approaches have very good applicability in high dimensions and best worst-case performance. The cache-conscious approach shows better performance for all parameters as compared to basic R-Tree variants, comparable performance as compared to improved R-Tree variants, data clustering approach and sorting MBRs approach, and lower performance than top-down approaches [34, 42].

A significantly better bulk-loading and query execution time for all MapReduce-based spatial indexing approaches than traditional serial programming environment is strongly ev-

ident from the survey work carried out in Section 3. A comparison of the existing hierarchical indexed and packed key-value storage spatial index implementations in MapReduce, as presented in Table 2, for parameters, spatial proximity of distributed data on cluster nodes, index-build time, efficiency of query execution, load balancing, data transmission overhead through the network, and applicability in high dimensions, latency for random and sequential access for a large number of concurrent reads, effect of cluster scaling on query execution, effect of index-node size, effect of packet-data size and performance for non-uniformly distributed spatial dataset have been done. The former approach is better for random access spatial queries, such as search queries, while the latter approach is better for sequential access spatial queries, such as spatial-join queries. The spatio-temporal, spatial index is useful for queries, such as tracking mobile objects with respect to time. Among the hierarchical indexed dataset, the uniform data partitioning based grid index shows poor performance on all parameters, however, it is quite strong towards applicability in high dimensions.

In-depth analysis of spatial indexes in MapReduce, presented in Section 3, is not available as compared to the traditional serial programming environment, presented in Section 2. The comparison of spatial index implemented in MapReduce has been done with parallel spatial databases only, however, comparison with other spatial indexes implemented in MapReduce is rarely available. The performance of disjoint decomposition based indexes, Quadtree-based indexes, for index building and query processing is well established [13, 33, 36, 48]. The Quadtree-based approaches provide better spatial proximity and data distribution, efficiency for search queries and low network transfer overhead as compared to other data partitioning approaches, however, their storage requirement is more which is evident from their high index building time [7, 8, 72]. It has been analyzed from the survey that not much work has been done on implementing Quadtree indexes and Quadtree-based data partitioning in MapReduce. The quadtree partitioning based spatial indexes are the best, but these are very poor for the index building time and applicability in high dimensions [7, 8, 72].

In the survey, it has been seen that query processing is highly dependent on the size and nature of the dataset, and indexes show varying performance with different type of dataset. Besides it, the explored indexes in MapReduce have not been deeply analyzed for uniformly distributed, non-uniformly distributed and skewed data of varying sizes [6, 7, 8, 17, 18, 47, 49, 54, 66, 68, 72]. It has been observed that some of the existing implementations on MapReduce have not considered the effect of index-node size [7, 8, 17, 18, 22, 23, 24, 47, 49, 54, 66, 68, 72] and communication overhead [6, 17, 18, 22, 23, 24, 49, 54, 66, 68, 72]. The high communication overhead in MapReduce is due to its run-time scheduling scheme and the pull model that interfere the efficiency for queries [37, 57]. One method to reduce the communication overhead for the intermediate data generated during query processing in the Hadoop system is implemented in [51]. The authors used an independent distributed file system Parallel Secondo File System (PSFS) that avoids the transform and transfer of intermediate data through HDFS, and transfers data among database engines directly.

5. REFERENCES

[1] Hadoop. In *http://hadoop.apache.org*.

[2] HBase. In *http://hbase.apache.org*.

[3] OGC. In *http://www.opengis.orgltechno*.

[4] Performance Measurement of a Hadoop Cluster. In *http://www.acma.com/acma/pdfs /AMAX Emulex Hadoop Whitepaper.pdf*.

[5] R-Tree. In *http://en.wikipedia.org/wiki/R-tree*.

[6] D. Achakeev, M. Seidemann, M. Schmidt, and B. Seeger. Sort-Based Parallel Loading of R-Trees. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pages 62–70, 2012.

[7] A. Aji and F. Wang. High Performance Spatial Query Processing for Large Scale Scientific Data. In *Proceedings of the SIGMODPODS PhD Symposium*, pages 9–14, 2012.

[8] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. Saltz. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce. *Proceedings of the VLDB Endowment*, 6(11):1009–1020, 2013.

[9] R. M. Arasanal and D. U. Rumani. Improving MapReduce Performance through Complexity and Performance Based Data Placement in Heterogeneous Hadoop Clusters. In *Proceedings of the International Conference Distributed Computing and Internet Technology*, pages 115–125, 2013.

[10] L. Arge, M. de Berg, H. J. Haverkort, and K. Yi. The Priority R-Tree: A Practically Efficient and Worst-Case Optimal R-Tree. *ACM Transactions on Algorithms*, 4(1), 2008.

[11] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. *ACM SIGMOD Record*, 19(2):322–331, 1990.

[12] N. Beckmann and B. Seegar. A Revised R*-Tree in Comparison with Related Index Structures. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 799–812, 2009.

[13] S. Blanas, J. M. Patel, V. Ercegovac, J. Rao, E. J. Shekita, and Y. Tian. A Comparison of Join Algorithms for Log Processing in MapReduce. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 975–986, 2010.

[14] P. Bozanis and P. Foteinos. WeR-Trees. *Data and Knowledge Engineering*, 63(2):397–413, 2007.

[15] S. Brakatsoulas, D. Pfoser, and Y. Theodoridis. Revisiting R-Tree Construction Principles. In *Proceedings of the 6th Springer East European Conference on Advances in Databases and Information System*, pages 149–162, 2002.

[16] T. Brinkhoff, H.-P. Kriegel, and B. Seeger. Efficient Processing of Spatial Joins Using R-Trees. *ACM SIGMOD Record*, 22(2):237–246, 1993.

[17] A. Cary, Y. Yesha, M. Adjouadi, and N. Rishe. Leveraging Cloud Computing in Geodatabase Management. In *Proceedings of the IEEE International Conference on Granular Computing*, pages 73–78, 2010.

[18] A. Cary, Zhengguo, V. Hristidis, and N. Rishe. Experiences on Processing Spatial Data with MapReduce. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, pages 302–319, 2009.

[19] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable-A Distributed Storage System for Structured Data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):1–26, 2008.

[20] C. Doulkeridis and K. Norvag. A Survey of Large-Scale Analytical Query Processing in MapReduce. *VLDB Journal*, 23(3):355–380, 2013.

[21] A. Eldawy, Y. Li, M. F. Mokbel, and R. Janardan. CG_Hadoop: Computational Geometry in MapReduce. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 294–303, 2013.

[22] A. Eldawy and M. F. Mokbel. SpatialHadoop. In *http://spatialhadoop.cs.umn.edu/*.

[23] A. Eldawy and M. F. Mokbel. A Demonstration of SpatialHadoop: An Efficient MapReduce Framework for Spatial Data. *VLDB Journal*, 6(12):1230–1233, 2013.

[24] A. Eldawy and M. F. Mokbel. The Ecosystem of SpatialHadoop. *SIGSPATIAL Special*, 6(3):3–10, 2014.

[25] Y. J. Garcia, M. A. Lopez, and S. T. Leutenegger. A Greedy Algorithm for Bulk Loading R-Trees. In *Proceddings of the 6th ACM international symposium on Advances in geographic information system*, pages 163–164, 1998.

[26] D. Gavrila. R-Tree Index Optimization. In *Proceedings of the 6th International Symposium on Spatial Data Handling*, pages 771–791, 1994.

[27] H. Gupta, B. Chawda, S. Negi, T. A. Faruquie, and L. Subramanium. Processing Multi-Way Spatial Joins on MapReduce. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 113–124, 2013.

[28] R. H. Guting. An Introduction to Spatial Database Systems. *VLDB Journal*, 3(4):357–399, 1994.

[29] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. *ACM SIGMOD Record*, 14(2):47–57, 1984.

[30] B. Hedlund. Understanding Hadoop Clusters and the Network. In *http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/*.

[31] D. A. Heger. Hadoop Design, Architecture and MapReduce Performance. In *http://www.datanubes.com/mediac/HadoopArchPerfDHT.pdf*.

[32] E. Hoel and H. Samet. A Qualitative Comparison Study of Data Structures for Large Line Segment Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 205–214, 1992.

[33] E. G. Hoel and H. Samet. Performance of Data-Parallel Spatial Operations. In *Proceedings of the 20th International Conference on very Large Data Bases*, pages 156–167, 1994.

[34] S. Hwang, K. Kwon, S. K. Cha, and B. S. Lee. Performance Evaluation of Main-Memory R-Tree Variants. In *Proceedings of the International Symposium on Advances in Spatial and Temporal Databases*, pages 10–27, 2003.

[35] C. F. Ibrahim Kamel. Parallel R-Trees. *ACM SIGMOD Record*, 21(2):195–204, 1992.

[36] Jens, Dittrich, Jorge-Arnulfo, Quiane-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad. Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing). *Proceedings of the VLDB Endowment*, 3(1-2):515–529, 2010.

[37] D. Jiang, B. C. Ooi, L. Shi, and S. Wu. The Performance of MapReduce: An In-depth Study. *Proceedings of the VLDB Endowment*, 3(1-2):472–483, 2010.

[38] F. Jun, T. Zhixian, W. Mian, and X. Liming. HQ-Tree: A Distributed Spatial Index Based on Hadoop. *China communications*, 11(7):128–141, 2014.

[39] I. Kamel and C. Faloutsos. On packing R-trees. In *Proceedings of the 2nd International Conference on Information and Knowledge Management*, pages 490–499, 1993.

[40] I. Kamel and C. Faloutsos. Hilbert R-Tree: An Improved R-tree Using Fractals. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 500–509, 1994.

[41] D. Keim, B. Bustos, S. Berchtold, and H.-P. Kreigel. *Indexing, X-tree*. 2008.

[42] K. Kim, S. K. Cha, and K. Kwon. Optimizing Multidimensional Index Trees for Main Memory Access. *ACM SIGMOD Record*, 30(2):139–150, 2001.

[43] A. Lakshman and P. Malik. Cassandra-A Decentralized Structured Storage System. In *ACM SIGOPS Operating Systems Review*, pages 35–40, 2010.

[44] J. Lawder and P. King. Using Space-Filling Curves for Multi-Dimensional Indexing. In *Proceedings of the 17th British National Conference on Databases: Advances in Databases*, pages 20–35, 2000.

[45] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon. Parallel Data Processing with MapReduce: A Survey. *ACM SIGMOD Record*, 40(4):11–20, 2011.

[46] S. Leutenegger, M. Lopez, and J.Edgington. STR: A Simple and Efficient Algorithm for R-Tree Packing. In *Proceedings of the 13th IEEE International Conference on Data Engineering*, pages 497–506, 1997.

[47] H. Liao, J. Han, and J. Fang. Multi-Dimensional Index on Hadoop Distributed File System. In *Proceedings of the 5th IEEE International Conference on Networking, Architecture, and Storage*, pages 240–249, 2010.

[48] X. Liu, J. Han, Y. Zhong, C. Han, and X. He. Implementing WebGIS on Hadoop: A Case Study of Improving Small File I/O Performance on HDFS. In *Proceedings of the IEEE International Conference on Cluster Computing and Workshops*, pages 1–8, 2009.

[49] Y. Liu, N. Jing, L. Chen, and H. Chen. Parallel Bulk-Loading of Spatial Data with MapReduce: An R-Tree Case. *Wuhan University Journal of Natural Sciences*, 16(6):513–519, 2011.

[50] M.-L. Lo and C. V. Ravishankar. Spatial Joins Using Seeded Trees. *ACM SIGMOD Record*, 23(2):209–220, 1994.

[51] J. Lu and R. H. Guting. Parallel Secondo-Boosting Database Engines with Hadoop. In *Proceedings of the 18th IEEE International Conference on Parallel and Distributed Systems*, pages 738–743, 2012.

[52] Q. Ma, B. Yang, W. Qian, and A. Zhou. Query Processing of Massive Trajectory Data Based on MapReduce. In *Proceedings of the 1st International Workshop on Cloud Data Management*, pages 9–16, 2009.

[53] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis. *R-Trees: Theory and Applications*. 2006.

[54] S. Nishimura, S. Das, D. Agarwal, and A. E. Abbadi. MD-HBase, Design and Implementation of An Elastic Data Infrastructure for Cloud-Based Location Services. *Distributed Parallel Databases*, 31:289–319, 2013.

[55] A. Papadopoulos and Y. Manolopoulos. Parallel Bulk-Loading of Satial Data. *Parallel Computing*, 29(10):1419–1444, 2013.

[56] J. M. Patel and D. J. DeWitt. Partition Based SpatialMerge Join. *ACM SIGMOD Record*, 25(2):259–270, 1996.

[57] A. Pavlo, E. Paulson, A.Rasin, D. abadi, D. DeWitt, S. Madden, , and M. S. braker. A Comparison of Approaches to Large-Scale Data Analysis. In *Proceedings of the 35th ACM SIGMOD International Conference on Management of Data*, pages 165–178, 2009.

[58] J. Rao and K. A. Ross. Making B+- Trees Cache Conscious in Main Memory. *ACM SIGMOD Record*, 29(2):475–486, 2000.

[59] N. Roussopoulos and D. Leifker. Direct Spatial Search on Pictorial Databases Using Packed R-Trees. *ACM SIGMOD Record*, 14(4):17–31, 1985.

[60] H. Samet. *The Design and Analysis of Spatial Data Structures*. 1990.

[61] B. Schnitzer and S. T. Leutenegger. Master-Client R-Trees: A New parallel R-Tree Architecture. In *Proceedings of the 11th IEEE International Conference on Scientific and Statistical Database Management*, pages 68–77, 1999.

[62] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. In *Proceedings of the 13th International Conference on Very Large Data Bases*, pages 507–518, 1987.

[63] S. Shekhar and S. Chawla. *Spatial Databases-A Tour*. 2003.

[64] K.-L. Tan, B. C. Ooi, and D. J. Abel. Exploiting Spatial Indexes for Semijoin-Based Join Processing in Distributed Spatial Database. *IEEE Transactions on Knowledge and Data Engineering*, 12(6):920–937, 2000.

[65] K. Wang, J. Han, B. Tu, J. Dai, W. Zhou, and X. Song. Accelerating Spatial Data Processing with MapReduce. In *Proceedings of the 16th IEEE International Conference on Parallel and Distributed Systems*, pages 229 – 236, 2010.

[66] Y. Wang and S. Weng. Research and Implementation on Spatial Data Storage and Operation Based on Hadoop Platform. In *Proceedings of the 2nd IITA International Conference on Geoscience and Remote Sensing*, pages 275 – 278, 2010.

[67] X. Wu and C. Zang. A New Spatial Index Structure for GIS Data. In *Proceedings of the 3rd IEEE International Conference on Multimedia and Ubiquitous Engineering*, pages 471–476, 2009.

[68] L. Xun and Z. Wenfeng. Parallel Spatial Index Algorithm based on Hilbert Partition. In *Proceedings of the IEEE International Conference on Computational and Information Sciences*, pages 876–879, 2013.

[69] C. Zhang, F. Li, and J. Jestes. Efficient Parallel kNN Joins for Large Data in MapReduce. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 38–49, 2012.

[70] S. Zhang, J. Han, Z. Liu, K. Hwang, and Z. Xu. SJMR: Parallelizing Spatial Join with MapReduce on Clusters. In *Proceedings of the IEEE International Conference on Cluster Computing and Workshops*, pages 1–8, 2009.

[71] K. Zheng and Y. Fu. Research on Vector Spatial Data Storage Schema Based on Hadoop Platform. *International Journal of Database Theory and Application*, 6(5):85–94, 2013.

[72] Y. Zhong, J. Han, T. Zhang, Z. Li, J. Fang, and G. Chen. Towards Parallel Spatial Query Processing for Big Spatial Data. In *Proceedings of the IEEE 26th International Conference on Parallel and Distributed Processing*, pages 2085 – 2094, 2012.

Archimedes: Efficient Query Processing over Probabilistic Knowledge Bases

Yang Chen*, Xiaofeng Zhou*, Kun Li†, Daisy Zhe Wang*
*Department of Computer and Information Science and Engineering, University of Florida
†Google, Inc.

ABSTRACT

We present the ARCHIMEDES system for efficient query processing over probabilistic knowledge bases. We design ARCHIMEDES for knowledge bases containing incomplete and uncertain information due to limitations of information sources and human knowledge. Answering queries over these knowledge bases requires efficient probabilistic inference. In this paper, we describe ARCHIMEDES's efficient knowledge expansion and query-driven inference over UDA-GIST, an in-database unified data- and graph-parallel computation framework. With an efficient inference engine, ARCHIMEDES produces reasonable results for queries over large uncertain knowledge bases. We use the Reverb-Sherlock and Wikilinks knowledge bases to show ARCHIMEDES achieves satisfactory quality with real-time performance.

1 Introduction

Recent years have seen a drastic rise in the construction of web knowledge bases (KBs), e.g., DBPedia, Freebase, NELL, Probase, and YAGO. Meanwhile, due to the uncertainty of information extraction algorithms and the limitations of human knowledge, current knowledge bases are still incomplete and uncertain, resulting in suboptimal query results [5, 34]. The objective of this paper is to extend our previous research on knowledge expansion [5], query-driven inference [35], and the UDA-GIST computation framework [17] to build a prototype knowledge base system, ARCHIMEDES, to support efficient knowledge expansion with uncertain Horn clauses and query-driven probabilistic inference.

Knowledge Expansion. ARCHIMEDES applies large sets of Horn clauses to derive implicit knowledge from existing knowledge bases. The rules are constructed by state-of-the-art first-order mining algorithms [6, 4, 9, 25]. It employs a novel relational model [5] to apply batches of inference rules using relational operations and performs probabilistic inference by query-driven MCMC [18].

Query-Driven Inference. Observing that queries are often relevant to small parts of the knowledge graphs [26, 30], ARCHIMEDES applies MCMC only to the K-hop networks to achieve real-time performance. Specialized MLN inference algorithms [10, 14, 23] can improve inference quality over the K-hop network, but MCMC is more widely supported by existent big data frameworks, e.g., UDA-GIST [17] and GraphLab [19].

UDA-GIST. UDA-GIST [17] is an in-database analytics framework unifying data-parallel and graph-parallel computation. State-of-the-art big data frameworks support either data-parallel or graph-parallel computation. GraphLab [19], for example, supports only graph-parallel computation; Spark [32, 33] and MapReduce [7], on the other hand, support only data-parallel computation. UDA-GIST unifies these two types of parallel computation in a cohesive scalable system.

We evaluate ARCHIMEDES on Sherlock-Reverb [25, 8] and Wikilink [29]. These datasets contain large-scale, incomplete, and uncertain knowledge. We compare with Tuffy [22], the probabilistic inference engine of DeepDive, and GraphLab. We show that ARCHIMEDES produces competent result with efficient first-order reasoning and query-driven inference supported by the UDA-GIST in-database framework. We demonstrate the system with ARCHIMEDESONE [35], an interactive query interface. All our code and data are available online[1].

To summarize, we solve the problem of efficient query processing in probabilistic knowledge bases with three novel contributions:

- **Knowledge expansion:** Derive implicit knowledge from knowledge bases using large rule sets;
- **Query-driven inference:** Improve inference performance by focusing MCMC on the query variables;
- **Efficient computation:** Leverage the UDA-GIST unified data- and graph-parallel computation framework.

We organize the remainder of this paper as follows. Section 2 describes the overview of ARCHIMEDES system design. Sections 3 to 5 describe the system components in detail. Section 6 presents experimental evaluation with public knowledge bases. Section 7 discusses related work, and Section 8 concludes the paper.

[1]http://dsr.cise.ufl.edu/projects/probkb-web-scale-probabilistic-knowledge-base.

Entities $\mathcal{E}$	Classes $\mathcal{C}$	Relations $\mathcal{R}$	Relationships Π
Ruth Gruber, New York City, Brooklyn	W (Writer) = {Ruth Gruber}, C (City) = {New York City}, P (Place) = {Brooklyn}	BornIn(W, P), BornIn(W, C), LiveIn(W, P), LiveIn(W, C), LocateIn(P, C)	0.96 BornIn(Ruth Gruber, New York City) 0.93 BornIn(Ruth Gruber, Brooklyn)

Rules $\mathcal{L}$
1.40 $\forall w \in W\ \forall p \in P$ (LiveIn(w, p) $\leftarrow$ BornIn(w, p))
1.53 $\forall w \in W\ \forall c \in C$ (LiveIn(w, c) $\leftarrow$ BornIn(w, c))
0.32 $\forall p \in P\ \forall c \in C\ \forall w \in W$ (LocateIn(p, c) $\leftarrow$ LiveIn(w, p) $\wedge$ LiveIn(w, c))
0.52 $\forall p \in P\ \forall c \in C\ \forall w \in W$ (LocateIn(p, c) $\leftarrow$ BornIn(w, p) $\wedge$ BornIn(w, c))
∞ $\forall c_1 \in C\ \forall c_2 \in C\ \forall w \in W$ (BornIn(w, c_1) $\wedge$ BornIn(w, c_2) $\rightarrow$ $c_1 = c_2$)

Table 1: Example probabilistic knowledge base constructed from Reverb-Sherlock extractions.

2 Probabilistic Knowledge Bases

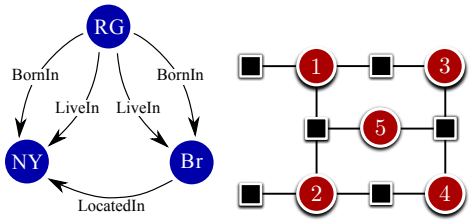
A probabilistic knowledge base extends first-order knowledge bases to support uncertain facts and rules. The primary goal of modeling uncertainty is to represent knowledge mined by probabilistic information extraction algorithms that contain uncertain facts and rules, as illustrated by the Reverb-Sherlock KB in Table 1. We formally define a probabilistic knowledge base below [5].

Definition 1. We define a *probabilistic knowledge base* to be a 5-tuple Γ=($\mathcal{E}$,$\mathcal{C}$,$\mathcal{R}$,Π,$\mathcal{L}$), where

1. $\mathcal{E} = \{e_1, \ldots, e_{|\mathcal{E}|}\}$ is a set of *entities*. Each entity $e \in \mathcal{E}$ refers to a real-world object.
2. $\mathcal{C} = \{C_1, \ldots, C_{|\mathcal{C}|}\}$ is a set of *classes* (or *types*). Each class $C \in \mathcal{C}$ is a subset of $\mathcal{E}$: $C \subseteq \mathcal{E}$.
3. $\mathcal{R} = \{R_1, \ldots, R_{|\mathcal{R}|}\}$ is a set of *relations*. Each $R \in \mathcal{R}$ defines a binary relation on $C_i, C_j \in \mathcal{C}$: $R \subseteq C_i \times C_j$. We call C_i, C_j the *domain* and *range* and use $R(C_i, C_j)$ to denote the relation with its domain and range.
4. $\Pi = \{(r_1, w_1), \ldots, (r_{|\Pi|}, w_{|\Pi|})\}$ is a set of *weighted facts* (or *relationships*). For each $(r, w) \in \Pi$, r is a tuple (R, x, y), where $R(C_i, C_j) \in \mathcal{R}$, $x \in C_i, y \in C_j$, and $(x, y) \in R$; $w \in \mathbb{R}$ is a weight indicating how likely r is true. We also use $R(x, y)$ to denote the tuple (R, x, y).
5. $\mathcal{L} = \{(F_1, W_1), \ldots, (F_{|\mathcal{L}|}, W_{|\mathcal{L}|})\}$ is a set of *weighted clauses* (or *rules*). It defines a *Markov logic network*. For each $(F, W) \in \mathcal{L}$, F is a first-order logic clause, and $W \in \mathbb{R}$ is a weight indicating how likely F holds.

As in [5], we confine $\mathcal{L}$ to Horn clauses with binary predicates. Horn clauses prove useful in various knowledge base inference tasks [21, 5, 25]. Their similar structures facilitate efficient inference engines leveraging the KB relational model in Section 3.

Example 1. Table 1 shows an example probabilistic KB constructed from Reverb [8] extractions and Sherlock [25] rules. The knowledge base describes the birth place and city of a writer Ruth Gruber with relations "BornIn," "LiveIn," and "LocateIn." The extracted facts state that Ruth Gruber was born in New York City and Brooklyn with weights 0.96 and 0.93, respectively, assigned by IE algorithms. The weighted rules infer Ruth Gruber's living place based on his birth place, and a hard rule with an infinite positive weight states that a person was born in only one city. □



(a) Knowledge graph. **(b) Factor graph.**

ID	Fact
1	BornIn(Ruth Gruber, New York City)
2	BornIn(Ruth Gruber, Brooklyn)
3	LiveIn(Ruth Gruber, New York City)
4	LiveIn(Ruth Gruber, Brooklyn)
5	LocatedIn(Brooklyn, New York City)

(c) Variables 1-5 in the factor graph (b).

Figure 1: Factor graph representation of the Reverb-Sherlock knowledge base.

We view a probabilistic knowledge base as a template for constructing ground factor graphs [24]. A *factor graph* is a set of factors $\Phi = \{\phi_1, \ldots, \phi_N\}$, where each factor ϕ_i is a function $\phi_i(\mathbf{X}_i)$ over a random vector $\mathbf{X}_i$ indicating the probabilistic correlations among the random variables in $\mathbf{X}_i$. These factors together determine a joint probability distribution over the random vector $\mathbf{X}$ consisting of all the random variables in the factors [16]. Factor graphs are visually represented as graphs. In the graph representation, each node is a fact X_i (circle) or factor $\phi_i(\mathbf{X}_i)$ (square) with variables $\mathbf{X}_i$ as its neighbors. Figure 1(b) shows an example factor graph representation of the knowledge graph Figure 1(a). Each factor in Figure 1(b) is defined by a ground fact, e.g., BornIn(Ruth Gruber, New York City) with weight 0.96, or a ground rule, e.g., LiveIn(Ruth

Gruber, New York City) ← BornIn(Ruth Gruber, New York City) with weight 1.40. We use factor graphs to describe these correlations among the facts.

In a factor graph $\Phi = \{\phi_1, \ldots, \phi_N\}$, the factors together determine a joint probability distribution over the random vector $\mathbf{X}$ consisting of all the random variables in the factor graph:

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \prod_i \phi_i(\mathbf{X}_i) = \frac{1}{Z} \exp\left(\sum_i W_i n_i(\mathbf{x})\right), \quad (1)$$

where $n_i(\mathbf{x})$ is the number of true groundings of rule F_i in $\mathbf{x}$, W_i is its weight, and Z is the partition function, i.e., normalization constant. ARCHIMEDES answers user queries by computing the marginal probability $P(X = x)$, the probability distribution of a query node X defined by (1). The computation of marginal probabilities is called *marginal inference* in probabilistic graphical models literature. Exact inference is tractable for only limited families of graphical models [16], and state-of-the-art MLN inference engines use sampling algorithms including Markov chain Monte Carlo (MCMC) and MC-SAT [24, 23, 22]. Observing the ground factor graphs of real knowledge bases are large [5] while user queries often focus on small parts of the knowledge graph [35], ARCHIMEDES employs a query-driven approach to focus MCMC on the query nodes to avoid computation over the entire factor graph.

2.1 System Architecture

To efficiently process queries, we design three key components of ARCHIMEDES: an inference engine for efficient knowledge expansion to derive implicit knowledge from existing KBs [5], query-driven inference to compute probabilities of the query facts [35], and the UDA-GIST framework for in-database data-parallel and graph-parallel analytics [17]. We provide a user interface for load, search, and update queries, as described in [35]. The system architecture is shown in Figure 2.

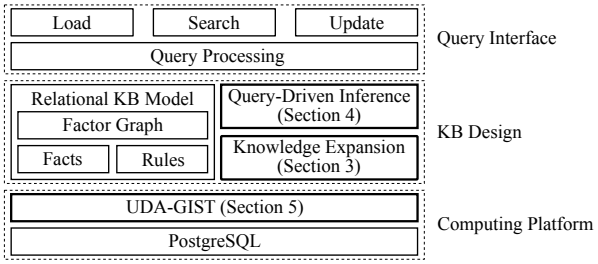


Figure 2: ARCHIMEDES System Components.

ARCHIMEDES models facts, rules, and the factor graph in relational tables. The relational model enables it to efficiently perform knowledge expansion by joining the facts and rules tables. The knowledge expansion and query-driven inference using MCMC exemplify applications requiring both data-parallel and graph-parallel computation. They are efficiently supported by the UDA-GIST in-database analytics framework by unifying the UDAs from relational databases and GIST from graph analytics with a shared in-memory state. We describe the details of each component in Sections 3 to 5.

3 Knowledge Expansion

To efficiently apply the inference rules, we represent a knowledge base as relational tables. This relational model is first introduced by ProbKB [5] and proves efficient in rule mining [4] by applying rules in batches using join queries. The main challenge with inference rules is that they have flexible structures. To adapt for different structures, we define structural equivalence to divide rules into equivalent classes so that each equivalent class has a fixed table format. In particular, we call two first-order clauses *structurally equivalent* if they differ only in entities, types, and predicates.

Example 2. Consider the following inference rules:

1. isMarriedTo(x, y) ← isMarriedTo(y, x);
2. isInterestedIn(x, y) ← influences(y, x) ;
3. influences(x, y) ← directed(x, z), actedIn(y, z);
4. influences(x, y) ← worksAt(x, z), worksAt(y, z).

Rules 1 and 2 are structurally equivalent since their only differences are the predicates (isMarriedTo, influences, isInterestedIn). Similarly, Rules 3 and 4 are structurally equivalent. Therefore, we store Rules 1 and 2 in one table with the columns specifying the predicates of the head and body, as shown in Table 2 (left). We store Rules 3 and 4 in Table 2 (right), its columns storing the head and first, second predicates of the rule body. □

Head	Body
isMarriedTo	isMarriedTo
isInterestedIn	influences

Head	Body1	Body2
influences	directed	actedIn
influences	worksAt	worksAt

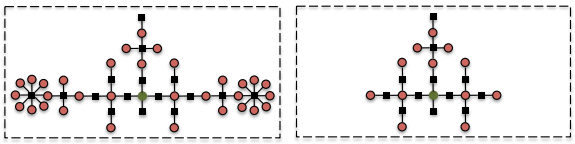
Table 2: (Left) Relational table for rules 1 and 2. (Right) Relational table for rules 3 and 4.

Based on the relational model, we express the knowledge expansion algorithm as join queries between the facts and rules tables, one join for each rules table. The details of the join queries are described in [5]. Our experiments show that applying rules in batches results in a 200-300 times of speedup over the state-of-the-art approaches. The result of knowledge expansion is a ground factor graph $\Phi = \{\phi_1, \ldots, \phi_N\}$, where each factor $\phi_i(\mathbf{X}_i)$ represents a ground rule. The factor graph is modeled by a relational table, the columns storing predicate IDs of variables $X \in \mathbf{X}$ and weights of the factors. Performing probabilistic inference on this factor graph yields marginal probabilities of the query facts.

4 Query-Driven Inference

ARCHIMEDES uses query-driven inference to speed up MLN inference algorithms by focusing computation on the query facts. The query-driven inference algorithm is designed with the UDA-GIST analytics framework [17] to achieve efficient inference in a relational database system. Furthermore, we use K-hop approximation to focus computation on the query facts.

K-hop approximation. To achieve real-time response, we approximate the inference by extracting K-hop sub-networks of the ground factor graph, consisting of nodes within K hops from the query nodes. The K-hop approximation is based on the observation that neighbors of the query nodes have more influence than distant nodes. In Figure 3(a), for example, to compute the probability of the central node, we use the 2-hop sub-network in Figure 3(b) for approximation. To achieve real-time response, we use an additional network limit parameter to control the expansion of K-hop sub-networks as K increases. In our evaluation, we achieve an 18 times of speedup compared to inference over the entire factor graph by choosing $K = 2$, with an acceptable error of 0.04 in the computed probabilities.



(a) (b)

Figure 3: (a) The original factor graph. (b) 2-hop network.

UDA-GIST. We use the MCMC algorithms to compute the probabilities defined by the K-hop network. We optimize MCMC on the UDA-GIST in-database analytics framework [17]: we build the factor graph by relational operations with User Defined Aggregates (UDAs) and compute probabilities of query facts by MCMC with General Iterative State Transition (GIST). The combined UDA-GIST framework extends relational database systems to support algorithms requiring both data- and graph-parallel computation, including MCMC and MC-SAT. We describe the design of UDA-GIST in Section 5.

5 UDA-GIST

Most major DBMSes support User-Defined Aggregates (UDAs) for parallel data analytics. UDAs are suitable for data-parallel analytics where data are naively partitioned and computation is performed on the partitions in parallel. In the context of query processing over large probabilistic KB graphs, such data-parallel operators implement efficient propositional KB graph materialization, subgraph matching, and result generation.

However, UDAs do not support efficient statistical inference algorithms that perform iterative transitions over a large state, where the state is a graph-like data structure. The computation is not naively partitioned due to data dependency within the state (e.g., dependencies between nodes and edges in a graph) as DBMSes are fundamentally data driven and computation is tied to the processing of tuples. We refer to these iterative processing algorithms as graph parallel algorithms. MCMC and random walk over large probabilistic graphical graph are examples of such algorithms. The fundamental question is: *Can graph-parallel inference algorithms be efficiently implemented in DBMSes?*

The General Iterative State Transition (GIST) Operator. To answer the demand of supporting in-database graph-parallel analytics, we propose the GIST abstraction to generalize the GraphLab API [17]. GIST defines four abstract data types to describe state-transition algorithms: an in-memory *state* representing the state space, a *task* encoding the state transition task for each iteration, a *scheduler* responsible for the generation and scheduling of tasks, and a *convergence UDA (cUDA)* imposing the stopping condition of the GIST operations. An efficient GIST implementation also supports optimizations including (1) asynchronous parallelization of state transitions, (2) efficient and flexible state implementation, and (3) code generation.

The UDA-GIST Data Processing Framework. We integrate the GIST operator into DBMSes with UDAs and User-Defined Functions [2]. From the relational representation of a probabilistic KB graph, SQL queries and UDAs generate a large in-memory state representing the propositional KB graph. The GIST operator then runs parallel inference algorithms on the in-memory state. The query results are extracted from the converged state using an independent UDA function. UDA-GIST unifies data-parallel (e.g., graph materialization) and graph-parallel computation (e.g., inference) into an integrated in-database analytics framework.

6 Experiments

We evaluate ARCHIMEDES using Reverb-Sherlock [8, 25] Wikipedia KB with 407,247 facts and 30,912 first-order inference rules, a synthetic knowledge base with varying numbers of facts and rules ranging from 10K to 10M, and Wikilinks for cross-document coreference on UDA-GIST. We run the experiments on a 32-core machine with 64GB of RAM running Red Hat Linux 4.

6.1 Result of Knowledge Expansion

To evaluate knowledge expansion, we use Tuffy [22] as the baseline. Figures 4(a)(b) compare performance of ARCHIMEDES with Tuffy on the synthetic knowledge base with varying numbers of facts and rules. We see that ARCHIMEDES achieves more than 200 times of

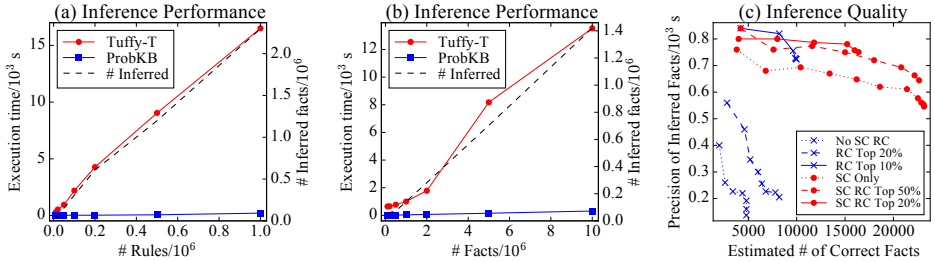


Figure 4: Knowledge expansion results. (a)(b) Performance comparison with Tuffy. (c) Quality improvement on Reverb-Sherlock.

speedup over Tuffy for 10^7 facts. The speedup benefits from the batch application of rules with join operations supported by the relational knowledge base model.

The precision of the inferred facts is shown in Figure 4(c). We use semantic constraints and rule cleaning to improve precision of the inferred facts [5]. As shown in the figure, both semantic constraints and rule cleaning improve precision. The raw Reverb-Sherlock dataset infers 4800 new correct facts at a precision of 0.14. The precision drops quickly when we generate new facts since unsound rules and ambiguous entities result in many erroneous facts. On the contrary, the precision significantly improves with our quality control methods: with top 10% rules we infer 9962 facts at a precision of 0.72; with semantic constraints, we infer 23,164 new facts at precision 0.55. Combining these two methods, we are able to infer 22,654 new facts at precision 0.65 using top 50% rules, and 16,394 new facts at precision 0.75 using top 20% rules.

6.2 Result of Query-Driven Inference

Figures 5(a)-(c) report the runtime results for query-driven inference by K-hop approximation with different numbers of hops from large, medium, and small clusters. We see that in all the networks, as the number of hops and size of the retrieved networks grow, it takes longer for inference. As a result, query-driven inference achieves a speedup of more than one order of magnitude compared to using the entire factor graph for computation. Meanwhile, we observe that the error rate in the computed probabilities drops to 0.04 with only 3000 neighboring nodes in the MCMC computation. Thus, query-driven inference efficiently answers user queries by focusing computation on the relevant neighbors with acceptable error rates in the computed probabilities.

6.3 Result of UDA-GIST

We evaluate the performance and scalability of UDA-GIST by cross-document coreference using the Wikilinks datasets [29]. The dataset contains about 40 millions mentions over 3 millions entities. We extract two datasets: Wikilink 1.5 (first 565 1.5M mentions from the 40M dataset) and Wikilink 40 (all 40M mentions in the dataset) from this Wikilink dataset. The Wikilink 40 dataset is 27 times larger than used in the current state-of-the-art [28]. The result is reported in Figures 5(d)(e). For the entire dataset, the state building takes approximately 10 minutes. We run 20 iterations each with 10^{11} pairwise mention comparisons. Each iteration takes approximately 1 hour and we see the graph converges at iteration 10 with precision 0.79, recall 0.83 and F_1 0.81. Using our solution, within a manageable 10-hour computation in a single system the coreference analysis can be performed on the entire Wikilink dataset, 27 times larger than achieved by the current state-of-the-art [28].

7 Related Work

Knowledge Base Construction. Knowledge bases are receiving increasing research and industrial interest, e.g.: DBpedia [1], Freebase [3], NELL [21], ProBase [31], and YAGO [20]. However, they are often incomplete and uncertain due to limitations of information sources and human knowledge. To model the correlations among uncertain facts, NELL [21] and DeepDive [34] use inference rules. We extend this approach to large rule sets with similar structures [6, 4, 9, 25] by modeling the rules as relational tables, enabling the applications of batches of inference rules with relational operators [5]. Our approach is scalable; it has been applied to large knowledge bases including Freebase [6, 4].

Probabilistic Inference. To compute the probabilities, general probabilistic inference algorithms–MCMC [27], Gibbs sampling [11], belief propagation [12]–or specialized MLN inference algorithms [10, 14, 23] are viable options. All the inference algorithms benefit from query-driven inference by avoiding computation on the entire graph [35, 26, 30]. In ARCHIMEDES, we use MCMC because of its wide use and existing support in UDA-GIST [18] and other state-of-the-art big data analytics frameworks we describe below.

Parallel computing. In recent years, various analytics frameworks have been developed to facilitate large-scale data analytics: MADlib [15], Spark [33, 32], MapReduce [7], GraphLab [19], and GraphX [13]. These frameworks support either data-parallel or graph-parallel computation. For example, the data-driven UDA operations in MADlib, Spark, and MapReduce provide data-parallel

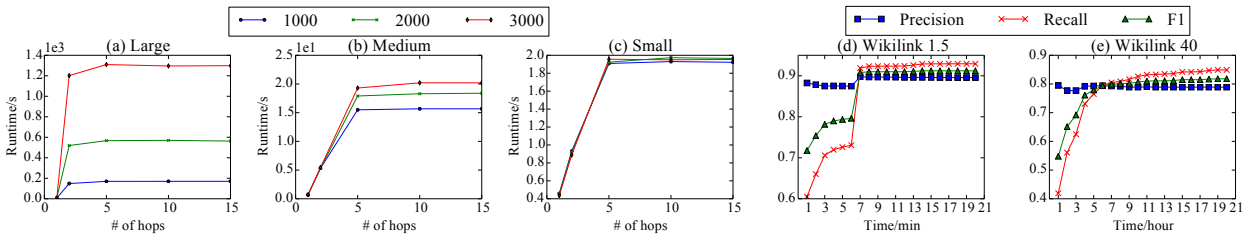


Figure 5: (a)-(c) Performance of query-driven inference. (d)(e) Performance improvement of UDA-GIST compared to GraphLab.

computation, but are inefficient for asynchronous graph-parallel computation like MCMC. GraphX, built on Spark, is based on a synchronous computation engine, making MCMC less efficient than GraphLab [17]. GraphLab, however, requires sequential graph construction and result extraction. The UDA-GIST framework improves on these works by integrating UDA and GIST with a shared in-memory state, thus unifying data- and graph-parallel computation frameworks in a DBMS.

8 Conclusion

In this paper, we present ARCHIMEDES for query processing over probabilistic knowledge bases. We extend the state-of-the-art query processing and optimization techniques to knowledge base systems by knowledge expansion and query-driven inference, supported by the UDA-GIST framework. UDA-GIST is an in-database analytics framework that unifies data-parallel and graph-parallel computation. We evaluate ARCHIMEDES with public knowledge bases including Reverb-Sherlock and Wikilink. We show ARCHIMEDES achieves real-time performance with satisfactory quality. In future work, we plan to improve the query processing algorithm and supporting framework with performance optimizations.

Acknowledgments. We acknowledge the support of NSF under IIS Award # 1526753, DARPA under FA8750-12-2-0348-2 (DEFT/CUBISM), and a gift from Google.

9 References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*. Springer, 2007.

[2] T. Bain, L. Davidson, R. Dewson, and C. Hawkins. User defined functions. In *SQL Server 2000 Stored Procedures Handbook*, pages 178–195. Springer, 2003.

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*. ACM, 2008.

[4] Y. Chen, S. Goldberg, D. Z. Wang, and S. S. Johri. Ontological pathfinding. In *SIGMOD*. ACM, 2016.

[5] Y. Chen and D. Z. Wang. Knowledge expansion over probabilistic knowledge bases. In *SIGMOD*. ACM, 2014.

[6] Y. Chen, D. Z. Wang, and S. Goldberg. Scalekb: Scalable learning and inference in large knowledge bases. *The VLDB Journal*, 2016.

[7] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 2008.

[8] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

[9] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, 2015.

[10] W. Gatterbauer and D. Suciu. Dissociation and propagation for approximate lifted inference with standard relational database management systems. *The VLDB Journal*, 2016.

[11] J. Gonzalez, Y. Low, A. Gretton, and C. Guestrin. Parallel gibbs sampling: From colored fields to thin junction trees. In *AISTATS*, 2011.

[12] J. Gonzalez, Y. Low, and C. Guestrin. Residual splash for optimally parallelizing belief propagation. In *AISTATS*, 2009.

[13] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica. Graphx: Graph processing in a distributed dataflow framework. In *OSDI*, 2014.

[14] E. Gribkoff and D. Suciu. Slimshot: in-database probabilistic inference for knowledge bases. *Proceedings of the VLDB Endowment*, 2016.

[15] J. M. Hellerstein, C. Ré, F. Schoppmann, D. Z. Wang, E. Fratkin, A. Gorajek, K. S. Ng, C. Welton, X. Feng, K. Li, et al. The madlib analytics library: or mad skills, the sql. *VLDB*, 2012.

[16] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[17] K. Li, D. Z. Wang, A. Dobra, and C. Dudley. Uda-gist: An in-database framework to unify data-parallel and state-parallel analytics. *VLDB*, 2015.

[18] K. Li, X. Zhou, D. Z. Wang, C. Grant, A. Dobra, and C. Dudley. In-database batch and query-time inference over probabilistic graphical models using uda-gist. *The VLDB Journal*, 2016.

[19] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *VLDB*, 2012.

[20] F. Mahdisoltani, J. Biega, and F. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*, 2014.

[21] T. Mitchell and et. al. Never-ending learning. In *AAAI*, 2015.

[22] F. Niu, C. Ré, A. Doan, and J. Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *VLDB*, 2011.

[23] H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, 2006.

[24] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 2006.

[25] S. Schoenmackers, O. Etzioni, D. S. Weld, and J. Davis. Learning first-order horn clauses from web text. In *EMNLP*, 2010.

[26] J. Shin, S. Wu, F. Wang, C. De Sa, C. Zhang, and C. Ré. Incremental knowledge base construction using deepdive. *VLDB*, 2015.

[27] S. Singh. *Scaling MCMC Inference and Belief Propagation to Large, Dense Graphical Models*. PhD thesis, University of Massachusetts Amherst, 2014.

[28] S. Singh, A. Subramanya, F. Pereira, and A. McCallum. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Proceedings of ACL-HLT*, 2011.

[29] S. Singh, A. Subramanya, F. Pereira, and A. McCallum. Wikilinks: A large-scale cross-document coreference corpus labeled via links to wikipedia. *University of Massachusetts, Amherst, Tech. Rep.*, 2012.

[30] M. L. Wick and A. McCallum. Query-aware mcmc. In *NIPS*, 2011.

[31] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*, 2012.

[32] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, 2012.

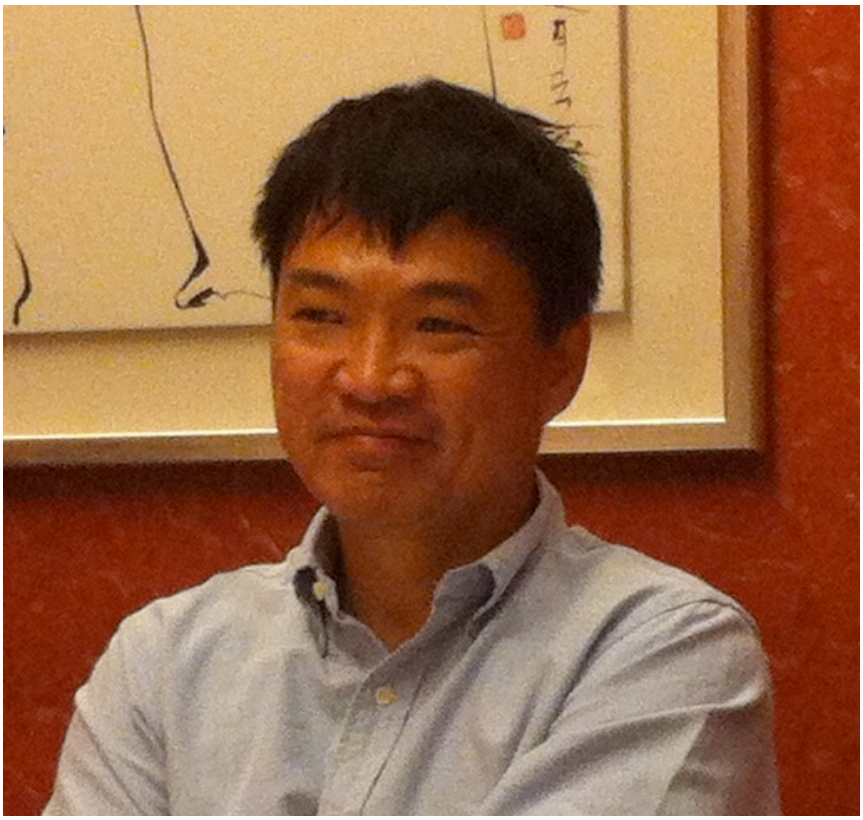
[33] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 2010.

[34] C. Zhang. *DeepDive: A Data Management System for Automatic Knowledge Base Construction*. PhD thesis, UW-Madison, 2015.

[35] X. Zhou, Y. Chen, and D. Z. Wang. Archimedesone: Query processing over probabilistic knowledge bases. *VLDB*, 2016.

Beng Chin Ooi Speaks Out on Building a Strong Database Group

Marianne Winslett and Vanessa Braganholo



Beng Chin Ooi
http://www.comp.nus.edu.sg/~ooibc/

Welcome to ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are at my office at the Advanced Digital Sciences Center in Singapore, an outpost of the University of Illinois. I have here with me today Beng Chin Ooi who is the dean of the school of computing at the National University of Singapore where he's been a professor of computer science for many years. Beng Chin is editor-in-chief for IEEE Transactions on Knowledge and Data Engineering. He is the recipient of the 2009 SIGMOD Contributions Award, and he is an IEEE and ACM Fellow and Fellow of Singapore National Academy of Science. He is the co-founder of two startups and his Ph.D. is from Monash University. So Beng Chin, welcome! ***(Please note that this interview took place in 2011)***

Okay, so, what's it like to be the dean?

That's an easy job. It's great! As a dean, I attend two morning meetings with the bosses and do lunch meetings with heads, vice deans, and assistant deans. So that means I have four meetings to attend a month, and the job is easy.

I don't believe you, but we're going to come back to that topic later. Don't believe him, readers!

OK, so let's switch to something technical. Cloud computing is hot. What will be the next big research issue in cloud computing?

For cloud computing, since it's a distributed cluster computing environment, issues such as data and system security, transaction management, efficiency, and query processing strategies are not yet fully resolved. They will be solved eventually, and of course, there are many other challenges that will come along as we make advancements in the hardware. For example, suppose the network bandwidth increases... then we will have different problems to solve. Also, with the introduction of the PCM (Phase Change Memory) chip and all other things, we will have to redesign architectures...

So why is transaction processing performance lower in the cloud than in a non-cloud environment?

In the cloud environment, because it's a distributed environment, suppose you want to enforce ACID properties. Then the locking overhead is going to take much longer, and therefore people go for less consistency for the sake of performance.

So what's the new hardware that you mentioned that you see coming in the future?

We expect the launch of PCM (Phase Change Memory) in big scale, and that will provide us much bigger memory storage. This will change the way we design the algorithms because now everything can be online rather than have to handle data from disk. Of course, PCM has its own problems such as read durability, and therefore we have to reduce the *reads and the writes on the PCM.*

Coming back to the performance issue, the relational databases in the cloud are facing a lot of competition from NoSQL and big data approaches. Do you think that relational databases can compete with those two when high throughput is important?

We have to be clear that these two types of systems have been designed to meet different requirements and even different budget constraints. So they serve different market segments and therefore, there is no real competition. Of course, you can always extend relational database systems to handle what we have seen in a Web 2.0 application, but by that it's likely we end up with two engines running in parallel with some data sharing between the two engines.

I hear that you love B-trees compared to, for example, R-trees. What's wrong with R-trees?

That is not quite true. I love R-trees as well, but the problem with R-trees is the speed to process. Multiple traverses will take up much longer time during the locking process, so it incurs in a higher locking overhead.

That's true, but KD-trees, do you like them any better?

A KD-tree is fine, but the way it partitions the space does not lead to a balanced B+ tree.

So what trees do you like for spatial-temporal type…?

I like the B+ tree because it's very efficient, dynamic, and it's self-adaptive to load and data distribution.

But it doesn't work for multidimensional search, does it?!

Once you can linearize the data properly, then we can use the B+ tree.

You mean like a Z-ordering or something?

Or the Z-curve or we make use of distance just like the way we do it in iDistance, where we try to argue that even in high-dimensional space, I can measure one object against a reference point. If I can do that, that means that I divide the space into the Voronoi cells efficiently and that makes it very efficient. Then I can measure the distance to the reference point. Once I use that reference point plus the distance, I can index the data points using the typical B+ tree without changing the structures of the B+ tree.

I see. Okay, great. In the last five years, you've turned your attention from just writing papers to having startups. How do you make a database startup successful?

I set up my first startup in 1999 when we tried to provide photo-sharing systems to the public. That system works like Flickr except that we use keywords rather than tagging as well as comments for users to provide keywords to describe the photos, for users to share photos with a friend, and so on. But somehow it did not take off. So, to some extent, I did a startup 11 years ago, not five years ago.

Okay, okay, and more recently?

More recently, I had a startup on peer-to-peer systems based on my work on Bestpeer[1], and recently it has started to draw attention from venture capitalists, so I started to draw some business, but that is in China.

So do startups in China need to be entirely Chinese to be successful?

Not necessarily. But it's much harder to get the license to operate the business.

> ***[…] for any department to be good in research, we need a huge number of good graduate students***

So, what can you say about the startup culture in China?

I think it's good because Chinese always like money. Just like me. I love money.

Okay, and is your startup in the Beijing area or …?

In Hangzhou.

Are there many startups there now?

Hangzhou is a good place for startups. In fact, Alibaba has its headquarter and campus in Hangzhou, and one of the biggest web and game companies called NetEase is also housed in Hangzhou.

Okay, so I see now that you haven't gone back to the question of how do we make a database startup successful. So, is it like any other startup? Or different?

It's much harder to have a startup on trying to sell database systems because the market seems to be cornered by a few big players and in reality, it is very hard to do database migrations. So nobody is going to buy a new database system just because the new system has a few more functionalities or because it is cheaper since the data migration process is very expensive and time-consuming.

Does that mean your company aiming at customers that don't already have an existing DBMS?

The company that we have (Bestpeer, a P2P company) aims at customers who want to integrate details from different sources of databases and of course you can use Bestpeer like a middleware to link up different systems to share the data.

OK, great!

So, some of our readers told me: "Ten years ago we could seldom find papers from National University of Singapore (NUS) in tier 1 venues of the database community. Now, Beng Chin's database group is such an important part of the community pushing the frontiers of everyone else's research." Another reader said that you "built up NUS into a strong database research university starting from very little. I think this is an exceptional accomplishment, he hired everyone in the database group there, he aggressively recruited good students, particularly from China. He established a culture of publishing in the top places. He also actively mentored Ph.D. graduates, and all this was done well before he was dean. He was just a faculty member taking initiative and making things happen. Over time, he got institutional recognition and support including eventually his current deanship." So, long story but first question: what inspired you to do all this?

First of all, I just want to prove myself. Prove to myself that I could be just as good as any other database researchers. So in order to prove myself on that front, I had to publish where the top people published. And therefore even when I was doing my Ph.D., I started to focus on publishing at the top places. It's a fact that for any department to be good in research, we need a huge number of good graduate students. Therefore, we have to go out to look for good students. We are not MIT or Stanford, which can

[1] http://www.bestpeer.com/

attract good students without doing extensive outreach to these students. So for us, we have to go around and look for good students. We've convinced our colleagues in China, in Eastern Europe, and a few other countries to recommend good students to us.

So, for our readers who want to build up their own research group or department, how are you able to change the culture of your environment?

That is easy because no one wants to be a loser. So, the first step is to recruit good people and once they are in, provide them the environment, the freedom to do the research that they want to do, and support them in what they want to do. Set the goals and tell them what they are expected to do and explain how they will be rewarded. Once all this is made transparent, people tend to accept. Transparency reduces conflicts and workload for the administrators.

So then what were the most difficult challenges that you had to overcome to do this? Did they maybe oppose you on a rewards angle? Or what was the hardest part?

All were equally easy. [laughs]

OK, so let's see. What kind of rewards did you offer people when you talk of reward?

In our systems, we have a performance-based increment as well as performance-based bonus. That forms the rewards for those people who have done well for the last year and for the last three years. That really pushes people to work hard because everyone can be motivated to do well. Also, people are motivated when they are recognized.

I see. Did you have to base that off of that Clarivate SCI index or were you able to use other indicators of research quality and quantity?

We look at more than just the papers or the SCI index. We look at the impact a person makes to the research he's working on, his community, and his contributions.

So in the long run, I see how you can measure that impact, for example, it will show up in the H-index and everything, but you said last one year and last three years. So how can you measure the impact over that kind of time?

For the last one year, we look at how they perform in terms of the way they teach, the way they do their research, what papers they have got and what kinds of awards they've gotten for the last one year. For the three years window periods, we look at a much longer period where they could have made slightly longer impact for certain years, but they may not have the papers. We also look at their whole career since they joined the school.

Set the goals and tell them what they are expected to do and explain how they will be rewarded.

OK. Database research in Asia. How is it different from database research in the rest of the world?

There's not much difference in the way we do research, but for Asians, it tends to be more algorithmic than system-oriented. So they build fewer systems because system development takes much longer time to materialize than just to focus on new problems and come out with solutions and show that they work.

Is that okay or is it a bad thing? That there's less systems research?

If we do not build systems, eventually we are driven to locate problems, and at times we end up creating artificial problems that could be new, novel, but on the other hand not applicable to the real world.

So system building is a reality check you would say.

Yes, that's true. There are problems and subproblems that evolve from the system. As we develop the system, we'll definitely encounter problems that cannot be managed or cannot be handled by the system that we tried to build.

OK. So I hear that you have played a role in raising the standard of database research in China. What have you done in that area and how do you see your role now?

I did not do as much as what has been said, but I do know most researchers in China very well and have talked with them often. I often advise them on what topics to move into and what not to do, how to protect the faculty's time, so that the faculty members can focus more on research.

Afternote: Beng Chin is an adjunct Chang Jiang professor at Zheajiang University.

So what should people not move into now?

They should move into user-driven research and try to build systems and address problems from the systems that they tried to develop for certain applications or for any other things that they have in mind to support.

And how can we protect faculty's time? What advice do you have for that?

For example, in China, they tend to organize more conferences. In order to organize conferences, you need faculty members to spend time organizing them and that will drain away a lot of a faculty member's time. Therefore, it's best to avoid organizing small conferences, those that have no consequences to the research quality. For NUS, we do not organize that many conferences. The last one we organized was VLDB, and we have not organized any for a long time.

OK. Would you recommend that new Computer Science Ph.D. graduates in the US and Europe consider a job in Asia?

Why not? Especially in Singapore. It's a very lovely and livable city. That's why you are here. [laughing].

To some degree, that's true. Your Ph.D. is from Australia, which follows the British system and the British offer this three-year Ph.D. with no coursework requirements. I've seen the kids coming out of these really short Ph.D. programs, and their resumes and abilities are about the same as that of a third-year graduate student in the U.S. On the other hand, U.S. Universities are very concerned because their Ph.D. students in CS can take five or six years to finish even though when they do finish they're ready to do research on their own. So at NUS, the CS degree's Ph.D. program is in between the US and the British system. You have personally worked to introduce courses and a written qualifying exam.

My question for you is: What is the right balance between the British system, which seems to me to be too short to teach people to do research, and then the American one, which takes a long time?

The graduate courses are necessary to provide a good foundation required for research. And of course, the qualifying exam (QE) is required since we take in students from different countries, with different standards, so we need the QE to weed out students who are not qualified enough to do Ph.D. With more courses to do, the students tend to take longer to graduate because they have to spend one year just to pass the courses and the QE, and have to do some courses in the second and third year. On the whole, it can take about four and a half years on average to graduate, but if a student works hard enough and works fast enough, he or she can still graduate within three years. It's not a problem. The real problem is those students who get very comfortable to stay on campus – after a while, they just do not want to leave and look for a job. If they can get an RA-ship, they hang on and just lead a student's life.

Is that bad? I mean you get to use them when they're really good at what they know: how to do research.

Yeah, that is good for faculty members. It may not be so good for the university because as a university, we want the students to leave as soon as they graduate because the resources are limited and we can use the resources to take in more students and educate more students.

OK. I've been told you're good at mining the strength of grad students even when the students are not very good. So how can you do good research with students who aren't very good? What's the secret? All of us have some students who are not so good. So how do we get good research out of students who are not so good?

That requires some time to understand the student's' strengths, to know what they can do, and from there we just ask them to do what they are good at. Of course, as a supervisor, I tend to guide them for the first few problems by telling them the likely solutions to solve certain problems. And after a while, they do learn the tricks, and as a requirement, they have to read ten papers a week in their first year and second year. And they have to write reports on the papers they read to me every Monday, and that builds up their foundation.

How long are the reports?

Just a few lines. It could be a couple of sentences, or it could be two or three paragraphs; but I told the students if they write the summaries based on the abstracts, I may call them up to get them to explain the papers to me.

So are these papers from all across the field? Or are these papers in the student's specific area?

They can read any papers they like, and sometimes I do point them to some papers that they have to read.

And so how long do you make them do this?

For the first two years.

That's a new tip for our readers.

So you have said that you run your students like an army. Another person told me "Beng Chin is a hard master to his students, very tough with them, he will scold them and may even check their code. However, he also rewards them, drinks and feasts with them, and plays with them. Basketball." And I believe it because apparently, that's a new basketball injury [pointing to his injured finger]. So even at the cost of your personal health, you play with them. My question is, how do you strike a balance between these two roles, which are quite different?

When it comes to working, it's about work. And we have to be serious about it; so we are very clear about our objectives, why we are here, and what are our goals. So we just want to just achieve that. The students know their role, and the students know my role, and therefore we work quite well when it comes to work. Once we start playing, there's no differentiation between supervisor and student. That's how I injured my two fingers.

So can you give an example goal that you would give a student?

Now, since I develop systems, all my students have to take part in developing the systems that I develop, even if they are not related to their thesis. I see it as part of their training. So we meet regularly to discuss the systems that are we building and of course they have to deliver what they are responsible for.

Afternote: since the interview, Beng Chin and his group have released Apache SINGA (a distributed machine learning engine), completed in-memory big data system, epiC, and a new storage engine, UStore, that supports blockchains and collaborative analytics.

I see. So a goal might be to write this module that satisfies this spec. So what kind of goals do you have that are not related to system building, that are more things that would maybe show up on their resume? Do you also have goals in that area?

We do. For their thesis, I tend to assign them a topic, and they focus on that topic and look for problems to solve. Quite often I show them the problems, but most of the topics revolve around the systems we're trying to build.

> ***[my students] have to read ten papers a week in their first year and second year, write reports about those papers and send me every Monday [...]***

OK, got it.

Someone told me "Beng Chin drinks Chinese wine like a fish. Ask him which one is his favorite and whether he turns Chinese wine into good papers." Is there a conversion function?

I learned to drink Chinese wine only 10+ years ago when I started to go to China to recruit students. Initially, I disliked the taste, but after a while, I got used to it. Honestly, I don't like the taste of any alcohol. I just like the after effect.

Afternote: Beng Chin has stopped drinking for the Nth time.

[laughing] How does Chinese wine differ from other wine?

Chinese wine is very strong, and it's strong in smell too. It's very strong in the content of alcohol. Usually, the alcohol content is about 53-60%.

Wow. So we won't talk anymore about that. We'll have to go downstairs to that wine store in the basement after we do the interview.

So I hear that you are a skilled painter. What do you paint?

I did watercolor when I was young. I like painting since young. So when I had free time and when I felt inspired, I started to paint (when I was young). But when I came to Singapore, I took up Chinese painting. I attended night classes for three years. And I did draw some Chinese paintings, and one of the friends who got my painting is Raghu.

OK, congratulations Raghu! What did he have to do to get one?

He came at the right time.

One of Beng Chin Ooi's paintings

Do you have any words of advice for fledgling or mid-career database researchers or practitioners?

I will say take their work seriously, work hard, and once they put their heart into it, then they will do well. Work hard today and let tomorrow take care of itself.

OK, very good. Among all your past research, do you have a favorite piece of work?

That could be my Ph.D. work when I built a GIS system on top of an existing database system. So what I did was to extend an existing database system by developing a cartridge to handle GIS operations on top of it. That was my first system.

If you magically had enough extra time to do one additional thing at work that you're not doing now, what would it be?

I would say charity work because it's a good way to contribute back to society.

So it would be charity work within IT aspect or charity…

Just for the general public because there are many people out there who need our help and our time.

Excellent. So, what charities in Singapore – or anywhere – are you involved with?

Embarrassingly, none. Therefore, I feel quite bad.

All right! If you can change one thing about yourself as a computer science researcher, what would it be?

I should not have stopped programming. I should have continued with my programming, and after stopping for so many years, now I'm not good at it.

Well if you had continued programming, you would have had less time to do all this other stuff. The startup, the department building, all that …

That is true. Because of my other roles, that's why I gave up programming. I used to program until 10 or so years ago.

Do you miss it?

Yep.

Maybe when you retire?

When I retire, I will open a restaurant. I will open a bar, be a bar attendant…

Oh, a restaurant and bar. What kind of food will you offer? Because I did hear you are a good cook. So what kind of food will be at your restaurant?

Chinese food. Because I know how to cook many dishes and I tend to do reverse engineering when I go to a restaurant, trying to figure out what are the ingredients, how do they cook, and I try to experiment with the food that I ate.

OK. Great.

And why I'd like to be a bar attendant? Because I like to drink and therefore I can serve as I listen to stories. [laughing]

Well great, thank you for telling me stories today.

Report from the third workshop on Algorithms and Systems for MapReduce and Beyond (BeyondMR'16)

Foto N. Afrati
National Technical University of Athens, Greece
afrati@softlab.ntua.gr

Jan Hidders
Vrije Universiteit Brussel, Belgium
jan.hidders@vub.ac.be

Christopher Ré
Stanford University, USA
chrismre@cs.stanford.edu

Jacek Sroka
University of Warsaw, Poland
j.sroka@mimuw.edu.pl

Jeffrey Ullman
Stanford University, USA
ullman@cs.stanford.edu

ABSTRACT

This report summarizes the presentations and discussions of the third workshop on Algorithms and Systems for MapReduce and Beyond (BeyondMR'16). The BeyondMR workshop was held in conjunction with the 2016 SIGMOD conference in San Francisco, California, USA on July 1, 2016. The goal of the workshop was to bring together researchers and practitioners to explore algorithms, computational models, architectures, languages and interfaces for systems that need large-scale parallelization and systems designed to support efficient parallelization and fault tolerance. These include specialized programming and data-management systems based on MapReduce and extensions, graph processing systems, data-intensive workflow and dataflow systems. The program featured two very well attended invited talks by Ion Stoica from AMPLab, University of California Berkeley and Carlos Guestrin from the University of Washington.

1. INTRODUCTION

The third BeyondMR workshop explored algorithms, computational models, architectures, languages and interfaces for systems that need large-scale parallelization and systems designed to support efficient parallelization and fault tolerance. The list of covered topics includes specialized programming and data-management systems based on MapReduce and extensions, graph processing systems, data-intensive workflow and dataflow systems.

After moving from EDBT to SIGMOD, the workshop successfully attracted 19 submission from which the program committee led by Christopher Ré from University of Stanford accepted 5 regular and 5 short papers.

2. REVIEW OF PRESENTED WORK

The proceedings of the workshop were published in the ACM Digital Library [1]. Below we present a short overview of the results followed, in Section 3, by a summary of two highly attended keynotes.

(Short Paper) Bridging the gap: Towards optimization across linear and relational algebra

Andreas Kunft from TU Berlin, Germany presented this paper [9] on behalf of co-authors Alexander Alexandrov, Asterios Katsifodimos and Volker Markl. Data cleaning and preprocessing is typically an initial step of advanced data analysis pipelines. As a theoretical foundation for the first step a relational algebra is used and for the second step a linear algebra. The authors propose to unify those two algebras into a common theoretical foundation. They explore and reason about optimizations across the two algebras in a suitable intermediate language representation. They propose *Lara DSL*, which is embedded in Scala and offers abstract data types for both algebras, i.e., bags and matrices. They also show-case the added benefits of unification and the optimizations that come thereof. A number of holistic optimizations are derived from the unified formal model and implemented under the assumption of a full view of the algorithm code including matrix blocking through joins and row-wise aggregation pushdown.

(Short Paper) Faucet: a user-level, modular technique for flow control in dataflow engines

Andrea Lattuada from the Systems Group, ETH Zürich, Switzerland presented this paper [10] on behalf of co-authors Frank McSherry and Zaheer

Chothia. This short paper introduces *Faucet*, which is a modular control flow approach for organizing distributed dataflow processing with arbitrary topologies including cyclicity. The advantages of Faucet over backpressure techniques are: (i) the implementation only relies on existing progress information exposed by the system and does not require changes to the underlying dataflow system, (ii) it can be applied selectively to certain parts of the dataflow graph, and (iii) it is designed to support a wide variety of use cases, topologies and workloads. The authors have tested their implementation on an example where variability in rates of produced and consumed tuples challenges the flow control techniques employed by systems like Storm, Heron, and Spark. They were able to keep the computation stable and resource bound while introducing at most 20% runtime overhead over an unconstrained implementation.

(Short Paper) Model-Centric Computation Abstractions in Machine Learning Applications

Judy Qiu from Indiana University, Bloomington, USA presented this paper [22] on behalf of co-authors Bingjing Zhang and Peng Bo. This paper considers parallel machine learning as a combination of training data-centric and model parameter-centric processing. It first presents four types of data-centric computation models for distributed machine learning, where they types are characterized by (1) whether the access of the parallel workers to the parameter models is synchronized or not, and if it is how the order of access is determined and (2) whether the workers get access to only the latest model parameters or also to stale model parameters. Several existing systems for distributed machine learning are analyzed and classified according to the presented types of computation models. Subsequently new model-centric abstractions are introduced to improve model update rate and increase model convergence speed. The effectiveness of these abstractions is demonstrated by using Latent Dirichlet Allocation (LDA) as an example, and experimental results show that an efficient parallel model update pipeline can achieve similar or higher model convergence speed compared to existing work.

(Regular Paper) DFA Minimization in Map-Reduce

Gösta Grahne from Concordia University, Montreal, Canada presented this paper [6] on behalf of co-authors Shahab Harrafi, Iraj Hedayati and Ali Moallemi. It features MapReduce implementations of two of the most prominent DFA minimization methods, namely Moore's and Hopcroft's algorithms. Extensive experiments, on various types of DFA's, with up to 217 states, validate that the MapReduce implementation of Hopcroft's algorithm is more efficient, both in terms of running time and communication cost. It was also confirmed that both algorithms are sensitive to skewed input, the Hopcroft's algorithm being intrinsically so.

(Regular Paper) Cross-System NoSQL Data Transformations with NotaQL

Johannes Schildgen from the University of Kaiserslautern presented this paper [16] on behalf of co-authors Thomas Lottermann and Stefan Deßloch. This full paper presents the language NotaQL which allows to concisely express transformations between different NoSQL data formats as are found in wide-column stores, document stores, key-value stores and even CSV files. The language supports a range of input and output formats, as well as different transformation engines for these formats. The language is output-oriented in the sense that the output format determines the structure of the transformation expressions. Finally, the paper presents an implementation of this language based on Apache Spark.

(Regular Paper) On Exploring Efficient Shuffle Design for In-Memory MapReduce

Haronubo Daikoku from University of Tsukuba, Japan presented this paper [4] co-authored with Hideyuki Kawashima and Osamu Tatebe. The authors have studied the efficiency of shuffle phase in MapReduce type systems that run on supercomputer hardware with shared-memory multiprocessor like InfiniBand. There are several design decisions which need to be made in such implementations to adapt MapReduce from commodity hardware communicating over Ethernet to specialized hardware relaying on MPI-based communication. The authors have implemented their own in-memory MapReduce system in C/C++ and used it to compare the efficiency of the data exchange algorithms in the shuffle phase. Specifically they have tested a fully-connected algorithm that mimics standard MapReduce solutions where each reduce process maintains a link to all map processes and a pairwise algorithm where in subsequent steps the processes communicate in pairs. They also have analyzed the effect on shuffle phase of the Remote Direct Memory Access (RDMA) mechanism which enables one machine to read and write data on the local memory of another.

(Short Paper) Toward Elastic Memory Management for Cloud Data Analytics

Jingjing Wang from University of Washington, USA presented this paper [19] co-authored with Magdalena Balazinska. The short paper discusses elastic memory management in modern Big data systems. It starts with demonstrating the negative impact of GC on the execution time of data analytics queries in a modern, Java-based system and shows how changing the heap size directly impacts the execution time. Then, it describes how to modify the JVM to enable dynamic modifications of the application heap layout and thus allow elastic management of its memory utilization. Next, it presents a machine-learning based technique for predicting the GC overhead for an application and whether that application is expected to run out of memory. Finally, an algorithm for dynamic memory management in a Big data analytics system is discussed.

(Regular Paper) Some-Pairs Problems

Jeffrey Ullman from Stanford University, USA presented this paper [17] co-authored with Jonathan Ullman. The paper considers the "some pairs" problem, where we are given two sets X and Y, and wish to detect the presence of pairs (x, y), one from each set, that meet some criterion, e.g., x and y are sufficiently close according to some distance measure. This paper looks at MapReduce algorithms for solving such a problem, in particular looking at the reducer-size vs. replication-rate tradeoff from Sarma et al, VLDB, 2013 [15]. There are two obvious approaches: (1) assume you care about all pairs and don't worry about taking advantage of the fact that you only care about some pairs (2) use one reducer for each of the pairs you care about, and nothing else. The principal result of the paper is a proof that for any X and Y and subset of the pairs that you care about, there is no MapReduce algorithm that has a significantly better replication rate than the better of the two obvious approaches.

(Regular Paper) Tight Bounds on One- and Two-Pass MapReduce Algorithms for Matrix Multiplication

Prakash Ramanan from Wichita State University, Wichita, USA presented this paper [13] co-authored with Ashita Nagar. This paper studies one- and two-pass MapReduce algorithms for multiplying two matrices, and in particular the trade-off between communication cost and replication rate. For multiplying sparse matrices in one pass, it shows tight bounds on qr and wr^2 where q is the reducer size, r the replication rate and w the reducer workload. In fact, the work shows that the bound for qr follows from the bound for wr^2, which means that the latter is the stronger lower bound. Next, the paper considers two-pass algorithms, which have been shown to have less communication cost than one-pass algorithms, given a certain reducer size. For multiplying dense matrices it presents tight bounds on $q_f r_f r_s$ and $w_f r_f^2 r_s$, where the subscripts f and s correspond to the first and second pass, respectively. Using this bound on $q_f r_f r_s$, the paper presents a tight bound on the total communication cost as a function of g_f. The presented lower bounds hold for the two-pass algorithms that perform all the real-number multiplications in the first pass.

(Short Paper) Deterministic Load Balancing for Parallel Joins

Nivetha Singara Vadivelu from University of Wisconsin-Madison, USA presented this short paper [8] co-authored with Paraschos Koutris. This short paper discusses parallel joins and multiway joins where the input data is first distributed over r-dimensional hypercube and then blocks in this cube can be processed independently in parallel. There was already a lot of attention to this problem and efficient solutions were proposed that distribute the tuples by applying a random hash function to achieve with high probability the optimal load within polylogarithmic factor. In the paper the authors explore if it is possible to construct an efficient deterministic algorithm that distributes the tuples such that the load is always as close to the optimal value as possible. They also seek to obtain optimality guarantees under any skew conditions, and not only for the case of no data skew. A general lower bound is proposed for the load, which is based on maximum degrees of each value (or combination of values) in the relation. Then, two fast deterministic algorithms are presented: one that is optimal within a constant factor of the lower bound for one dimensional case and another one that is optimal within a polylogarithmic factor of the lower bound for two dimensional case. The second one extends the first with application of algorithm for vector load balancing problem.

3. SUMMARY OF KEYNOTES

Now we give a summary of the two keynotes, which contributed to the visibility of the workshop.

(Keynote) Spark: Past, Present, and Future

In his keynote, Ion Stoica from AMPLab, University of California Berkeley gave an overview of the decisions that, apart of timing and luck, lead to the

success of the Spark project [21]. Besides an expressive API that allows to reduce by several times the length of code needed for typical tasks as compared to Hadoop, the main advantage of Spark is its effectiveness. It comes from leveraging hardware and workload trends like the rapid increase in memory capacity, so that working sets in Big Data clusters fit in memory. Further efficiency gains come from using threads rather than JVM processes and dealing with fault recovery with lineage rather than persistent storage. Those ideas lead to significant speed-up in many use cases, especially in interactive processing which is often needed in machine learning applications.

Then prof. Stoica gave an overview of the Spark subprojects: *Spark SQL* [2], *Spark Streaming* [20], *MLlib* [12], *GraphX* [5] and *SparkR* [18] and highlighted recent developments including structured APIs (*Datasets* and *DataFrames*), *project Tungsten* and structured streaming. The first phase of project Tungsten was enabled by DataFrames and results in 5-20× speed up. It exploits cache locality and employs off-heap memory management. The second phase of project Tungsten introduces whole-stage code generation, which removes expensive iterator calls and allows fusing across multiple operators.

The future of Spark brings improvements in performance due to fine grain updates with *IndexedRDDs*, reducing latency with batch scheduling, generality with fine grain task computations, and finally easy of use.

(Keynote) Big Data, Small Cluster: Choosing 'big memory' (RAM, disks, SSDs) over big clusters

Carlos Guestrin from the University of Washington was the second keynote speaker. Although initially he was interested in constructing killer robots, his presentation was about the ideas from the database community that have surfaced in the machine learning community and influence its progress. The first of those ideas is columnar storage and compression of stored data, which allowed for huge speed improvements in projects that Carlos worked on in the past like GraphLab [11]. The second idea is the quantile sketch technique [7, 23] that was adapted to weighted datasets to allow for the development of the scalable end-to-end tree boosting system [3].

Next, he presented an analogy between the current aim in machine learning community to create composites systems and Database Managements System (DBMS) that freed database users from technical decisions like optimizing queries, planing indexes and their usage or using materialized views. Building a machine learning solution nowadays also requires many technical decisions and skills like model selection, parameter selection or implementing distributed execution on a cluster. It would be convenient if a composite solution would be created with build in machine learning algorithms, so that the user could only declare what he needs and the system would analyze the data and make appropriate technical decisions for him.

Finally, the last idea presented by the speaker is related to *provenance*. As machine learning adoption is sometimes slowed down, by lack of *trust* in the results [14], it would be helpful to understand the predictions and get their explanations. This would also allow to avoid the situations where the accuracy percentage is high but the features that are used to achieve this accuracy are only properties of the training set, and in practical scenarios this would not generalize and could lead to incorrect behavior. Furthermore, also in situations where the results are correct, the users would profit from understanding how they were achieved, e.g., that the user could like this new movie because he liked some other or that a patient should be diagnosed with a disease because this is suggested by a given subset of his medical examination results.

4. CONCLUSION

The presentations and keynotes at BeyondMR'16 provided an overview of current developments and emerging issues in the domain of algorithms, computational models, architectures, languages and interfaces for systems that need large-scale parallelization and systems designed to support efficient parallelization and fault tolerance. These proceedings suggest that while MapReduce was replaced by new models, there is an active area of research centered around data-management systems based on MapReduce and extensions, graph processing systems, data-intensive workflow and dataflow systems.

Acknowledgements: We would like to thank the PC members, keynote speakers, authors, local workshop organizers and attendees for making BeyondMR'16 a successful workshop. We also express our great appreciation for the support from Google Inc.

5. REFERENCES

[1] Foto N. Afrati, Jacek Sroka, and Jan Hidders, editors. *Proceedings of the 3rd ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond, BeyondMR@SIGMOD 2016, San Francisco,*

CA, USA, July 1, 2016. ACM, 2016. http://doi.acm.org/10.1145/2926534.

[2] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, and Matei Zaharia. Spark SQL: relational data processing in spark. In Timos K. Sellis, Susan B. Davidson, and Zachary G. Ives, editors, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 1383–1394. ACM, 2015.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[4] Harunobu Daikoku, Hideyuki Kawashima, and Osamu Tatebe. On exploring efficient shuffle design for in-memory mapreduce. In Afrati et al. [1], page 6. http://doi.acm.org/10.1145/2926534.2926538.

[5] Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. Graphx: Graph processing in a distributed dataflow framework. In Jason Flinn and Hank Levy, editors, *11th USENIX Symposium on Operating Systems Design and Implementation, OSDI '14, Broomfield, CO, USA, October 6-8, 2014.*, pages 599–613. USENIX Association, 2014.

[6] Gösta Grahne, Shahab Harrafi, Iraj Hedayati, and Ali Moallemi. DFA minimization in map-reduce. In Afrati et al. [1], page 4. http://doi.acm.org/10.1145/2926534.2926537.

[7] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, pages 58–66, New York, NY, USA, 2001. ACM.

[8] Paraschos Koutris and Nivetha Singara Vadivelu. Deterministic load balancing for parallel joins. In Afrati et al. [1], page 10. http://doi.acm.org/10.1145/2926534.2926536.

[9] Andreas Kunft, Alexander Alexandrov, Asterios Katsifodimos, and Volker Markl. Bridging the gap: towards optimization across linear and relational algebra. In Afrati et al. [1], page 1. http://doi.acm.org/10.1145/2926534.2926540.

[10] Andrea Lattuada, Frank McSherry, and Zaheer Chothia. Faucet: a user-level, modular technique for flow control in dataflow engines. In Afrati et al. [1], page 2. http://doi.acm.org/10.1145/2926534.2926544.

[11] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Graphlab: A new framework for parallel machine learning. In Peter Grünwald and Peter Spirtes, editors, *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010*, pages 340–349. AUAI Press, 2010.

[12] Xiangrui Meng, Joseph K. Bradley, Burak Yavuz, Evan R. Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D. B. Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. Mllib: Machine learning in apache spark. *CoRR*, abs/1505.06807, 2015.

[13] Prakash Ramanan and Ashita Nagar. Tight bounds on one- and two-pass mapreduce algorithms for matrix multiplication. In Afrati et al. [1], page 9. http://doi.acm.org/10.1145/2926534.2926542.

[14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, 2016. ACM.

[15] Anish Das Sarma, Foto N. Afrati, Semih Salihoglu, and Jeffrey D. Ullman. Upper and lower bounds on the cost of a map-reduce computation. In *Proceedings of the 39th international conference on Very Large Data Bases*, PVLDB'13, pages 277–288. VLDB Endowment, 2013.

[16] Johannes Schildgen, Thomas Lottermann, and Stefan Deßloch. Cross-system NoSQL data transformations with NotaQL. In Afrati et al. [1], page 5. http://doi.acm.org/10.1145/2926534.2926535.

[17] Jeffrey D. Ullman and Jonathan R. Ullman. Some pairs problems. In Afrati et al. [1], page 8. http://doi.acm.org/10.1145/2926534.2926543.

[18] Shivaram Venkataraman, Zongheng Yang, Davies Liu, Eric Liang, Hossein Falaki, Xiangrui Meng, Reynold Xin, Ali Ghodsi,

Michael J. Franklin, Ion Stoica, and Matei Zaharia. Sparkr: Scaling R programs with spark. In Fatma Özcan, Georgia Koutrika, and Sam Madden, editors, *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 1099–1104. ACM, 2016.

[19] Jingjing Wang and Magdalena Balazinska. Toward elastic memory management for cloud data analytics. In Afrati et al. [1], page 7. `http://doi.acm.org/10.1145/2926534.2926541`.

[20] Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. Discretized streams: fault-tolerant streaming computation at scale. In Michael Kaminsky and Mike Dahlin, editors, *ACM SIGOPS 24th Symposium on Operating Systems Principles, SOSP '13, Farmington, PA, USA, November 3-6, 2013*, pages 423–438. ACM, 2013.

[21] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache spark: a unified engine for big data processing. *Commun. ACM*, 59(11):56–65, 2016.

[22] Bingjing Zhang, Bo Peng, and Judy Qiu. Model-centric computation abstractions in machine learning applications. In Afrati et al. [1], page 3. `http://doi.acm.org/10.1145/2926534.2926539`.

[23] Qi Zhang and Wei Wang. A fast algorithm for approximate quantiles in high speed data streams. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management*, SSDBM '07, pages 29–29, Washington, DC, USA, 2007. IEEE Computer Society.