Data partitioning for single-round multi-join evaluation in massively parallel systems*

Tom J. Ameloot*
Hasselt University & transnational University of Limburg
tom.ameloot@uhasselt.be

Gaetano Geck
TU Dortmund University
gaetano.geck@udo.edu

Bas Ketsman†
Hasselt University & transnational University of Limburg
bas.ketsman@uhasselt.be

Frank Neven
Hasselt University & transnational University of Limburg
frank.neven@uhasselt.be

Thomas Schwentick
TU Dortmund University
thomas.schwentick@udo.edu

ABSTRACT

A dominant cost for query evaluation in modern massively distributed systems is the number of communication rounds. For this reason, there is a growing interest in single-round multiway join algorithms where data is first reshuffled over many servers and then evaluated in a parallel but communication-free way. The reshuffling itself is specified as a distribution policy. We introduce a correctness condition, called *parallel-correctness*, for the evaluation of queries w.r.t. a distribution policy. We provide a semantical characterization for when conjunctive queries (and extensions thereof) are parallel-correct and give matching complexity bounds for the associated decision problem.

Motivated by scenarios for workload optimization, we further consider the problem of parallel-correctness transfer from a query $\mathcal{Q}$ to a query $\mathcal{Q}'$, that is, whether $\mathcal{Q}'$ is parallel-correct for all distribution policies for which $\mathcal{Q}$ is parallel-correct. In this case, $\mathcal{Q}'$ can always be evaluated after $\mathcal{Q}$ *without* repartitioning the data. We provide a semantical characterization for parallel-correctness transfer and provide matching complexity bounds for the associated decision problem for conjunctive queries (and extensions). Finally, we investigate restrictions of queries and families of distribution policies with better complexities, including, for instance, the Hypercube distributions.

1. INTRODUCTION

The background scenario for our work is that of large-scale data analytics where massive parallelism is utilized to answer complex join queries. As, for instance, described by Chu et al. [6], data analytics engines face new kinds of workloads, where multiple large tables are joined, or where the query graph has cycles. Furthermore, recent in-memory systems (like, e.g., [9, 10, 16, 19]) can fit data in main memory by utilizing multitudes of servers. For such systems, performance is no longer dominated by the number of I/O requests to external memory as in traditional systems but by the communication cost for reshuffling data during query execution. When queries need to be evaluated in several rounds, such reshuffling can repartition the whole database and can thus be very expensive. For this reason, it is paramount to reduce the number of evaluation rounds.

While in traditional distributed query evaluation, multi-join queries are computed in several stages over a join tree possibly transferring data over the network at each step, we focus in this paper on query evaluation algorithms that only require *one* round of communication.[1] Such algorithms consist of two phases: a *distribution phase* (where data is repartitioned or reshuffled over the servers) followed by a naive *evaluation phase* where each server contributes to the query answer in isolation by evaluating the query at hand over the local data without any further communication. We refer to such algorithms as *one-round algorithms*. Afrati and Ullman [1] describe an algorithm that computes a multi-join query in a *single* communication round. The algorithm uses a technique that can be traced back to Ganguli, Silberschatz, and Tsur [7] and received quite some attention in the literature. For instance, Beame et al. [4, 5] refined the algorithm, named it *Hypercube*, and showed that it is a communication-optimal algorithm for distributed evaluation, while in a subsequent study, Chu et al. performed an empirical evaluation of Hypercube [6].

Specifically, for a given conjunctive query $\mathcal{Q}$, Hypercube defines a reshuffling of the data over the available servers: the algorithm organizes all the servers in a hypercube and assigns each fact to a subset of points within this cube. The Hypercube reshuffling is thus defined on the granularity of facts and assigns each fact in isolation (that is, independent of the presence or absence of any other facts) to a subset of the servers. This means that the Hypercube reshuffling is independent of the current distribution of the data and can therefore be applied locally at every server. Furthermore, once the data is repartitioned, the target query $\mathcal{Q}$ can be

*The original version of this article was published in PODS 2015 as [2].

*Postdoctoral Fellow of the Research Foundation - Flanders (FWO).

†PhD Fellow of the Research Foundation - Flanders (FWO).

[1]The novel algorithms, introduced, for instance, in [13] or [18] do process multiple joins at once but are targeted towards a sequential setting.

naively evaluated independently on all servers. Hence, Hypercube is a one-round algorithm. We note that the common term Hypercube (algorithm) refers to the described combination of Hypercube distribution/reshuffling followed by naive evaluation of the query at each server. For this reason, the "hypercube" aspect relates exclusively to the way the data is initially reshuffled. Beame et al. [5] obtained that choosing the optimal shape of the hypercube is related to the fractional edge packing of the query graph.

We present a framework for reasoning about the correctness of one-round algorithms for the evaluation of queries under *arbitrary* distribution policies. To target the widest possible range of repartitioning strategies, the initial distribution phase is modeled by a distribution policy that can be *any* mapping from facts to subsets of servers. In particular, this includes any primary horizontal fragmentation of the database as for instance hash partitioning or range partitioning [14].

In this setting, we study two fundamental problems:

Parallel-Correctness: Given a distribution policy and a query, can we be sure that the corresponding one-round algorithm will always compute the query result correctly — no matter the actual data?

Parallel-Correctness Transfer: Given two queries $\mathcal{Q}$ and $\mathcal{Q}'$, can we infer from the fact that $\mathcal{Q}$ is computed correctly under the current distribution policy, that $\mathcal{Q}'$ is computed correctly as well?

Apart from their fundamental nature, the just mentioned problems have practical relevance in settings where a workload of queries has to be evaluated. Recall that the naive one-round Hypercube algorithm requires a reshuffling of the base data for *every* separate query, it therefore makes sense to consider scenarios for which this reshuffling can be avoided. It is in this context that we consider the problem of parallel-correctness transfer. Specifically, parallel-correctness allows to decide whether a query can be evaluated on the current distribution without reshuffling the data, that is, with zero communication cost. Furthermore, we say that parallel-correctness *transfers* from $\mathcal{Q}$ to $\mathcal{Q}'$, denoted $\mathcal{Q} \xrightarrow{\text{pc}} \mathcal{Q}'$, when $\mathcal{Q}'$ is parallel-correct under *every* distribution policy for which $\mathcal{Q}$ is parallel-correct. This implies that $\mathcal{Q}'$ can *always* be evaluated after $\mathcal{Q}$ *without* redistributing the data. Therefore parallel-correctness transfer is particularly relevant in a setting of automatic data partitioning where an optimizer tries to automatically partition the data across multiple nodes to achieve overall optimal performance for a specific workload of queries (see, e.g., [12, 15]). Indeed, when parallel-correctness transfers from a query $\mathcal{Q}$ to a set of queries $\mathcal{S}$, then any distribution policy under which $\mathcal{Q}$ is parallel-correct can be picked to evaluate all queries in $\mathcal{S}$ without reshuffling the data. Of course, parallel-correctness transfer is a very strong notion. In our view, parallel-correctness transfer relates to optimization of data partitioning like query containment relates to traditional query optimization: a relevant fundamental property but perhaps too strong for immediate use.

We focus in this paper on conjunctive queries (possibly extended with union, inequalities and negation) and first consider the complexity of deciding parallel-correctness. The latter problem is equivalent to testing whether the distribution policy *saturates* the query, that is, whether for every *minimal* valuation of the conjunctive query there is a node in the network containing all facts required by that valuation. For various representations of distribution policies, testing parallel-correctness is Π_2^P-complete. These results continue to hold in the presence of union and inequalities. When negation is added, deciding parallel-correctness can no longer be reduced to testing properties of minimal valuations but might involve counterexample databases of exponential size. More specifically, in the presence of negation deciding parallel-correctness is coNEXPTIME-complete. Interestingly, the latter result is related to the new result that query containment for conjunctive queries with negation is coNEXPTIME-complete, as well.

Parallel-correctness transfer can be semantically characterized. In particular, $\mathcal{Q} \xrightarrow{\text{pc}} \mathcal{Q}'$ if and only if $\mathcal{Q}$ *covers* $\mathcal{Q}'$. The latter is a (value-based) containment condition for minimal valuations of $\mathcal{Q}'$ and $\mathcal{Q}$. Deciding transferability of parallel-correctness for conjunctive queries is Π_3^P-complete, again even in the presence of unions and inequalities. We note that the implied exponential time algorithm for parallel-correctness transfer does not rule out practical applicability since the running time is exponential in the size of the queries not in the size of a database. Still, it would be interesting to lower the complexity. Therefore, we consider a further condition that is merely necessary for parallel-correctness transfer, but can be tested more easily, in NP. We refer to this condition as *weak parallel-correctness transfer* and consider two settings where this necessary condition is also sufficient. The first setting applies when $\mathcal{Q}$ is strongly minimal. The second setting considers particular families of distribution policies which generalize the Hypercube distribution families. In particular this means that it is NP-complete to decide whether a query $\mathcal{Q}'$ is parallel-correct under all Hypercube distributions for a query $\mathcal{Q}$.

Outline. In Section 2, we introduce the necessary preliminaries regarding databases, queries, and distribution policies, including Hypercube distributions. In Section 3 and Section 4, we discuss parallel-correctness and parallel-correctness transfer. In Section 5, we consider the setting with lower complexity. We present concluding remarks together with direction for further research in Section 6. Although most of the results hold in the presence of union and inequality, the presentation will focus on CQs most of the time.

2. PRELIMINARIES

2.1 Databases and queries

Throughout the rest of the paper, we assume an infinite domain **dom** and a database scheme consisting of relation names with associated arities. A *(database) instance* I is simply a finite set of facts. A *conjunctive query (CQ)* $\mathcal{Q}$ is an expression of the form

$$H(\mathbf{x}) \leftarrow R_1(\mathbf{y_1}), \ldots, R_m(\mathbf{y_m})$$

where every R_i is a relation name and every $\mathbf{y_i}$ matches the arity of R_i. We require that every variable in $\mathbf{x}$ occurs in some $\mathbf{y}_i$. We refer to the *head atom* $H(\mathbf{x})$ by $head_{\mathcal{Q}}$ and to the set $\{R_1(\mathbf{y_1}), \ldots, R_m(\mathbf{y_m})\}$ by $body_{\mathcal{Q}}$.

We denote by $vars(\mathcal{Q})$ the set of all variables occurring in $\mathcal{Q}$. A *valuation* for a CQ $\mathcal{Q}$ is a total function $V : vars(\mathcal{Q}) \to \mathbf{dom}$. We refer to $V(body_{\mathcal{Q}})$ as the facts *required* by V. A valuation V *satisfies* $\mathcal{Q}$ on instance I if all facts required by

V are in I. In that case, V *derives* the fact $V(head_{\mathcal{Q}})$. The *result* of $\mathcal{Q}$ on instance I, denoted $\mathcal{Q}(I)$, is defined as the set of facts that can be derived by satisfying valuations for $\mathcal{Q}$ on I.

Example 2.1. Let I_e be the example database instance

$$\{R(a,b), R(b,a), R(b,c), S(a,a), S(c,a)\},$$

and $\mathcal{Q}_e$ be the example CQ

$$H(x_1,x_3) \leftarrow R(x_1,x_2), R(x_2,x_3), S(x_3,x_1).$$

Then $V_1 = \{x_1 \mapsto a, x_2 \mapsto b, x_3 \mapsto a\}$ and $V_2 = \{x_1 \mapsto a, x_2 \mapsto b, x_3 \mapsto c\}$ are the only satisfying valuations. Consequently, $\mathcal{Q}_e(I_e) = \{H(a,a), H(a,c)\}$. □

We denote the class of all CQs by **CQ**.

2.2 Distribution policies

A *network* $\mathcal{N}$ is a nonempty finite set of node names. A *distribution policy* $\boldsymbol{P} = (U, rfacts_{\boldsymbol{P}})$ for a network $\mathcal{N}$ consists of a universe U and a total function $rfacts_{\boldsymbol{P}}$ that maps each node of $\mathcal{N}$ to a set of facts in $facts(U)$.[2] Here, $facts(U)$ denotes the set of all possible facts over U. A node κ is *responsible for a fact* $\boldsymbol{f}$ (under policy $\boldsymbol{P}$) if $\boldsymbol{f} \in rfacts_{\boldsymbol{P}}(\kappa)$. For a distribution policy $\boldsymbol{P}$, an instance I over $\mathcal{D}$ and a $\kappa \in \mathcal{N}$, let $loc\text{-}inst_{\boldsymbol{P},I}(\kappa)$ denote $I \cap rfacts_{\boldsymbol{P}}(\kappa)$, that is, the set of facts in I for which node κ is responsible. We refer to a given instance I as the *global instance* and to $loc\text{-}inst_{\boldsymbol{P},I}(\kappa)$ as the *local instance on node* κ.

The result $[\mathcal{Q}, \boldsymbol{P}](I)$ of the distributed evaluation in one round of a query $\mathcal{Q}$ on an instance I under a distribution policy $\boldsymbol{P}$ is defined as the union of the results of $\mathcal{Q}$ evaluated over every local instance. Formally,

$$[\mathcal{Q}, \boldsymbol{P}](I) \stackrel{\text{def}}{=} \bigcup_{\kappa \in \mathcal{N}} \mathcal{Q}(loc\text{-}inst_{\boldsymbol{P},I}(\kappa)).$$

Example 2.2. Continuing Example 2.1, consider a network $\mathcal{N}_e$ consisting of two nodes $\{\kappa_1, \kappa_2\}$. Let $\boldsymbol{P}_1 = (\{a,b,c\}, rfacts_{\boldsymbol{P}})$ be the distribution policy that assigns all R-facts to both nodes κ_1 and κ_2, and every fact $S(d_1,d_2)$ to node κ_1 when $d_1 = d_2$ and to node κ_2 otherwise. Then,

$$loc\text{-}inst_{\boldsymbol{P}_1,I_e}(\kappa_1) = \{R(a,b), R(b,a), R(b,c), S(a,a)\},$$

and

$$loc\text{-}inst_{\boldsymbol{P}_1,I_e}(\kappa_2) = \{R(a,b), R(b,a), R(b,c), S(c,a)\}.$$

Furthermore,

$$[\mathcal{Q}_e, \boldsymbol{P}_1](I_e) = \mathcal{Q}_e\big(loc\text{-}inst_{\boldsymbol{P}_1,I_e}(\kappa_1)\big) \cup \mathcal{Q}_e\big(loc\text{-}inst_{\boldsymbol{P}_1,I_e}(\kappa_2)\big),$$

which is just $\{H(a,b)\} \cup \{H(a,c)\}$.

Next, consider the alternative distribution policy $\boldsymbol{P}_2$ that assigns all R-facts to node κ_1 and all S-facts to node κ_2, then $[\mathcal{Q}_e, \boldsymbol{P}_2](I_e) = \emptyset$. □

Obviously, every primary horizontal fragmentation (see, e.g., [14]) can be modeled as a distribution policy. Consider, for instance, a range partitioning on a relation Customer that assigns tuples to network nodes determined by a threshold on the area code.

[2]We mention that for Hypercube distributions, the view is reversed: facts are assigned to nodes. However, both views are essentially equivalent and we will freely adopt the view that fits best for the purpose at hand.

2.3 Hypercube distributions

A Hypercube distribution distributes the space of all valuations of $\mathcal{Q}$ over the computing servers in an instance independent way through hashing of domain values. Let $\mathcal{Q}$ be a CQ of the following form

$$H(\mathbf{x}) \leftarrow R_1(\mathbf{y_1}), \ldots, R_m(\mathbf{y_m}),$$

and let $|\mathcal{N}| = p$. A Hypercube distribution is parameterized by numbers $p_1, \ldots, p_k$ for which $p = p_1 \times \cdots \times p_k$. Here, k is equal to the number of distinct variables in Q, that is, $|vars(\mathcal{Q})| = k$. For simplicity, assume $vars(\mathcal{Q}) = \{x_1, \ldots, x_k\}$. Still, there can be variables in $vars(\mathcal{Q})$ that do not occur in the head $H(\mathbf{x})$. The distribution organizes the p servers in a hypercube of k dimensions where the size of the i-th dimension is p_i. So, every server corresponds to one point in $\{1, \ldots, p_1\} \times \cdots \times \{1, \ldots, p_k\}$. Furthermore, for $i \in \{1, \ldots, k\}$, let h_i be a hash function mapping each value in **dom** to $\{1, \ldots, p_i\}$.

Then the Hypercube distribution policy $\boldsymbol{P}_H$ assigns each fact to a set of points in the hypercube and is defined as follows: each fact $R_i(a_1, \ldots, a_\ell)$ in the local instance that can be mapped to an atom $R_i(x_{i_1}, \ldots, x_{i_\ell})$ in $\mathcal{Q}$ is sent to every coordinate $(\alpha_1, \ldots, \alpha_k)$ for which $\alpha_{i_j} = h_{i_j}(a_j)$ for all j in $\{1, \ldots, \ell\}$. In particular, this means that when a variable x_r occurs in $R_i(x_{i_1}, \ldots, x_{i_\ell})$ on position j then the r-th dimension of the coordinate has to be $h_r(a_j)$; otherwise, it can be any value in the codomain of h_r.

Example 2.3. Consider query $\mathcal{Q}_e$

$$H(x_1,x_3) \leftarrow R(x_1,x_2), R(x_2,x_3), S(x_3,x_1).$$

from Example 2.1, and a network $\mathcal{N}$ and numbers p_1, p_2, p_3 with $|\mathcal{N}| = p = p_1 \times p_2 \times p_3$. So, every computing node is addressed by a triple (i_1, i_2, i_3) with $i_j \in \{1, \ldots, p_j\}$ for $j \leq 3$. We abuse notation and denote, for instance, by $(i_1, i_2, \star)$ the set of coordinates that match on (i_1, i_2), that is, $\{(i_1, i_2, i_3) \mid i_3 \leq p_3\}$. Then, for any choice of hash functions, $\boldsymbol{P}_H$ assigns each

- $R(d_1, d_2)$ to
  - $\big(h_1(d_1), h_2(d_2), \star\big)$ because of $R(x_1, x_2)$; and
  - $\big(\star, h_2(d_1), h_3(d_2)\big)$ because of $R(x_2, x_3)$;
- $S(d_1, d_2)$ to $\big(h_1(d_2), \star, h_3(d_1)\big)$ because of $S(x_3, x_1)$.

It is not hard to see that, for each instance I,

$$\mathcal{Q}_e(I) = \bigcup_{\kappa \in \mathcal{N}} \mathcal{Q}_e\big(loc\text{-}inst_{\boldsymbol{P},I}(\kappa)\big) = [\mathcal{Q}_e, \boldsymbol{P}_H](I),$$

and thus the one-round algorithm evaluates $\mathcal{Q}_e$ correctly. We will refer to this property as parallel-correctness in the next section. □

3. PARALLEL-CORRECTNESS

In this section, we introduce the notion of parallel-correctness. Informally, it states for a query and a distribution policy that the naive one-round evaluation algorithm yields the correct result, for every possible instance. Specifically, this algorithm first distributes (reshuffles) the data over the computing nodes according to $\boldsymbol{P}$ and then evaluates $\mathcal{Q}$ in a subsequent parallel step at every computing node. Notice that, since $\boldsymbol{P}$ is defined on the granularity of a fact, the

reshuffling does not depend on the current distribution of the data and can be done in parallel as well.

First, we define parallel-correctness w.r.t. a given instance:

Definition 3.1. A query $\mathcal{Q}$ is *parallel-correct on instance I under distribution policy $\boldsymbol{P}$* if $\mathcal{Q}(I) = [\mathcal{Q}, \boldsymbol{P}](I)$.

We note that parallel-correctness is the combination of

- *parallel-soundness*: $[\mathcal{Q}, \boldsymbol{P}](I) \subseteq \mathcal{Q}(I)$, and
- *parallel-completeness*: $\mathcal{Q}(I) \subseteq [\mathcal{Q}, \boldsymbol{P}](I)$.

For monotone queries, like conjunctive queries, parallel-soundness is granted, and therefore parallel-correctness and parallel-soundness coincide. Next, we lift parallel-correctness to all instances:

Definition 3.2. A query $\mathcal{Q}$ is *parallel-correct under distribution policy $\boldsymbol{P} = (U, rfacts_{\boldsymbol{P}})$*, if $\mathcal{Q}$ is parallel-correct on all instances $I \subseteq facts\big(U\big)$.

While Definitions 3.1 and 3.2 are in terms of general queries, in the rest of this section, we only consider (extensions of) conjunctive queries.

3.1 Conjunctive queries

We first focus on a characterization of parallel-correctness. It is easy to see that a CQ $\mathcal{Q}$ is parallel-correct under distribution policy $\boldsymbol{P} = (U, rfacts_{\boldsymbol{P}})$ if, for each valuation for $\mathcal{Q}$, the required facts meet at some node. That is, if the following condition holds:

$$\text{For every valuation } V \text{ for } \mathcal{Q} \text{ over } U, \text{ there is a node } \kappa \in \mathcal{N} \text{ such that } V(body_{\mathcal{Q}}) \subseteq rfacts_{\boldsymbol{P}}(\kappa). \qquad (PC_0)$$

Even though Condition (PC_0) is sufficient for parallel-correctness, it is not necessary as the following example shows.

Example 3.3. We consider the CQ $\mathcal{Q}$,

$$H(x, z) \leftarrow R(x, y), R(y, z), R(x, x),$$

and the valuation $V = \{x \mapsto a, y \mapsto b, z \mapsto a\}$. Let further $\mathcal{N} = \{\kappa_1, \kappa_2\}$ and let $\boldsymbol{P}$ distribute every fact except $R(a, b)$ onto node κ_1 and every fact except $R(b, a)$ onto node κ_2. Since $R(a, b)$ and $R(b, a)$ do not meet under $\boldsymbol{P}$, valuation V witnesses the failure of Condition (PC_0) for $\boldsymbol{P}$ and $\mathcal{Q}$.

However, $\mathcal{Q}$ is parallel-correct under $\boldsymbol{P}$. Indeed, every valuation that derives a fact $\boldsymbol{f}$ with the help of the facts $R(a, b)$ and $R(b, a)$, also requires the fact $R(a, a)$ (or $R(b, b)$). But then, $R(a, a)$ (or $R(b, b)$) alone is sufficient to derive $\boldsymbol{f}$ by mapping all variables to a (or b). Therefore, if $\boldsymbol{f} \in \mathcal{Q}(I)$, for some instance I, then $\boldsymbol{f} \in [\mathcal{Q}, \boldsymbol{P}](I)$ and thus $\mathcal{Q}$ is parallel-correct under $\boldsymbol{P}$. □

It turns out that for a semantical characterization only such valuations have to be considered that are minimal in the following sense:

Definition 3.4. Let $\mathcal{Q}$ be a CQ. A valuation V for $\mathcal{Q}$ is *minimal* for $\mathcal{Q}$ if there does *not* exist a valuation V' for $\mathcal{Q}$ that derives the same head fact with a strict subset of body facts, that is, such that $V(body_{\mathcal{Q}}) \subsetneq V'(body_{\mathcal{Q}})$ and $V(head_{\mathcal{Q}}) = V'(head_{\mathcal{Q}})$.

Example 3.5. For a simple example of a minimal valuation and a non-minimal valuation, consider the CQ $\mathcal{Q}$,

$$H(x, z) \leftarrow R(x, y), R(y, z), R(x, x).$$

Both $V_1 = \{x \mapsto a, y \mapsto b, z \mapsto a\}$ and $V_2 = \{x \mapsto a, y \mapsto a, z \mapsto a\}$ are valuations for $\mathcal{Q}$. Notice that both valuations agree on the head variables of $\mathcal{Q}$, but they require different sets of facts. In particular, for V_1 to be satisfying on I, instance I must contain the facts $R(a, b), R(b, a)$, and $R(a, a)$, while V_2 only requires I to contain $R(a, a)$. This observation implies that V_1 is not minimal for $\mathcal{Q}$. Further, since V_2 requires only one fact for $\mathcal{Q}$, valuation V_2 is minimal for $\mathcal{Q}$. □

It turns out that it suffices to restrict valuations to minimal valuations in Condition (PC_0) to get a sufficient *and* necessary condition for parallel-correctness.

Proposition 3.6. *Let $\mathcal{Q}$ be a CQ. Then $\mathcal{Q}$ is parallel-correct under distribution policy $\boldsymbol{P}$ if and only if the following holds:*

$$\text{For every minimal valuation } V \text{ for } \mathcal{Q} \text{ over } U, \text{ there is a node } \kappa \in \mathcal{N} \text{ such that } V(body_{\mathcal{Q}}) \subseteq rfacts_{\boldsymbol{P}}(\kappa). \qquad (PC_1)$$

We emphasize that the word *minimal* is the only difference between Conditions (PC_0) and (PC_1).[3] The latter conditions are so fundamental when reasoning over parallel-correctness that they deserve their own terminology:

Definition 3.7. For a CQ $\mathcal{Q}$ and a distribution policy $\boldsymbol{P}$:

- $\boldsymbol{P}$ *saturates* $\mathcal{Q}$ if they fulfill Condition (PC_1); and,
- $\boldsymbol{P}$ *strongly saturates* $\mathcal{Q}$ if they fulfill Condition (PC_0).

Every Hypercube distribution $\boldsymbol{P}_H$ for a conjunctive query $\mathcal{Q}$ strongly saturates $\mathcal{Q}$. Indeed, consider Example 2.3. Then, for every valuation V, all facts

$$R\big(V(x_1), V(x_2)\big), R\big(V(x_2), V(x_3)\big), S\big(V(x_3), V(x_1)\big)$$

meet at node $\big(h_1(V(x_1)), h_2(V(x_2)), h_3(V(x_3))\big)$. Therefore, $\mathcal{Q}$ is parallel-correct under $\boldsymbol{P}_H$.

The quantifier structure in Condition (PC_1) hints at a Π_2^p upper bound for the complexity of testing parallel-correctness.[4] Of course, the exact complexity can not be judged without having a bound on the number of nodes κ and the complexity of the test $V(body_{\mathcal{Q}}) \subseteq rfacts_{\boldsymbol{P}}(\kappa)$. The largest classes of distribution policies for which we established the Π_2^p upper bound, are gathered in the set $\mathfrak{P}_{\text{npoly}}$ that contains classes $\mathcal{P}$ of distribution policies, for which each policy comes with an algorithm $\mathcal{A}$ and a bound n on the representation size of nodes in the network, respectively, such that whether a node κ is responsible for a fact $\boldsymbol{f}$ is decided by $\mathcal{A}$ *nondeterministically* in time $\mathcal{O}(n^k)$, for some k that depends only on $\mathcal{P}$.

It turns out that the problem of testing parallel-correctness is also Π_2^p-hard, even for the simple class $\mathcal{P}_{\text{fin}}$ of distribution policies, for which all pairs $(\kappa, \boldsymbol{f})$ of a node and a fact are

[3] We mention that Conditions (PC_0) were named (C_0) and (C_1), respectively, in [2].

[4] This holds, even if one takes into account that testing minimality of V requires an additional existential quantification of a valuation V' that might serve as a witness, in case V is not minimal.

explicitly enumerated. Thus, in a sense, Condition (PC_1) can essentially not be simplified.

To state the results more formally, we define the following two algorithmic problems.

PCI($\mathbf{CQ}, \mathcal{P}$)	
Input:	$\mathcal{Q} \in \mathbf{CQ}$, $\boldsymbol{P} \in \mathcal{P}$, instance I
Question:	Is $\mathcal{Q}$ parallel-correct on I under $\boldsymbol{P}$?

PC($\mathbf{CQ}, \mathcal{P}$)	
Input:	$\mathcal{Q} \in \mathbf{CQ}$, $\boldsymbol{P} \in \mathcal{P}$
Question:	Is $\mathcal{Q}$ parallel-correct under $\boldsymbol{P}$?

Theorem 3.8. *Problems* PC($\boldsymbol{CQ}, \mathcal{P}$) *and* PCI($\boldsymbol{CQ}, \mathcal{P}$) *are* Π_2^p*-complete, for every policy class* $\mathcal{P} \in \{\mathcal{P}_{fin}\} \cup \mathfrak{P}_{npoly}$.

The upper bounds follow from the characterization in Proposition 3.6 and the fact that pairs $(\kappa, \boldsymbol{f})$ can be tested in NP.

We note that Proposition 3.6 continues to hold true in the presence of union and inequalities (under a suitable definition of minimal valuation for unions of CQs) leading to the same complexity bounds as stated in Theorem 3.8 [8].

3.2 Conjunctive queries with negation

In this section, we consider conjunctive queries with negation. Specifically, queries can be of the form

$$H(\mathbf{x}) \leftarrow R_1(\mathbf{y_1}), \ldots, R_m(\mathbf{y_m}), \neg S_1(\mathbf{z_1}), \ldots, \neg S_n(\mathbf{z_n}).$$

To ensure safety, we require that every variable in $\mathbf{x}$ occurs in some $\mathbf{y}_i$ or $\mathbf{z}_j$, and that every variable occurring in a negated atom has to occur in a positive atom as well. A valuation V now derives a fact $H\big(V(\mathbf{x})\big)$ on an instance I if every positive atom $R_i\big(V(\mathbf{y_i})\big)$ occurs in I while none of the negative atoms $S_j\big(V(\mathbf{z_j})\big)$ do. We refer to the class of conjunctive queries with negation as $\mathbf{CQ}^\neg$.

We note that, as queries in $\mathbf{CQ}^\neg$ are no longer monotone, parallel-soundness is no longer guaranteed and thus parallel-correctness need not coincide with parallel-soundness.

We illustrate through an example that in the case of conjunctive queries *with negation*, the parallel-correctness problem becomes much more involved, since it might involve counterexample databases of exponential size. We emphasize that this exponential explosion can only occur if, as in our framework, the arity of the relations in the database schema are not a-priori bounded by some constant.

Example 3.9. Let $\mathcal{Q}$ be the following conjunctive query with negation:

$$\begin{aligned} H() \quad \leftarrow \quad & \texttt{Val}(w_0, w_0), \texttt{Val}(w_1, w_1), \neg\texttt{Val}(w_0, w_1), \\ & \texttt{Val}(x_1, x_1), \ldots, \texttt{Val}(x_n, x_n), \neg\texttt{Rel}(x_1, \ldots, x_n). \end{aligned}$$

Let $\boldsymbol{P}$ be the policy defined over universe $U = \{0, 1\}$ and two-node network $\{\kappa_1, \kappa_2\}$, which distributes all facts except $\texttt{Rel}(0, \ldots, 0)$ to node κ_1 and only fact $\texttt{Rel}(0, \ldots, 0)$ to node κ_2.

Query $\mathcal{Q}$ is not parallel-sound under policy $\boldsymbol{P}$, as witnessed by the counter-example $I \stackrel{\text{def}}{=} \{\texttt{Val}(0, 0), \texttt{Val}(1, 1)\} \cup \{\texttt{Rel}(a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in \{0, 1\}^n\}$. Indeed, $\mathcal{Q}(I) = \emptyset$ but $\mathcal{Q}(\textit{loc-inst}_{\boldsymbol{P},I}(\kappa_1)) \neq \emptyset$, as witnessed by the valuation that maps all variables to 0.

However, I has $2^n + 2$ facts and is a counter-example of minimal size as can easily be shown as follows. First, it is impossible that $\mathcal{Q}(I^*) \neq \emptyset$ and $\mathcal{Q}(\textit{loc-inst}_{\boldsymbol{P},I^*}(\kappa_1)) = \emptyset$, for any I^*, since $\texttt{Rel}(0, \ldots, 0)$ is the only fact that can be missing at node κ_1, and $\mathcal{Q}$ is antimonotonic with respect to $\texttt{Rel}$. On the other hand, if $\mathcal{Q}(\textit{loc-inst}_{\boldsymbol{P},I^*}(\kappa_1)) \neq \emptyset$, then the literals $\texttt{Val}(w_0, w_0), \texttt{Val}(w_1, w_1)$, and $\neg\texttt{Val}(w_0, w_1)$ ensure that there are at least two different data values (and thus 0 and 1) in I^*. But then $\mathcal{Q}(I^*) = \emptyset$ can only hold if all 2^n n-tuples over $\{0, 1\}$ are in I^*. □

Although this example requires an exponential size counter-example, in this particular case, the existence of the counter-example is easy to conclude. However, the following result shows that, in general, there is essentially no better algorithm than guessing an exponential size counter-example.

Theorem 3.10. *[8] For every class* $\mathcal{P} \in \mathfrak{P}_{npoly}$ *of distribution policies, the following problems are coNEXPTIME-complete.*

- Parallel-Sound($\boldsymbol{UCQ}^\neg, \mathcal{P}$)
- Parallel-Complete($\boldsymbol{UCQ}^\neg, \mathcal{P}$)
- Parallel-Correct($\boldsymbol{UCQ}^\neg, \mathcal{P}$)

The result and, in particular, the lower bound even holds if $\mathfrak{P}_{\text{npoly}}$ is replaced by the corresponding $\mathfrak{P}_{\text{poly}}$, where the decision algorithm for pairs $(\kappa, \boldsymbol{f})$ is deterministic and in polynomial time.

The proof of the lower bounds comes along an unexpected route and exhibits a reduction from query containment for $\mathbf{CQ}^\neg$ to parallel-correctness for $\mathbf{CQ}^\neg$. Specifically, query containment asks the question whether, given two queries $\mathcal{Q}$ and $\mathcal{Q}'$, it holds that $\mathcal{Q}(I) \subseteq \mathcal{Q}'(I)$ for all instances I. The latter is denoted by $\mathcal{Q} \subseteq \mathcal{Q}'$. It is shown in [8] that query containment for $\mathbf{CQ}^\neg$ is coNEXPTIME-complete, implying coNEXPTIME-hardness for parallel-correctness as well. The result regarding containment of $\mathbf{CQ}^\neg$ answers the observation in [11] that the Π_2^p-completeness result for query containment for $\mathbf{CQ}^\neg$ mentioned in [17] only holds for fixed database schemas (or a fixed arity bound, for that matter).

4. PARALLEL-CORRECTNESS TRANSFER

As mentioned in the introduction, the one-round Hypercube algorithm requires a reshuffling of the data before the evaluation of a new query. In the context of multiple query evaluation, where an optimizer tries to automatically partition the base data across multiple nodes to achieve overall optimal performance for a specific workload (see, e.g., [12, 15]), it makes sense to consider scenarios in which such reshuffling can be avoided. To this end, *parallel-correctness transfer* was introduced in [2] which states that a subsequent query $\mathcal{Q}'$ can always be evaluated over a distribution for which a query $\mathcal{Q}$ is parallel-correct.

Definition 4.1. For two queries $\mathcal{Q}$ and $\mathcal{Q}'$ over the same input schema, *parallel-correctness transfers from* $\mathcal{Q}$ *to* $\mathcal{Q}'$ if $\mathcal{Q}'$ is parallel-correct under every distribution policy for which $\mathcal{Q}$ is parallel-correct. In this case, we write $\mathcal{Q} \xrightarrow{\text{pc}} \mathcal{Q}'$.

Example 4.2. We illustrate parallel-correctness transfer with the help of the following example queries:

$$\begin{aligned} &\mathcal{Q}_1 : H() \leftarrow S(x), R(x, x), T(x). \\ &\mathcal{Q}_2 : H() \leftarrow R(x, x), T(x). \\ &\mathcal{Q}_3 : H() \leftarrow S(x), R(x, y), T(y). \\ &\mathcal{Q}_4 : H() \leftarrow R(x, y), T(y). \end{aligned}$$

Figure 1 (a) shows how these queries relate with respect to parallel-correctness transfer. As an example, $\mathcal{Q}_3 \xrightarrow{\text{pc}} \mathcal{Q}_1$. As Figure 1 (b) illustrates, this relationships is entirely orthogonal to query containment. Indeed, there are examples where parallel-correctness transfer and query containment coincide ($\mathcal{Q}_3$ vs. $\mathcal{Q}_4$), where they hold in opposite directions ($\mathcal{Q}_4$ vs. $\mathcal{Q}_2$) and where one but not the other holds ($\mathcal{Q}_3$ vs. $\mathcal{Q}_2$ and $\mathcal{Q}_1$ vs. $\mathcal{Q}_4$, respectively). □

It turns out that, just like parallel-correctness, parallel-correctness transfer can be characterized in terms of minimal valuations. For this, we need the following notion:

Definition 4.3. For two CQs $\mathcal{Q}$ and $\mathcal{Q}'$, we say that $\mathcal{Q}$ *covers* $\mathcal{Q}'$ if the following holds:

> for every minimal valuation V' for $\mathcal{Q}'$, there is a minimal valuation V for $\mathcal{Q}$, such that $V'(body_{\mathcal{Q}'}) \subseteq V(body_{\mathcal{Q}})$.

Proposition 4.4. *For two CQs $\mathcal{Q}$ and $\mathcal{Q}'$, parallel-correctness transfers from $\mathcal{Q}$ to $\mathcal{Q}'$ if and only if $\mathcal{Q}$ covers $\mathcal{Q}'$.*

Proposition 4.4, allows us to pinpoint the complexity of parallel-correctness transferability. For a formal statement we define the following algorithmic problem:

PC-TRANS (**CQ**)	
Input:	Queries $\mathcal{Q}$ and $\mathcal{Q}'$ from **CQ**
Question:	Does parallel-correctness transfer from $\mathcal{Q}$ to $\mathcal{Q}'$?

When the defining condition of "covers" is spelled out by rewriting "minimal valuations" one gets a characterization with a Π_3-structure. Again, it can be shown that this is essentially optimal.

Theorem 4.5. *Problem* PC-TRANS($\boldsymbol{CQ}$) *is Π_3^p-complete.*

The upper bounds follow directly from the characterization in Proposition 4.4, implying that these characterizations are essentially optimal. We note that the same complexity bounds continue to hold in the presence of inequalities and for unions of conjunctive queries [3].

5. LOWERING COMPLEXITY

In static analysis of conjunctive queries, one is used to face NP-complete algorithmic problems, most prominently, containment testing. Since queries are often small, especially compared with the data, NP-algorithms might still be helpful in query optimization. We saw in the previous two sections that the complexity of parallel-correctness and parallel-correctness transfer are higher than that, namely Π_2^p-complete and Π_3^p-complete, respectively, even without union, inequalities and negation.

In this section, we consider two settings, in which, for both problems, the complexity drops to the "usual" NP-complete level. In both cases, the complexity reduction is based on a simpler condition for parallel-correctness transfer which can be tested in NP. We state this condition next.

5.1 A necessary condition for pc-transfer

We use the following two additional notions, the first of which is the simpler condition for parallel-correctness transfer (in restricted settings), and the second characterizes the first, as we will see soon.

Definition 5.1. For CQs $\mathcal{Q}, \mathcal{Q}'$ and a distribution policy $\boldsymbol{P}$:

- parallel correctness *weakly transfers* from a CQ $\mathcal{Q}$ to $\mathcal{Q}'$, if $\mathcal{Q}'$ is parallel-correct under every policy $\boldsymbol{P}$ that strongly saturates $\mathcal{Q}$;[5] and
- $\mathcal{Q}$ *weakly covers* $\mathcal{Q}'$, if there are mappings ρ and θ such that
  - ρ maps the variables of $\mathcal{Q}$ to some variables (ρ is a *substitution*),
  - θ maps the variables of $\mathcal{Q}'$ to variables of $\mathcal{Q}'$ such that $head_{\theta(\mathcal{Q}')} = head_{\mathcal{Q}'}$ and $body_{\theta(\mathcal{Q}')} \subseteq body_{\mathcal{Q}'}$ (θ is a *simplification*),
  - and $body_{\theta(\mathcal{Q}')} \subseteq body_{\rho(\mathcal{Q})}$.

Example 5.2. We consider the queries $\mathcal{Q}$

$$H(w) \leftarrow R(u', u), R(u, v), R(v, w), R(u, w)$$

and $\mathcal{Q}'$

$$H'(y) \leftarrow R(x', x), R(x, x), R(x, y), R(y, z).$$

Then $\mathcal{Q}$ weakly covers $\mathcal{Q}'$. Indeed, the substitution ρ can map u, u' to x, v to y and w to z, and θ can map x' to x and leave the other variables alone. Since $head_{\theta(\mathcal{Q}')} = head_{\mathcal{Q}'}$, and $body_{\theta(\mathcal{Q}')} = \{R(x,x), R(x,y), R(y,z)\} \subseteq body_{\mathcal{Q}'}$, θ is indeed a simplification and, furthermore,
$body_{\theta(\mathcal{Q}')} \subseteq \{R(x,x), R(x,y), R(y,z), R(x,z)\} = body_{\rho(\mathcal{Q})}$.
However, we show that parallel-correctness does *not* transfer from $\mathcal{Q}$ to $\mathcal{Q}'$. Towards a contradiction, we assume that parallel-correctness *does* transfer. We consider the valuation V' mapping the variables x, x', z to a and y to b for some $a \neq b$. Then, $V'(body_{\mathcal{Q}'}) = \{R(a,a), R(a,b), R(b,a)\}$, V' derives $H'(b)$, and V' is minimal. Thanks to Proposition 4.4, there must be a minimal valuation V for $\mathcal{Q}$ such that

$$\{R(a,a), R(a,b), R(b,a)\} \subseteq V(body_{\mathcal{Q}}) \qquad (\dagger)$$

holds. We distinguish two cases.

Case 1 $V(w) = a$. Thus V derives $H(a)$. However, the valuation that maps all variables to a also derives $H(a)$ and shows that V is not minimal.

Case 2 $V(w) = b$. Thus V derives $H(b)$. However, the valuation that maps all other variables to a also derives $H(b)$ and only requires the facts $R(a,a)$ and $R(a,b)$. Therefore, V is not minimal.

Hence, no minimal valuation V for $\mathcal{Q}$ exists satisfying (†), the desired contradiction. Therefore, $\mathcal{Q} \not\xrightarrow{\text{pc}} \mathcal{Q}'$.

On the other hand, it follows from Proposition 5.4 that parallel-correctness *weakly* transfers from $\mathcal{Q}$ to $\mathcal{Q}'$. □

We note that, as the naming already suggests, if parallel-correctness transfers from $\mathcal{Q}$ to $\mathcal{Q}'$ it also weakly transfers, and if $\mathcal{Q}$ covers $\mathcal{Q}'$ it also weakly covers it.

Although the definition of "weakly covers" seems quite involved, it can be tested in NP.

Proposition 5.3. *The decision problem, whether a given CQ $\mathcal{Q}$ weakly covers a given CQ $\mathcal{Q}'$ is* NP*-complete.*

[5]Recall that the notion of strong saturation is introduced in Definition 3.7.

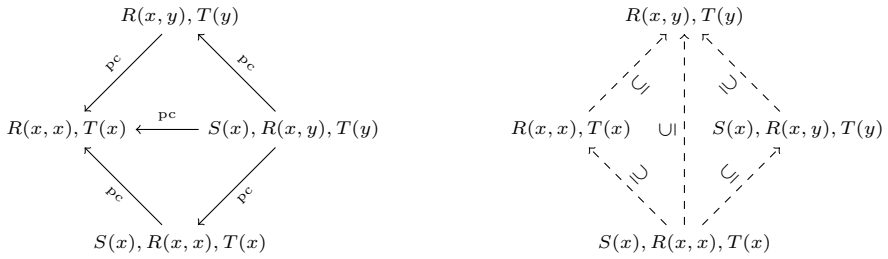

Figure 1: Relationship between the queries of Example 4.2 with respect to (a) parallel-correctness transfer and (b) query containment.

It turns out that "weak cover" characterizes "weak transfer" and thus the following counterpart of Proposition 4.4 holds:

Proposition 5.4. *For CQs $\mathcal{Q}$ and $\mathcal{Q}'$, parallel correctness weakly transfers from $\mathcal{Q}$ to $\mathcal{Q}'$ if and only if $\mathcal{Q}$ weakly covers $\mathcal{Q}'$.*

Thus, if in some setting, weak transferability and transferability coincide, Propositions 5.3 and 5.4 yield an NP upper bound for parallel-correctness transfer. We present two such settings in the following two subsections.

5.2 Strongly minimal queries

Definition 5.5. A CQ query is *strongly minimal* if all its valuations are minimal.

We write $\mathbf{CQ}[sm]$ for the class of strongly minimal CQs.

Although strong minimality is a non-trivial notion (with a coNP-complete decision problem), there are some very common examples like full queries (where all variables occur in the head) and queries without self-joins (where every relation name occurs at most once), and further kinds of queries like

$$H(x_1, x_2) \leftarrow R(x_1, x_3), R(x_2, x_3), S(x_3, x_2).$$

For strongly minimal conjunctive queries, transfer and weak transfer coincide and therefore the following holds.

Theorem 5.6. pc-trans($\boldsymbol{CQ}[sm], \boldsymbol{CQ}$) *is* NP*-complete.*

Also the complexity of parallel-correctness drops for strongly minimal queries, if the representation of the distribution policy allows to figure out in polynomial time whether there is a node that is responsible for a given set of facts. In particular, the following holds:

Theorem 5.7. *For policy class $\mathcal{P} \in \mathfrak{P}_{poly}$,* PCI($\boldsymbol{CQ}[sm], \mathcal{P}$) *and* PC($\boldsymbol{CQ}[sm], \mathcal{P}$) *are in* coNP.

5.3 Feasible Families of Distribution Policies

Parallel-correctness transfer can be seen as a generalization of parallel-correctness. In both cases, the goal is to decide whether a query can be correctly evaluated by evaluating it locally at each node. However, for parallel-correctness *transfer*, the question whether $\mathcal{Q}'$ is parallel-correct is not asked for a particular distribution policy but for the *family*[6] of those distribution policies for which $\mathcal{Q}$ is parallel-correct, which by Proposition 3.6 is just the family of all distribution policies that saturate $\mathcal{Q}$.

[6]A family of distribution policies is just a set of distribution policies.

Definition 5.8. A query is *parallel-correct for a family $\mathcal{F}$* of distribution policies if it is parallel-correct under every distribution policy from $\mathcal{F}$.

We next define a criterion for families of distribution policies which guarantees that parallel-correctness can be tested in NP.

For an instance I, a distribution policy $\boldsymbol{P}$ is called $(\mathcal{Q}, I)$-*scattered* if, for each node κ, there is a valuation V for $\mathcal{Q}$ such that $\textit{loc-inst}_{\boldsymbol{P},I}(\kappa) \subseteq V(\textit{body}_{\mathcal{Q}})$. Intuitively, a $(\mathcal{Q}, I)$-scattered policy ensures that facts are sufficiently spread out such that every network node only contains a subset of the facts related to one valuation. We call a family $\mathcal{F}$ of distribution policies $\mathcal{Q}$-*scattered* if $\mathcal{F}$ contains a $(\mathcal{Q}, I)$-scattered policy, for every instance I. A $\mathcal{Q}$-scattered family of distribution policies that strongly saturate $\mathcal{Q}$ is called a $\mathcal{Q}$-*family*.

It turns out that parallel-correctness with respect to $\mathcal{Q}$-families can be characterized in terms of weak coverability,

Proposition 5.9. *Let $\mathcal{Q}$ be a CQ and let $\mathcal{F}$ be a $\mathcal{Q}$-family. Then, for every CQ $\mathcal{Q}'$, query $\mathcal{Q}'$ is parallel correct for $\mathcal{F}$ if and only if $\mathcal{Q}$ weakly covers $\mathcal{Q}'$.*

And thus, Proposition 5.3 yields another NP-result. Indeed, the following can be shown:

Theorem 5.10. *It is* NP*-complete to decide, for given CQs $\mathcal{Q}$ and $\mathcal{Q}'$, whether $\mathcal{Q}'$ is parallel-correct for $\mathcal{Q}$-families of distribution policies.*

For a CQ $\mathcal{Q}$, let $\mathcal{H}_{\mathcal{Q}}$ denote the family

$$\{\boldsymbol{P}_H \mid H \text{ is a hypercube for } \mathcal{Q}\}$$

of distribution policies. Then the next lemma specifies that Hypercube distributions for a CQ $\mathcal{Q}$ form a $\mathcal{Q}$-family.

Lemma 5.11. *Let $\mathcal{Q}$ be a CQ. Then $\mathcal{H}_{\mathcal{Q}}$ is a $\mathcal{Q}$-family.*

As a corollary, we now have:

Corollary 5.12. *It is* NP*-complete to decide, for given conjunctive queries $\mathcal{Q}, \mathcal{Q}'$, whether $\mathcal{Q}'$ is parallel-correct for $\mathcal{H}_{\mathcal{Q}}$.*

6. DISCUSSION

Parallel-correctness serves as a framework for studying correctness and implications of data partitioning in the context of one-round query evaluation algorithms. A main insight of the work up to now is that testing for parallel-correctness as well as the related problem of parallel-correctness transfer reduces to reasoning about minimal valuations in the context of conjunctive queries (even in the presence of union and inequalities) but becomes considerably more involved when negation is allowed.

There are many questions left unexplored and we therefore highlight possible directions for further research.

From a foundational perspective, it would be interesting to explore the decidability boundary for parallel-correctness and transfer when considering more expressive query languages or even other data models. Obviously, the problems become undecidable when considering first-order logic, but one could consider monotone languages or for instance guarded fragment queries. At the same time, it would be interesting to find settings that render the problems tractable, for instance, by restricting the class of queries or by limiting to certain classes of distribution policies.

Parallel-correctness transfer is a rather strong notion as it requires that a query $\mathcal{Q}'$ is parallel-correct for *every* distribution policy for which another query $\mathcal{Q}$ is parallel-correct. As a consequence, query $\mathcal{Q}'$ can always be executed after $\mathcal{Q}$ without reshuffling of the data. From a practical perspective, however, it could be interesting to determine, given $\mathcal{Q}$ and $\mathcal{Q}'$, whether there is at least one distribution policy under which both queries are correct. Other questions concern the least costly way to migrate from one distribution to another. As an example, assume a distribution $\boldsymbol{P}$ on which $\mathcal{Q}$ is parallel-correct but $\mathcal{Q}'$ is not. Find a distribution $\boldsymbol{P}'$ under which $\mathcal{Q}'$ is parallel-correct and that minimizes the cost to migrate from $\boldsymbol{P}$ to $\boldsymbol{P}'$. Similar questions can be considered for a workload of queries.

Even though the naive one-round evaluation model considered in this paper suffices for Hypercube, it is rather restrictive. Other possibilities are to consider more complex aggregator functions than union and to allow for a different query than the original one to be executed at computing nodes. Furthermore, it could be interesting to generalize the framework beyond one-round algorithms, that is, towards evaluation algorithms that comprise of several rounds.

Acknowledgments

We thank Serge Abiteboul, Luc Segoufin, Cristina Sirangelo, Dan Suciu, and Thomas Zeume for helpful remarks and Jan-Eric Lenssen for careful proof reading.

7. REFERENCES

[1] F. N. Afrati and J. D. Ullman. Optimizing joins in a map-reduce environment. In *EDBT 2010, 13th International Conference on Extending Database Technology*, pages 99–110, 2010.

[2] T. J. Ameloot, G. Geck, B. Ketsman, F. Neven, and T. Schwentick. Parallel-correctness and transferability for conjunctive queries. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015*, pages 47–58, 2015.

[3] T. J. Ameloot, G. Geck, B. Ketsman, F. Neven, and T. Schwentick. Parallel-correctness and transferability for conjunctive queries. Invited Journal Submission, 2015.

[4] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd Symposium on Principles of Database Systems*, PODS '13, pages 273–284, 2013.

[5] P. Beame, P. Koutris, and D. Suciu. Skew in parallel query processing. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '14, pages 212–223, 2014.

[6] S. Chu, M. Balazinska, and D. Suciu. From theory to practice: Efficient join query evaluation in a parallel database system. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 63–78, 2015.

[7] S. Ganguly, A. Silberschatz, and S. Tsur. Parallel bottom-up processing of datalog queries. *J. Log. Program.*, 14(1&2):101–126, 1992.

[8] G. Geck, B. Ketsman, F. Neven, and T. Schwentick. Parallel-correctness and containment for conjunctive queries with union and negation. In *International Conference on Database Theory*, pages 9:1–9:17, 2016.

[9] D. Halperin, V. Teixeira de Almeida, L. L. Choo, S. Chu, P. Koutris, D. Moritz, J. Ortiz, V. Ruamviboonsuk, J. Wang, A. Whitaker, S. Xu, M. Balazinska, B. Howe, and D. Suciu. Demonstration of the myria big data management service. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 881–884, 2014.

[10] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis. Dremel: Interactive analysis of web-scale datasets. *Proc. VLDB Endow.*, 3(1-2):330–339, Sept. 2010.

[11] M. Mugnier, G. Simonet, and M. Thomazo. On the complexity of entailment in existential conjunctive first-order logic with atomic negation. *Inf. Comput.*, 215:8–31, 2012.

[12] R. Nehme and N. Bruno. Automated partitioning design in parallel database systems. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 1137–1148, 2011.

[13] H. Q. Ngo, E. Porat, C. Ré, and A. Rudra. Worst-case optimal join algorithms. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012*, pages 37–48, 2012.

[14] M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems, Third Edition*. Springer, 2011.

[15] J. Rao, C. Zhang, N. Megiddo, and G. Lohman. Automating physical database design in a parallel database. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, SIGMOD '02, pages 558–569, 2002.

[16] J. Shute, R. Vingralek, B. Samwel, B. Handy, C. Whipkey, E. Rollins, M. Oancea, K. Littlefield, D. Menestrina, S. Ellner, J. Cieslewicz, I. Rae, T. Stancescu, and H. Apte. F1: A distributed sql database that scales. *Proc. VLDB Endow.*, 6(11):1068–1079, Aug. 2013.

[17] J. D. Ullman. Information integration using logical views. *Theor. Comput. Sci.*, 239(2):189–210, 2000.

[18] T. L. Veldhuizen. Triejoin: A simple, worst-case optimal join algorithm. In *Proc. 17th International Conference on Database Theory (ICDT)*, pages 96–106, 2014.

[19] R. S. Xin, J. Rosen, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica. Shark: Sql and rich analytics at scale. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 13–24, 2013.