A Worst-Case Optimal Multi-Round Algorithm for Parallel Computation of Conjunctive Queries

Bas Ketsman* Dan Suciu†

Abstract

We study the optimal communication cost for computing a full conjunctive query $\mathcal{Q}$ over p distributed servers. Two prior results were known. First, for one-round algorithms over skew-free data the optimal communication cost per server is $m/p^{1/\tau^*(\mathcal{Q})}$, where m is the size of the largest input relation, and τ^* is the fractional vertex covering number of the query hypergraph. Second, for multi-round algorithms and unrestricted database instances, it was shown that any algorithm requires at least $m/p^{1/\rho^*(\mathcal{Q})}$ communication cost per server, where $\rho^*(\mathcal{Q})$ is the fractional edge covering number of the query hypergraph; but no matching algorithms were known for this case (except for two restricted queries: chains and cycles).

In this paper we describe a multi-round algorithm that computes any query with load $m/p^{1/\rho^*(\mathcal{Q})}$ per server, in the case when all input relations are binary. Thus, we prove this to be the optimal load for all queries over binary input relations. Our algorithm represents a non-trivial extension of previous algorithms for chains and cycles, and exploits some unique properties of graphs, which no longer hold for hyper-graphs.

1 Introduction

In recent years, a new family of query evaluation algorithms has emerged with proven optimal cost. Traditional query engines choose a join order, then compute the joins one at a time, and, as a consequence, may compute intermediate results that are much larger than the final query output. This leads to suboptimal evaluation algorithms, even if the best join order is chosen by the optimizer. In contrast, the new class of algorithms achieve optimality by computing all joins at once, and thus avoiding to compute potentially large intermediate results. For complex queries on very large datasets, the new class of algorithms can be significantly more efficient than traditional query plans.

In the case of sequential algorithms, running on a single server, the complexity is given by the runtime. The optimal runtime for a full conjunctive query $\mathcal{Q}$ on a database instance where the largest relation has m tuples is given by the largest possible query output. Atserias, Grohe, and Marx [4] have shown that the query output has size $\leq m^{\rho^*(\mathcal{Q})}$, where $\rho^*(\mathcal{Q})$ is the optimal edge covering number of the query hypergraph and m is the size of the largest relation; this bound is referred to as the AGM bound. Several algorithms have been described in the literature that run within this bound: Leapfrog-trie-join [16], NPRR [13], and Generic Join [14].

In this paper we study distributed algorithms, which run on a cluster of p servers, usually called a *shared nothing* architecture. A formal model for a shared nothing architecture was described in [12] and is called the Massively Parallel Communication model, hence MPC. In this model the servers compute a query in several rounds, where each round consists of a local computation followed by a communication step where the entire data is reshuffled among the p servers. The complexity of an algorithm in the MPC model is given by the *communication cost*, defined as the largest amount of data received by any single server during any round of communication, and also called *load per server*. An ideal load is $\tilde{O}(m/p)$, with m the size of the largest relation, but often queries require a larger load.[1] Unlike the sequential setting, the optimal load for computing full conjunctive queries in the MPC model is open. However, two important partial results have been described recently.

First, Koutris et al. [6] consider the MPC model under two restrictions: there is only one round of communication, and the database instances are skew free. A simplified definition of a skew-free database is one where every data value occurs at most m/p times in any input relation. Under these restrictions, the authors showed that the optimal load is $O(m/p^{1/\tau^*(\mathcal{Q})})$, where $\tau^*(\mathcal{Q})$ is the optimal fractional vertex covering number of the query hypergraph.

A second result was shown by the same authors in [11]. They dropped both restrictions, and proved the following lower bound: any algorithm, with r number of rounds, must incur a communication cost that is at least $m/(r \cdot p^{1/\rho^*(\mathcal{Q})})$. They also gave matching algorithms, but only for cycles and for chain queries: no matching algorithm is known in general. The proof of the lower bound relies on the AGM bound, and can be described informally as fol-

*PhD Fellow of the Research Foundation - Flanders (FWO)
†Supported by NSF IIS-1247469 and NSF AiTF 1535565

[1] In this paper, $\tilde{O}$ hides a factor $\log^r(p)$, with r the largest arity of any relation.

lows. Consider a worst-case database instance, on which the query's output is the AGM bound $m^{\rho^*(\mathcal{Q})}$. Assume an algorithm that computes the output of this query using a load L. Then, any server receives at most $r \cdot L$ tuples from each input relation, and thus it can output at most $(r \cdot L)^{\rho^*(\mathcal{Q})}$ tuples, by the AGM bound. Since the p servers must return all output tuples, $p \cdot (r \cdot L)^{\rho^*(\mathcal{Q})} \geq m^{\rho^*(\mathcal{Q})}$, and therefore the load is $\geq m/(r \cdot p^{1/\rho^*(\mathcal{Q})})$. We assume $O(1)$ rounds throughout this paper, and then the lower bound becomes $\Omega(m/p^{1/\rho^*(\mathcal{Q})})$.

These two lower bounds work under quite different assumptions. The first lower bound of $m/p^{1/\tau^*(\mathcal{Q})}$ works for one round of communication and holds even for databases where each value occurs at most once in any input relation (in particular, they are skew-free). Therefore, the query's output is no larger than m, the size of the largest input relation. The second lower bound, $m/p^{1/\rho^*(\mathcal{Q})}$, removes any restrictions on the number of rounds, and applies to databases where the query returns the largest output, in other words where the AGM bound is tight. The fractional vertex cover $\tau^*(\mathcal{Q})$ seems related to skew-free databases, where the output is small, while the fractional edge cover $\rho^*(\mathcal{Q})$ seems related to skewed databases where the output is large. Since no matching algorithm is known for the general case, it is open which of these two bounds, if any, will turn out to be optimal.

In this paper we prove that any full conjunctive query where the input relations are binary can be computed with load $\tilde{O}(m/p^{1/\rho^*(\mathcal{Q})})$. Since this matches the general lower bound, our algorithm is optimal, and proves that the optimal load is given by $\rho^*(\mathcal{Q})$ rather than $\tau^*(\mathcal{Q})$, when all input relations are binary.

Our techniques build upon, and extend in non-trivial ways, techniques introduced by Koutris et al. [11] for computing chain queries and cycle queries in multiple rounds with load $m/p^{1/\rho^*(\mathcal{Q})}$. Their main insight was a simple algorithm for computing the two-way semi-join query,

$$\mathrm{H}(x, y) \text{ :- } \mathrm{R}(x), \mathrm{S}(x, y), \mathrm{T}(y),$$

in two rounds, with load m/p. If we restrict the number of rounds to one, then any algorithm requires a load $\geq m/p^{1/2}$, because $\tau^*(\mathcal{Q}) = 2$. But in two rounds we can simply compute $\mathrm{R}(x), \mathrm{S}(x, y)$ first, with load m/p by hash-partitioning on x, then compute the second semi-join by hash-partitioning on y and using the fact that the intermediate result is no larger than the size of S. Thus, the two-way semi-join can be computed with load $m/p^{1/\rho^*(\mathcal{Q})}$, since $\rho^*(\mathcal{Q}) = 1$. The algorithm for chains and cycles in [11] consists of repeatedly separating the values in the database into light and heavy hitters, and handling the heavy hitters in multiple rounds, through semi-join reductions. The specific techniques however apply only to chains and cycles (and can be generalized to trees). The paper left open the optimal communication cost for queries beyond chains and cycles.

Our result represents a non-trivial generalization of the algorithms for chains and cycles. We use the same building blocks as in [11], namely splitting the input values into light and heavy hitters, then computing the query separately on the two sets, but we require several new ideas to compute arbitrary queries over binary relations. We use three key properties of graphs (which, in general, do not hold on arbitrary hypergraphs): $\rho^*(\mathcal{Q}) + \tau^*(\mathcal{Q})$ equals the number of nodes; $\tau^*(\mathcal{Q}) \leq \rho^*(\mathcal{Q})$; and the graph admits a half-integral optimal fractional edge packing. Our algorithm starts by computing an optimal half-integral edge packing f for the query's graph. Then the exact strategy depends on the structure of f. The easy case is when f is tight, meaning that f describes both an edge packing and edge cover. The general case is when f is non-tight, and here we identify fragments of the query having a tight edge packing and compute them as before.

Thus, our results show that, at least in the case of queries over binary relations, the communication cost for multi-round algorithms is given by $\rho^*(\mathcal{Q})$ rather than $\tau^*(\mathcal{Q})$. We leave open the optimal communication cost for queries over relations of arbitrary arity. In particular, it is not known what this bound might be for queries where $\tau^*(\mathcal{Q}) > \rho^*(\mathcal{Q})$. In general, the best algorithm we know to compute the query over skew-free database instances is the one-round algorithm with load $m/p^{1/\tau^*(\mathcal{Q})}$. (The two-way semi-join query is an exception where $\tau^*(\mathcal{Q}) > \rho^*(\mathcal{Q})$ and we happen to know how to compute it in two rounds with load given by $\rho^*(\mathcal{Q})$.) On the other hand, the best multi-round lower bound we know follows from the AGM bound and is $m/p^{1/\rho^*(\mathcal{Q})}$: to close the gap one needs to either design a new multi-round algorithm for skew-free databases, or prove a new multi-round bound that is stronger than that implied by the AGM bound.

Outline We introduce the necessary definitions in Section 2. Section 3 describes the MPC model. In Section 4 we introduce the basic building blocks which are used in Section 5 to describe our main result, a worst-case optimal multi-round algorithm for parallel-evaluation of conjunctive queries. In Section 6 we briefly relate to the external memory model. We conclude in Section 7.

2 Definitions

Instances and Queries For a set S and positive integer n we denote by $|S|$ the number of elements in S and by $[n]$ the set $\{1, \ldots, n\}$. Let **dom** be an infinite domain of data values. A *schema* σ is a finite set of relation symbols R with associated arities $ar(R)$. For relation symbol R and tuple $\boldsymbol{t} = (d_1, \ldots, d_n)$ over **dom**, we call $R(\boldsymbol{t})$ a fact if $n = ar(R)$. A fact $R(\boldsymbol{t})$ is said to be over database schema σ if $R \in \sigma$. An *instance* I over σ is a finite set of facts over σ. For relation name R we denote by R^I the tuples in instance I with relation symbol R, that is $R^I = \{\boldsymbol{t} \mid R(\boldsymbol{t}) \in I\}$. A *query* over input schema σ_1 and output schema σ_2 is a generic mapping from σ_1-instances

to σ_2-instances.

Conjunctive Queries We assume an infinite set of variables **var** disjoint from **dom**. An *atom* over schema σ is of the form $R(\boldsymbol{x})$, where R is a relation symbol in σ and $\boldsymbol{x}$ is a tuple $(x_1, \ldots, x_n)$ of variables from **var** with $n = ar(R)$. A *conjunctive query* (CQ) $\mathcal{Q}$ over input schema σ is a query of the form

$$\mathrm{H}(\boldsymbol{x_0}) \text{ :- } \mathrm{S}_1(\boldsymbol{x_1}), \mathrm{S}_2(\boldsymbol{x_2}), \ldots, \mathrm{S}_\ell(\boldsymbol{x_\ell}),$$

where $\mathrm{S}_i(\boldsymbol{x_i})$ are atoms over σ and $\mathrm{H}(\boldsymbol{x_0})$ is an atom with $H \notin \sigma$. Atom $\mathrm{H}(\boldsymbol{x_0})$ is called the *head* of $\mathcal{Q}$, and the set $\{\mathrm{S}_i(\boldsymbol{x_i}) \mid i \in [\ell]\}$ its *body*. We assume that every variable in the head of a query also occurs in its body. For conjunctive query $\mathcal{Q}$ and subset C of body atoms, we sometimes write $\mathcal{Q}_C$ to denote the query obtained by removing from $\mathcal{Q}$ all atoms not in C.

By $vars(\mathcal{Q})$ we denote the variables of query $\mathcal{Q}$ and by $atoms(\mathcal{Q})$ its body atoms. If $vars(\mathcal{Q}) = \boldsymbol{x_0}$ we call $\mathcal{Q}$ *full*. If $\mathrm{S}_i \neq \mathrm{S}_j$ for all $i \neq j$, with $i, j \in [\ell]$, we say that $\mathcal{Q}$ is *without self-joins*. In the remainder of this document we only consider CQs that are full and without self-joins. For simplicity, we also assume each variable to occur at most once in every atom. We call a variable *isolated* if it occurs only in unary atoms.

The semantics of conjunctive queries is defined in terms of valuations. A *valuation* V for query $\mathcal{Q}$ is a mapping from the variables in $\mathcal{Q}$ to values in **dom**. A valuation V for $\mathcal{Q}$ *satisfies* on instance I if all the facts obtained by applying V over the body atoms of $\mathcal{Q}$ are in I. The output $\mathcal{Q}(I)$ of a conjunctive query $\mathcal{Q}$ over instance I is the set of facts obtained by applying satisfying valuations for $\mathcal{Q}$ on I over its head atom.

For query $\mathcal{Q}$, we define the degree of variable $x \in vars(\mathcal{Q})$ as the number of atoms containing x, and the *degree* of $\mathcal{Q}$, denoted $d(\mathcal{Q})$, as the maximum degree of any variable in $\mathcal{Q}$. For set of variables $X \subseteq vars(\mathcal{Q})$, the *residual query* $\mathcal{Q}[X]$ is the query obtained by removing variables X from $\mathcal{Q}$, and decreasing accordingly the arities of the relations that contain those variables.

Example 1. *For an example, consider the chain query,*

$$\mathrm{L2}(x_1, x_2, x_3, x_4) \text{ :- } \mathrm{S}_1(x_1, x_2), \mathrm{S}_2(x_2, x_3), \mathrm{S}_3(x_3, x_4).$$

The residual query $L2[\{x_2, x_3\}]$ *takes the form:*

$$\overline{\mathrm{L2}}(x_1, x_4) \text{ :- } \overline{\mathrm{S}}_1(x_1), \overline{\mathrm{S}}_2(), \overline{\mathrm{S}}_3(x_4).$$

Henceforth, we omit mentioning schemas for queries and instances when they are clear from the context.

Fractional Covers and Packings We recall the definition of fractional edge (vertex) packing and fractional vertex (edge) cover of a query's hypergraph. Let f be a mapping from the atoms in $\mathcal{Q}$ to positive weights; f is a *fractional edge packing* if

$$\sum_{a : x \in vars(a)} f(a) \leq 1, \text{ for every } x \in vars(\mathcal{Q}), \qquad (1)$$

where a is over $atoms(\mathcal{Q})$. If instead

$$\sum_{a : x \in vars(a)} f(a) \geq 1, \text{ for every } x \in vars(\mathcal{Q}), \qquad (2)$$

where a is again over $atoms(\mathcal{Q})$, then f is called a *fractional edge cover* for $\mathcal{Q}$. In both cases we refer to the sum $\sum_{a \in atoms(\mathcal{Q})} f(a)$ as the *value* of f for $\mathcal{Q}$, and are mostly interested in mappings f with optimal value. Optimal means maximum for fractional edge packings, and minimum for fractional edge covers.

Both LP-problems have a natural dual over the variables of $\mathcal{Q}$: A mapping f' from the variables in $\mathcal{Q}$ to positive weights is a *fractional vertex cover* if

$$\sum_{x \in vars(a)} f'(x) \geq 1, \text{ for every } a \in atoms(\mathcal{Q}), \qquad (3)$$

and describes a *fractional vertex packing* if instead

$$\sum_{x \in vars(a)} f'(x) \leq 1, \text{ for every } a \in atoms(\mathcal{Q}). \qquad (4)$$

Similar as before, for mappings f' we refer to the result of $\sum_{x \in vars(\mathcal{Q})} f'(x)$ as their *value* for $\mathcal{Q}$ and we are interested in finding fractional vertex covers with minimum value, and fractional vertex packings with maximum value. At optimality, the value of the fractional edge packing coincides with the value of the fractional vertex cover, and is called the *fractional vertex covering number* for $\mathcal{Q}$, denoted $\tau^*(\mathcal{Q})$. Similarly, at optimality the value of fractional edge cover coincides with the value of fractional vertex packing, and is called the *fractional edge covering number*, denoted $\rho^*(\mathcal{Q})$.

In general there is no clear relation between $\tau^*(\mathcal{Q})$ and $\rho^*(\mathcal{Q})$, unless f or f' are tight. We call f tight if the inequalities in conditions (1) and (2) are equalities. Similar for f' and conditions (3) and (4). A tight fractional edge packing is also a fractional edge cover, thus implying $\tau^*(\mathcal{Q}) \geq \rho^*(\mathcal{Q})$. Similarly, a tight fractional vertex cover is also a fractional vertex packing, thus $\tau^*(\mathcal{Q}) \leq \rho^*(\mathcal{Q})$.

Degree Information Let I be an input instance and S a relation symbol. For a value $c \in$ **dom**, and column i of relation S, $i \in [ar(S)]$, denote by $\deg_S(c, i)$ the frequency of value c in column i of relation S^I. When S is unary, $\deg_S(c, 1) = 1$ if $S(c) \in S^I$ and $\deg_S(c, 1) = 0$ if $S(c) \notin S^I$.

Given a query $\mathcal{Q}$, we are usually interested in the frequencies of values for variables, rather than relation columns. Therefore we introduce the following abbreviations: For this, consider value $c \in$ **dom**, variable $x \in vars(\mathcal{Q})$, and let a be a body atom of $\mathcal{Q}$ where x

is incident to. By $\deg_S(c, x)$ we denote the frequency of value c in in the column of S^I corresponding to the position of x in atom a. For example, for atom $S(x, y)$, $\deg_S(c, x) = \deg_S(c, 1)$ and $\deg_S(c, y) = \deg_S(c, 2)$. Further, by $\deg(c, x)$ we denote the maximal frequency over all relations where x is incident to in $\mathcal{Q}$. The latter means that $\deg(c, x) = \max_R\{\deg_R(c, x)\}$, where R is over those relations incident to variable x in $\mathcal{Q}$.

3 The Multi-Round MPC Model

We study query evaluation in the *Massively Parallel Communication* model (MPC) [12, 5], which takes a number p of servers and allows to express large-scale data processing algorithms in a sequence of rounds. Each round consists of two phases: a global communication phase in which data is reshuffled; and a local computation phase in which each server processes its local data fragment. Servers are assumed to be part of a complete communication network and have unlimited computation power. For a given algorithm, the maximum load is expressed in number of tuples and denotes the maximum amount of tuples that a server receives during any of the communication rounds. Initially, the input instance is assumed to be uniformly partitioned over the available servers, thus yielding load m/p. The query's answer is also distributed over the p servers, and consists of the union of all output fragments on all servers.

Worst-Case Load In this paper we consider the same framework as in [11] and are interested in algorithms for computing conjunctive queries with worst-case optimal load. The latter particularly means that we place no restrictions on the input instance, which thus may have skew. Skew refers to the maximum frequency of values occurring in certain columns of instance relations.

Allowing p rounds enables servers to collect the entire input instance and thus generate the desired output for a given query with load at most m/p in a trivial way. To prevent this undesirable behaviour from happening, we consider only algorithms that run in a constant number of rounds. As our focus is on large-scale data we consider data complexity and additionally assume that the size of the data is significantly larger than the number of servers: $m > p^2$.

Lower Bound The following lower-bound was obtained recently:

Lemma 1 ([11]). *For any randomized algorithm computing a CQ $\mathcal{Q}$ over p servers in $O(1)$ rounds, there exists an input instance where all relations have size $\leq m$ such that the load is $\Omega\left(m/p^{1/\rho^*(\mathcal{Q})}\right)$ w.h.p. in the random choices of the algorithm.*

Henceforth we say that a CQ $\mathcal{Q}$ has an *X-load algorithm* if an algorithm exists to compute $\mathcal{Q}$ in the above described model using p servers in a constant number of rounds with load at most $\tilde{O}\left(m/p^{1/X}\right)$ w.h.p., where m denotes the size of the largest relation, and X is a non-negative number, usually $\rho^*(\mathcal{Q})$.

For specific classes of CQs, like chain queries and cycle queries, a $\rho^*(\mathcal{Q})$-load algorithm is given in [11], thus showing that for these queries the lower bound is tight.

4 Building Blocks

In this section we set the stage by introducing the basic building blocks for our worst-case optimal multi-round algorithm, which is described in Section 5.

4.1 Hypercube

The hypercube algorithm was introduced by Afrati and Ullman [1] in the context of Mapreduce, where it is called *shares algorithm*, and has since then been intensively studied [5, 6, 3, 11]. For a conjunctive query $\mathcal{Q}$, the hypercube algorithm defines a reshuffling strategy based on hashing of data values allowing to compute $\mathcal{Q}$ through local evaluation on servers in parallel.

First, we assign to every variable x_i of $\mathcal{Q}$ a number p_i, which is called the *share* of x_i. Then, the p servers are conceptually ordered in a space $\mathcal{S} = [p_1] \times [p_2] \times \cdots \times [p_{|vars(\mathcal{Q})|}]$, where every coordinate in $\mathcal{S}$ points to a unique server. Here it is assumed that $\prod p_i \leq p$. Additionally, to every variable a randomly chosen hash function $h_i : \mathbf{dom} \to [p_i]$ is assigned which maps data values to a bucket in the associated share.

During its single communication round, the HC algorithm sends every fact $R(a_1, \ldots, a_n)$ over atom $R(x_{i_1}, \ldots, x_{i_n})$ to the set of servers agreeing on the partial coordinate defined by its values and associated hash functions. More concrete, it sends the fact to all servers with coordinates in the set

$$\{\boldsymbol{c} \in \mathcal{S} \mid \bigwedge_{j \in [n]} c_{i_j} = h_{i_j}(a_j)\}.$$

It has been shown that the HC algorithm can compute CQs in the one round setting with optimal load when values are light, as made precise in the following lemma:

Lemma 2 ([6]). *Let $\mathcal{Q}$ be a CQ, I an instance, m be the maximal size over all relations in I, and f a fractional vertex cover for $\mathcal{Q}$. Define $p_i = p^{f(x_i)/(\sum_j f(x_j))}$ for every variable x_i. Suppose that for every atom $a_j = S_j(\boldsymbol{x_j})$ in $\mathcal{Q}$, every subset of variables $\boldsymbol{y} \subseteq \boldsymbol{x_j}$, and every tuple $\boldsymbol{t}$ over $\boldsymbol{y}$: $deg_{S_j}(\boldsymbol{t}, \boldsymbol{y}) \leq m/\Pi_{x_i \in \boldsymbol{y}} p_i$ (when this condition holds we say the database is skew-free). Then the HC algorithm with shares $\boldsymbol{p} = (p_1, \ldots, p_k)$ has a load $\tilde{O}(m/p^{1/\sum_i f(x_i)})$ with high probability over the choices of the random hash functions.*

Here, $\deg_{S_j}(\boldsymbol{t}, \boldsymbol{y})$ is defined as the frequency of tuple $\boldsymbol{t}$ over the columns of relation S_j that correspond to variables $\boldsymbol{y}$.

By choosing f the optimal fractional vertex cover, we obtain a one round, $\tau^*(\mathcal{Q})$-load algorithm, but in this paper we will also use the HC algorithm with different, non-optimal fractional vertex cover. When each relation has arity at most two, the skew-free condition simplifies to the following: for every variable x_i and value $v \in \mathbf{dom}$, $\deg(v, x_i) \leq m/p^{f(x_i)/\sum_j f(x_j)}$.

4.2 Semi-Join Decompositions

It is a common strategy in distributed query evaluation to reduce relations using semi-joins before sending them over the network. Interestingly, it has been observed in [11] that semi-join queries can be computed in the multi-round MPC setting with significantly less load compared to single-round algorithms, even when instances have no skew. That technique extends to any *guarded* query $\mathcal{Q}$, which is a conjunctive query that has some atom $b \in \mathit{atoms}(\mathcal{Q})$ for which $\mathit{vars}(b) = \mathit{vars}(\mathcal{Q})$. We call b a *guard atom* for $\mathcal{Q}$.

Lemma 3 ([11])**.** *Let $\mathcal{Q}$ be a guarded query, then $\mathcal{Q}$ can be computed using a constant number of rounds with load $\tilde{O}(m/p)$ over p servers.*

The number of rounds depends on the number of non-guard atoms. If the guard has arity at most two (which we assume through the paper), only two rounds are needed.

Example 2. *We illustrate the underlying algorithm by an example. For this, consider the generalized semi-join query* H *from the introduction,* $\mathrm{H}(x, y) \text{ :- } \mathrm{R}(x), \mathrm{S}(x, y), \mathrm{T}(y)$.

The algorithm takes only two rounds: the first round computes $\mathrm{S}'(x, y) \text{ :- } \mathrm{R}(x), \mathrm{S}(x, y)$*, and the second round computes* $\mathrm{H}(x, y) \text{ :- } \mathrm{S}'(x, y), \mathrm{T}(y)$*. We explain in detail the first round only: the second round is similar, since the size of* S' *is no larger than that of* S*. Call a value c in column x of $S(x, y)$ a light hitter if $deg_S(c, x) \leq m/p$: otherwise we call it a heavy hitter. The database has at most p heavy hitters and we may assume throughout the paper that all p servers know all heavy hitters c: this can be achieved in a preprocessing phase, using another round of communication. The algorithm treats separately the subsets of the relations* $\mathrm{R}(x)$ *and* $\mathrm{S}(x, y)$ *where x is a light hitter or a heavy hitter. For the light hitters, the algorithm uses a standard partitioned hash-join on the variable x: since all values are light, the maximum load per server is $\tilde{O}(m/p)$ w.h.p. For the heavy hitters, the algorithm uses a standard broadcast join: broadcast all heavy-hitter tuples in* $\mathrm{R}(x)$ *to all p servers, and* $\mathrm{S}(x, y)$ *is not reshuffled, then join the local fragment of* $\mathrm{S}(x, y)$ *with the entire relation* $\mathrm{R}(x)$*. Since there are $\leq p$ heavy hitters, the size of the broadcast relation* $\mathrm{R}(x)$ *is $\leq p$, hence the load per server is $\leq m/p$ because $p^2 \leq m$.*

Notice that the optimal load for computing H over instances without skew in the one-round MPC setting requires load $m/p^{1/2}$, by Lemma 2, which is significantly higher than load m/p.

Definition 1. Let $\mathcal{Q}$ be a query. A *semi-join decomposition for* $\mathcal{Q}$ is a minimal subset of atoms $\mathcal{V} \subseteq \mathit{atoms}(\mathcal{Q})$ such that every atom $a \notin \mathcal{V}$ is contained in some atom $b \in \mathcal{V}$, more precisely $\mathit{vars}(a) \subseteq \mathit{vars}(b)$. Given a semi-join decomposition $\mathcal{V}$ of $\mathcal{Q}$, the *reduced query* $\mathcal{Q}_v$ is defined as the query consisting of all atoms in $\mathcal{V}$.

If S is a relation symbol in $\mathcal{V}$ we sometimes denote S^v its occurrence in $\mathcal{Q}_v$. The *semi-join reduction* consists of $|\mathcal{V}|$ queries of the form

$$\mathrm{S}^v(\boldsymbol{x}) \text{ :- } \mathrm{S}(\boldsymbol{x}), \mathrm{R}_1(\boldsymbol{x}_1), \mathrm{R}_2(\boldsymbol{x}_2), \ldots, \mathrm{R}_k(\boldsymbol{x}_k),$$

where R_i are all atoms with $\boldsymbol{x}_i \subseteq \boldsymbol{x}$. It is straightforward to check that any algorithm for computing $\mathcal{Q}_v$ can be used to compute $\mathcal{Q}$ in two steps: (1) compute all semi-join reductions to obtain reduced relations S^v, and (2) compute $\mathcal{Q}_v$ on the reduced relations S^v.

A semi-join decomposition is not necessarily unique (e.g., when two atoms have $\mathit{vars}(a) = \mathit{vars}(b)$), but always exists. We can easily obtain one by iteratively removing atoms from $\mathcal{Q}$ that are variable-contained in other atoms. The remaining atoms form a semi-join decomposition $\mathcal{V}$ for $\mathcal{Q}$.

Example 3. *For an example consider query $\mathcal{Q}$,*

$$\mathrm{H}(x, y, z) \text{ :- } \mathrm{R}(x, y), \mathrm{S}(y, z), \mathrm{T}(y).$$

Then, the semi-join decomposition $\mathcal{V}$ for $\mathcal{Q}$ is the set $\{\mathrm{R}(x, y), \mathrm{S}(y, z)\}$*. The semi-join reduction consists of queries:*

$$\mathrm{R}^v(x, y) \text{ :- } \mathrm{R}(x, y), \mathrm{T}(y) \text{ and } \mathrm{S}^v(y, z) \text{ :- } \mathrm{S}(y, z), \mathrm{T}(y),$$

and the reduced query $\mathcal{Q}_v$ takes the form:

$$\mathrm{H}(x, y, z) \text{ :- } \mathrm{R}^v(x, y), \mathrm{S}^v(y, z).$$

We have the following generalization of Lemma 3:

Lemma 4. *Let $\mathcal{Q}$ be a CQ over relations with arity at most two and $\mathcal{V}$ a semi-join decomposition for $\mathcal{Q}$. All queries in the semi-join reduction can be computed in two rounds using p servers with load at most $\tilde{O}(m/p)$, where m is the maximal size of relations.*

4.3 Heavy-Hitter Configurations

Finally, we introduce the notion heavy-hitter configuration, which allows to partition instances based on where heavy hitters are located in the query structure. A similar technique is used in [11].

For an instance I, query $\mathcal{Q}$, and given threshold value $\delta \in [0, 1]$, we call a value c for variable x *heavy* (w.r.t. δ) if $\deg(c, x) > m/p^\delta$, with m the number of tuples in

the largest relation of I; and *light* otherwise. Notice that our definition of heavy and light is relation independent, therefore a variable x may have up to $k \cdot p^\delta$ many heavy hitters, with k the number of distinct atoms where x is incident to.

A *heavy-hitter configuration* Ψ for query $\mathcal{Q}$ is a pair (H, δ) with δ a threshold value and H a subset of the variables in $\mathcal{Q}$. For any atom $S(x_1, \ldots, x_k)$ in $\mathcal{Q}$, we call a fact $S(c_1, \ldots, c_k)$ (from instance I) compatible with Ψ if for all variables x_i and corresponding values c_i: $\deg(c_i, x_i) > m/p^\delta$ if $x_i \in H$, and $\deg(c_i, x_i) \leq m/p^\delta$ if $x_i \notin H$. By $I|_\Psi$ we denote the subset of instance I containing all facts compatible with configuration Ψ. We call I_Ψ the Ψ-*compatible* instance (w.r.t I). It is straightforward to check that $\bigcup_{H \subseteq vars(\mathcal{Q})} I|_{(H,\delta)} = I$, for every $\delta \in [0, 1]$.

A case of special interest is when $H = \emptyset$. Then we call a tuple in I_Ψ a *light hitter*, and call I_Ψ the *skew-free subinstance* of I, with respect to chosen threshold value δ. In particular, if $\delta = 0$ then $I_\Psi = I$, and if $\delta = 1$ then I_Ψ contains only values with degree $\leq m/p$.

Example 4. *For an example, let $p = 4$ and consider the join query $R(x, y), S(y, z)$ over instance I,*

$$I = \{\mathrm{R}(a, d), \mathrm{R}(b, d), \mathrm{R}(c, e), \mathrm{S}(d, a), \mathrm{S}(e, b), \mathrm{S}(c, d)\}.$$

Thus, $m = 3$. For threshold value $\delta = 1/2$, we have $m/p^\delta = 3/2$ and we distinguish the following heavy-hitter configurations:

$$I|_{(\{y\},\delta)} = \{\mathrm{R}(a, d), \mathrm{R}(b, d), \mathrm{S}(d, a)\},$$
$$I|_{(\emptyset,\delta)} = \{\mathrm{R}(c, e), \mathrm{S}(e, b), \mathrm{S}(c, d)\}.$$

For all other sets $H \subseteq vars(\mathcal{Q})$: $I|_{(H,\delta)} = \emptyset$.

Sometimes we want to express a subinstance in which certain values are fixed. For this, let $\mathcal{Q}$ be a query, I an instance, and $X \subseteq vars(\mathcal{Q})$. Let $\boldsymbol{t} = (c_1, \ldots, c_n)$ be a sequence of values from **dom**, denoting exactly one value for each variable in $X = \{x_1, \ldots, x_n\}$. Then, by $I_{\boldsymbol{t},X}$ we denote the subinstance of I consisting of all facts from I that are compatible with choice $\boldsymbol{t}$ for variables X in $\mathcal{Q}$. More specifically, $I|_{\boldsymbol{t},X}$ consists of all relations $R^{I|_{\boldsymbol{t},X}}$ defined as follows: if $R(y_1, \ldots, y_k)$ is an atom in the query, then $R^{I|_{\boldsymbol{t},X}} = \{R(d_1, \ldots, d_k) \in I \mid \forall y_i, x_j : y_i = x_j \Rightarrow d_i = c_j\}$. For example, recall instance I and query $\mathcal{Q}$ from Example 4, in which we want to express the subinstance of I compatible with value c on the position of variable x. Then $I_{c,x} = \{\mathrm{R}(c, e), \mathrm{S}(d, a), \mathrm{S}(e, b), \mathrm{S}(c, d)\}$. In other words, $I_{c,x}$ restricts the values of $R(x, y)$, but leaves the values of $S(y, z)$ unrestricted since this atom does not contain x.

A CQ $\mathcal{Q}$ can be computed over an instance I by fixing a threshold value and evaluating $\mathcal{Q}$ over each Ψ-compatible subinstance of I separately:

Lemma 5. *For CQ $\mathcal{Q}$, threshold value δ, and $X \subseteq vars(\mathcal{Q})$:*

1. $\mathcal{Q}(I) = \bigcup_{H \subseteq vars(\mathcal{Q})} \mathcal{Q}(I|_{H,\delta})$; *and*
2. $\mathcal{Q}(I) = \bigcup_{\boldsymbol{t} \in \boldsymbol{dom}^{|X|}} \mathcal{Q}(I_{\boldsymbol{t},X})$.

By the assumption that degree information is known for every value in the considered input instance, servers can recognize Ψ-compatible facts autonomously for specified configurations.

5 A Worst-Case Optimal Load Algorithm

In this section we present our main result:

Theorem 3. *Every CQ $\mathcal{Q}$ over relations with arity at most two can be computed with a $\rho^*(\mathcal{Q})$-load algorithm using at most seven rounds.*

We assume that all heavy hitters are known by all servers. If that is not the case, then one additional round of computation is required to compute and broadcast all heavy hitters. In a nutshell, our strategy is to first identify structural properties of the target query and then attack each heavy-hitter configuration in parallel using customized algorithms. More precise, we focus on a distraction-free subset of conjunctive queries, called simple queries, defined in Section 5.1. Here we also show that algorithms for simple queries can be generalized to the more general class of queries considered in Theorem 3. We then gradually provide algorithms for simple queries with desired load: for queries with tight edge packing first, in Section 5.3, then for queries without such packing in Section 5.4.

5.1 Simple and Connected Queries

We introduce the class of connected CQs first, then proceed to simple queries.

Connected Queries

A CQ is called *connected* when its query graph is connected, meaning that for every pair of variables $x, y \in vars(\mathcal{Q})$ there is a sequence of atoms $a_1, \ldots, a_n$ such that $x \in vars(a_1)$, $y \in vars(a_n)$, and for every $i \in [1, n-1]$: $vars(a_i) \cap vars(a_{i+1}) \neq \emptyset$. A subset C of atoms in CQ $\mathcal{Q}$ is called a *connected-component* of $\mathcal{Q}$ if $\mathcal{Q}_C$ is connected and C is maximal with this property.

By definition, the edge covering number is additive, meaning that the sum of fractional edge covering numbers over the connected components of a query equals its fractional edge covering number. We use this observation in the following lemma:

Lemma 6. *Let $\mathcal{Q}$ be a CQ and A and B be disjoint subsets of atoms from $\mathcal{Q}$, with $atoms(\mathcal{Q}) = A \cup B$ and $vars(A) \cap vars(B) = \emptyset$. Then,*

1. $\rho^*(\mathcal{Q}_A) + \rho^*(\mathcal{Q}_B) = \rho^*(\mathcal{Q})$; *and*

2. *$\mathcal{Q}$ has a $\rho^*(\mathcal{Q})$-load algorithm if $\mathcal{Q}_A$ has a $\rho^*(\mathcal{Q}_A)$-load algorithm and $\mathcal{Q}_B$ has a $\rho^*(\mathcal{Q}_B)$-load algorithm.*

Simple Queries

We now introduce the class of simple CQs. A CQ $\mathcal{Q}$ is called *simple* if (i) $\mathcal{Q}$ has only binary atoms and (ii) there are no distinct atoms $a, b \in atoms(\mathcal{Q})$ for which $vars(a) \subseteq vars(b)$. Notice that simple queries denote exactly those CQs representing a simple graph without self-loops (i.e. edges $R(x, x)$). The following lemma shows that we can transform every query into a simple query by doing a semi-join decomposition, without affecting the fractional edge covering number.

Lemma 7. *Let $\mathcal{Q}$ be a connected CQ where each relation has arity at most 2, and let $\mathcal{V}$ be a semi-join decomposition for $\mathcal{Q}$ (see Def. 1). Then (1) the reduced query $\mathcal{Q}_v$ is simple, and (2) $\rho^*(\mathcal{Q}) = \rho^*(\mathcal{Q}_v)$.*

Proof. (1) Since the query is connected, every atom that has only one variable has that variable occurring in at least one atom with two variables and, thus, will be eliminated by the semi-join decomposition. For (2), we show that (i) every fractional edge cover for $\mathcal{Q}_v$ defines also a fractional edge cover for $\mathcal{Q}$ with same value, and that (ii) every fractional edge cover for $\mathcal{Q}$ can be transformed into a fractional edge cover for $\mathcal{Q}_v$ with same value. The equality then follows.

For (i), observe that, by construction, every atom b^v in $\mathcal{Q}_v$ corresponds to an atom b in $\mathcal{Q}$ which is a guard in $\mathcal{Q}_b$. Therefore, given fractional edge cover f for $\mathcal{Q}_v$, fractional edge cover f', where $f'(b) = f(b^v)$ if $\mathcal{Q}_b$ exists and $f'(b) = 0$ otherwise, is as desired.

For (ii), let f be a fractional edge cover for $\mathcal{Q}$. Notice that for every atom $a \in atoms(\mathcal{Q})$ there is an atom $b \in \mathcal{V}$. Let π be the function mapping atoms a onto the corresponding atoms b, particularly their annotated versions, as described.[2] We define π so that $\pi(\emptyset)$ denotes the set of all atoms not in the range of π. Then, for every atom $b \in atoms(\mathcal{Q}_v)$, let $f'(b) = \sum_{a \in \pi^{-1}(b)} f(a)$. It now follows by construction that f' satisfies the LP-constraints for $\mathcal{Q}_v$, and that f' has the same value for $\mathcal{Q}_v$ as f for $\mathcal{Q}$. □

Example 5. *For an example we recall query $\mathcal{Q}$ and semi-join decomposition $\mathcal{V}$ from Example 3. Here, the optimal edge cover f for $\mathcal{Q}_v$ is unique and assigns weight 1 to both its atoms. It is easy to observe that f describes also an optimal edge cover for $\mathcal{Q}$.*

By the above lemma, a similar result as for unconnected queries can be obtained for queries with a semi-join decomposition:

[2]Notice that π exists but is not necessarily unique, as multiple atoms b may exist. Then one can arbitrarily choose one of the available options.

Lemma 8. *Let $\mathcal{Q}$ be a connected CQ where all relations have arity at most 2, and let $\mathcal{V}$ be a semi-join decomposition for $\mathcal{Q}$. If $\mathcal{Q}_v$ has a $\rho^*(\mathcal{Q}_v)$-load algorithm in r rounds, then $\mathcal{Q}$ has a $\rho^*(\mathcal{Q})$-load algorithm in $r + 2$ rounds.*

Proof. Let p be the number of available servers and m the size of the largest considered relation.

First, compute the semi-join reductions for $\mathcal{V}$, in two rounds, following the strategy in Lemma 4, with load at most $\tilde{O}(m/p)$. Next, compute $\mathcal{Q}_v$ over the reduced relations using the algorithm assumed by the lemma. Notice that the size of the semi-join reduced relations is at most m, and $\rho^*(\mathcal{Q}') = \rho^*(\mathcal{Q})$, thus the load is as desired. □

5.2 Light Hitters

At a very high level, our strategy for computing $\mathcal{Q}(I)$ is to consider all configurations of heavy hitters, and compute the query separately for each such configuration. We start by discussing the case when all variables are light: here, we compute the query using a particular choice of fractional vertex cover in Lemma 2. To explain it, we review a couple of nice properties from graph theory.

Of particular interest in this context is the result that, for simple graphs, every optimal fractional edge packing has a half-integral equivalent. The latter means a fractional edge packing with optimal value, and over range $\{0, 1/2, 1\}$. As shown in [15], one can obtain such packing by searching for an optimal fractional edge packing that also maximizes the number of 0-weight edges. Interestingly, these packings have a very particular structure, in which 1/2-weight edges express variable-disjoint odd-length cycles. Henceforth we denote the set of atoms in $\mathcal{Q}$ with weight 1/2 under fractional edge packing f by $\mathcal{C}_f(\mathcal{Q})$. Translated to conjunctive queries, we now have the following property:

Lemma 9 ([15])**.** *Let $\mathcal{Q}$ be a simple CQ. Then $\mathcal{Q}$ has a half-integral fractional edge packing with value $\tau^*(\mathcal{Q})$, and set $\mathcal{C}_f(\mathcal{Q})$ contains only disjoint odd-length cycles.*

In the case of simple graphs, there exists a simple relationship between the fractional vertex covering number and fractional edge covering number, which does not hold in general on hypergraphs:

Lemma 10 ([15])**.** *For any simple CQ $\mathcal{Q}$,*

1. *$\tau^*(\mathcal{Q}) \leq |vars(\mathcal{Q})|/2 \leq \rho^*(\mathcal{Q})$; and*
2. *$\tau^*(\mathcal{Q}) + \rho^*(\mathcal{Q}) = |vars(\mathcal{Q})|$.*

The proof of Lemma 10.1 follows by considering the assignment $f(x) = 1/2$ to every variable x. Since f is both a fractional vertex packing and a fractional vertex cover ($\mathcal{Q}$ is simple, hence every edge $R(x, y)$ connects two distinct nodes x, y and thus is covered, $f(x) + f(y) = 1$), with value is $\sum_x f(x) = |vars(\mathcal{Q})|/2$, it implies $\tau^*(\mathcal{Q}) \leq |vars(\mathcal{Q})|/2 \leq \rho^*(\mathcal{Q})$. The proof of Lemma 10.2 follows by noting that f is a vertex packing iff f' is a vertex

cover, where $f'(x) = 1 - f(x)$, and furthermore $\sum_x f(x) + \sum_x f'(x) = |vars(\mathcal{Q})|$.

Our algorithm computes the query on the light hitters by running the HC algorithm using the vertex cover $f(x) = 1/2$ for every variable x, implying:

Lemma 11. *Let $\mathcal{Q}$ be a simple CQ. Consider the heavy-hitter configuration $\Psi = (\emptyset, 1/|vars(\mathcal{Q})|)$. Then $\mathcal{Q}$ can be computed by a $2/|vars(\mathcal{Q})|$-load algorithm in one round on the instance I_Ψ. We call the tuples in I_Ψ* light hitters.

The proof follows by applying the HC algorithm in Lemma 2, and noting that $1/(\sum_j f(x_j)) = 2/|vars(\mathcal{Q})|$, and that the share of each variable x_i is $p^{f(x_i)/\sum_j f(x_j)} = p^{1/|vars(\mathcal{Q})|}$. In particular, the lemma gives a $\rho^*(\mathcal{Q})$-load algorithm for the subinstance I_Ψ.

Thus, our strategy for computing $\mathcal{Q}(I)$ is to consider all possible heavy-hitter configurations $\Psi = (H, \delta)$, for some δ (to be determined) and for all choices of heavy hitter variables $H \subseteq vars(\mathcal{Q})$ and compute the query separately on I_Ψ.

5.3 Tight Edge Packings

We show how to compute a conjunctive query under the assumption that it has a tight half-integral edge packing f; in the next section we remove this restriction. As observed before (Sec. 2), existence of a tight f implies $\rho^*(\mathcal{Q}) \leq \tau^*(\mathcal{Q})$, thus for simple CQs $\mathcal{Q}$ we have $\tau^*(\mathcal{Q}) = \rho^*(\mathcal{Q}) = |vars(\mathcal{Q})|/2$, by Lemma 10. Throughout this section we denote $L = vars(\mathcal{Q}) \setminus H = vars(\mathcal{Q}[H])$. To start, we show how to handle a simple case, when $\mathcal{Q}[H]$ has no binary atoms.

Lemma 12. *Let $\mathcal{Q}$ be a simple CQ. Assume that $\mathcal{Q}$ admits a tight edge packing f, and let $\Psi = (H, 1/|vars(\mathcal{Q})|)$ be a heavy-hitter configuration, where $\mathcal{Q}[H]$ has no binary atoms. Then $\mathcal{Q}$ can be computed over the Ψ-compatible instance by a $\rho^*(\mathcal{Q})$-load algorithm in two rounds.*

Proof. Let I be the Ψ-compatible instance. (Recall that I is the subset of some instance J with maximal-relation size m.) If $|vars(\mathcal{Q})| \leq 2$ the result follows trivially, as then $\mathcal{Q}$ consists of either a single atom (nothing needs to be done), or two disjoint unary atoms (Lemma 6 applies). Henceforth we assume $|vars(\mathcal{Q})| > 2$, which implies $\rho^*(\mathcal{Q}) \geq 3/2$ (with $\rho^*(\mathcal{Q}) = 3/2$ when $\mathcal{Q}$ is the triangle query).

Let I be an arbitrary Ψ-compatible instance. For the computation of $\mathcal{Q}(I)$ we consider a different threshold value, namely $\delta' = 2/|vars(\mathcal{Q})|$ and split instance I in $I_{\Psi'}$ for each heavy-hitter configuration $\Psi' = (H', \delta')$. Notice that heavy-hitters under δ remain heavy under δ', particularly because $m/p^{1/|vars(\mathcal{Q})|} > m/p^{2/|vars(\mathcal{Q})|}$. We now compute $\mathcal{Q}$ for every subinstance $I_{\Psi'}$ over the p servers in parallel. For this let $L' = vars(\mathcal{Q}) \setminus H'$.

We first show the case $L' = \emptyset$. Then instance $I_{\Psi'}$ is very small. More precise $|I'_\Psi| \leq O(p^{4/vars(\mathcal{Q})})$. Thus, we can send all facts in $I_{\Psi'}$ to a single server and compute $\mathcal{Q}(I_{\Psi'})$ locally. The load is as desired by assumption $m > p^2$, $|vars(\mathcal{Q})| \geq 3$, and $\rho^*(\mathcal{Q}) = vars(\mathcal{Q})/2$.

If $|L'| \geq 1$. We observe $|L'| \leq |vars(\mathcal{Q})|/2$. Indeed, every variable in L' is incident to a set of relations with 1 as the sum of weights (due to tightness of the packing), and no variables in L' share a relation, thus $|L'| \leq \rho^*(\mathcal{Q}) = |vars(\mathcal{Q})|/2$. Further, by the assumption that $\mathcal{Q}[H]$ has no binary atoms, it follows that $\mathcal{Q}[H']$ has no binary atoms either.

Now, to compute $\mathcal{Q}$, let $\mathcal{Q}'$ be the query obtained by removing from $\mathcal{Q}$ all atoms not incident to a variable in L'. Then $\mathcal{Q}'$ consists of unconnected components all representing a star with variable from L' in its center.

For $\mathcal{Q}(I_{\Psi'})$ we hash-partition the tuples over relations in $\mathcal{Q}'$ as for computing $\mathcal{Q}'(I_{\Psi'})$ using the HC-algorithm over $p^{2|L|/|vars(\mathcal{Q})|}$ servers, by allocating weight 1 for all variables from L' and weight 0 to all others. The load is at most $m/p^{2/|vars(\mathcal{Q})|}$ by Lemma 2.

Tuples over relations not in $\mathcal{Q}'$ are broadcast to all servers. As all these tuples are heavy, this does not affect the load (the analysis is analogous to case $L' = \emptyset$). Then the final output is obtained by computing $\mathcal{Q}'$ on every server. □

By Lemma 12, in the remainder of the section we assume $\mathcal{Q}[H]$ has at least one binary atom, thus $|L| \geq 2$. As a warm-up towards the general case, where $\mathcal{Q}[H]$ may generate unary and binary atoms, we first show the case where heavy-hitter configurations do not generate isolated variables: in that case semi-join decomposition $\mathcal{Q}_v$ of $\mathcal{Q}[H]$ is also simple (because the only unary relation symbols that can remain in the semi-join decomposition contain isolated variables).

Lemma 13. *Let $\mathcal{Q}$ be a simple CQ. Assume that $\mathcal{Q}$ admits a tight edge packing f, and let $\Psi = (H, 1/|vars(\mathcal{Q})|)$ be a heavy-hitter configuration, such that $\mathcal{Q}[H]$ is without isolated variables. Then $\mathcal{Q}$ can be computed over the Ψ-compatible instance by a $\rho^*(\mathcal{Q})$-load algorithm using three rounds.*

Proof. Let p be the number of available servers and I the Ψ-compatible instance. We use item 2 of Lemma 5 and write:

$$\mathcal{Q}(I) = \bigcup_{\boldsymbol{h}} \mathcal{Q}(I_{\boldsymbol{h},H}) \tag{5}$$

where $\boldsymbol{h}$ ranges over all heavy-hitter tuples. The algorithm proceeds by allocating for every heavy-hitter tuple a group of $p' = c \cdot p^{|L|/|vars(\mathcal{Q})|}$ exclusive servers, where $c = 1/d(\mathcal{Q})^{|H|}$ is a constant (recall that $d(\mathcal{Q})$ denotes the degree of the query graph). Then for every heavy-hitter tuple $\boldsymbol{h}$, $\mathcal{Q}(I_{\boldsymbol{h},H})$ is computed over its assigned set of servers.

Before giving the details, notice that the number of heavy-hitter tuples is at most $p^{|H|/|vars(\mathcal{Q})|}/c$ and $|L| +$

$|H| = |vars(\mathcal{Q})|$, thus we use a total number of servers $\leq p' \cdot p^{|H|/|vars(\mathcal{Q})|}/c = p$.

Next, we show how to compute $\mathcal{Q}(I_{\boldsymbol{h},H})$ over its p' servers. Consider an atom $R(x,y)$ that has a variable $x \in H$. If also $y \in H$, then, for the purpose of computing $\mathcal{Q}(I_{\boldsymbol{h},H})$, $R(x,y)$ is a single Boolean value (true or false), which we broadcast to all servers. Assume $y \in L$. Then R becomes a unary relation $R(y)$. To compute $\mathcal{Q}(I_{\boldsymbol{h},H})$ it thus suffices to compute $\mathcal{Q}[H]$, where the unary relations depend on the particular heavy-hitter tuple $\boldsymbol{h}$. To obtain the final output, output facts for $\mathcal{Q}[H]$ are augmented with values $\boldsymbol{h}$ and are in the output for $\mathcal{Q}$ only if the required booleans are present.

To show the computation of $\mathcal{Q}[H]$, let $\mathcal{V}$ be a semi-join decomposition for $\mathcal{Q}[H]$. The computation proceeds as follows:

Step 1: Compute the semi-join reductions for $\mathcal{V}$ in two rounds, using load m/p' (see Lemma 4). We note that $m/p' = m/(c \cdot p^{|L|/|vars(\mathcal{Q})|}) \leq m/(c \cdot p^{2/|vars(\mathcal{Q})|}) = m/(c \cdot p^{1/\rho^*(\mathcal{Q})})$, because $|L| \geq 2$.

Step 2: Compute the reduced query $\mathcal{Q}_v$ of $\mathcal{Q}[H]$, on the semi-join reduction from the previous step, in one round, using the $2/|vars(\mathcal{Q}_v)| = 2/|L|$-load algorithm in Lemma 11; recall that $\mathcal{Q}_v$ is simple, hence Lemma 11 applies.

We still need to prove *(1)* all tuples in $I_{\boldsymbol{h},H}$ are "light" (as per Lemma 11) for the residual query $\mathcal{Q}[H]$, hence the application of HC in Lemma 11 is correct, and *(2)* the load is within our budget. For *(1)* note that, by assumption, I is a $(H, 1/|vars(\mathcal{Q})|)$-compatible instance, hence every value of a variable $\notin H$ has degree $\leq m/p^{1/|vars(\mathcal{Q})|}$. Since $L = vars(\mathcal{Q}_v)$, it follows that its degree is also $\leq m/(p')^{1/|L|}$, proving that it is light according to Lemma 11. For *(2)*, we note that the load of the HC algorithm given by Lemma 11 is $m/(p')^{2/|vars(\mathcal{Q}_v)|} = m/(p')^{2/|L|} = m/p^{1/\rho^*(\mathcal{Q})}$, proving that this is a $\rho^*(\mathcal{Q})$-load algorithm. □

Example 6. *For an example illustrating the above algorithm, consider query $\mathcal{Q}$,*

$$\begin{aligned}\mathrm{H}(\boldsymbol{x}) \text{ :- } & \mathrm{R}_1(x_1,x_2), \mathrm{R}_2(x_2,x_3), \mathrm{R}_3(x_3,x_4), \mathrm{R}_4(x_4,x_5),\\ & \mathrm{R}_5(x_5,x_6), \mathrm{R}_6(x_6,x_1), \mathrm{R}_7(x_2,x_5), \mathrm{R}_8(x_3,x_6)\end{aligned}$$

and heavy-hitter configuration $\Psi = (H, 1/6)$ with $H = \{x_1, x_4\}$. For $\mathcal{Q}$, $\tau^*(\mathcal{Q}) = \rho^*(\mathcal{Q}) = 3$, and $\tau^*(\mathcal{Q}[H]) = 4$, which shows at least two rounds are necessary to compute $\mathcal{Q}$ with desired load over a Ψ-compatible instance. Residual query $\mathcal{Q}[H]$ takes the form:

$$\begin{aligned}\overline{\mathrm{H}}(\boldsymbol{x}') \text{ :- } & \overline{\mathrm{R}}_1(x_2), \overline{\mathrm{R}}_2(x_2,x_3), \overline{\mathrm{R}}_3(x_3), \overline{\mathrm{R}}_4(x_5),\\ & \overline{\mathrm{R}}_5(x_5,x_6), \overline{\mathrm{R}}_6(x_6), \overline{\mathrm{R}}_7(x_2,x_5), \overline{\mathrm{R}}_8(x_3,x_6).\end{aligned}$$

Semi-join decomposition $\mathcal{V}$ for $\mathcal{Q}[H]$ has the following semi-join reductions:

$$\begin{aligned}\overline{\mathrm{R}}_2^v(x_2,x_3) &\text{ :- } \overline{\mathrm{R}}_2(x_2,x_3), \overline{\mathrm{R}}_1(x_2), \overline{\mathrm{R}}_3(x_3).\\ \overline{\mathrm{R}}_5^v(x_5,x_6) &\text{ :- } \overline{\mathrm{R}}_5(x_5,x_6), \overline{\mathrm{R}}_4(x_5), \overline{\mathrm{R}}_6(x_6).\\ \overline{\mathrm{R}}_7^v(x_2,x_5) &\text{ :- } \overline{\mathrm{R}}_7(x_2,x_5).\\ \overline{\mathrm{R}}_8^v(x_3,x_6) &\text{ :- } \overline{\mathrm{R}}_8(x_3,x_6).\end{aligned}$$

As there are only $4 \cdot p^{1/3}$ heavy-hitter tuples for x_1 and x_4, we can compute $\mathcal{Q}$ for every heavy-hitter tuple h_1, h_4 separately over the remaining $p^{2/3}$ servers.[3] *The semi-joins can be computed with load $m/p^{2/3}$. Then, it remains to compute reduced query $\mathcal{Q}_v$,*

$$\begin{aligned}\mathrm{H}^v(x_2,x_3,x_5,x_6) \text{ :- } & \overline{\mathrm{R}}_2^v(x_2,x_3), \overline{\mathrm{R}}_5^v(x_5,x_6),\\ & \overline{\mathrm{R}}_7^v(x_2,x_5), \overline{\mathrm{R}}_8^v(x_3,x_6),\end{aligned}$$

using the HC algorithm, which can be done with load $m/p^{1/3}$ in one round over the assigned fraction of $p^{2/3}$ servers. Finally, we need to compute the cross product of the locally found tuples for $\mathcal{Q}_v$ with the output for query $\mathcal{Q}[vars(\mathcal{Q}) \setminus H]$ over the broadcast facts.

In the next lemma we drop the restriction that $\mathcal{Q}[H]$ can have no isolated variables. When the semi-join decomposition $\mathcal{Q}_v$ is the cartesian product of a simple query, and several isolated unary predicates, the above strategies no longer work. While we could use Lemma 6 to compute separately the connected components of $\mathcal{Q}_v$, the problem is that $\rho^*(\mathcal{Q}_v)$ (which is $\leq \rho^*(\mathcal{Q}[H])$) is too high. In Lemma 13 we used implicitly the fact that $\rho^*(\mathcal{Q}) = |H|/2 + \rho^*(\mathcal{Q}_v)$: this holds because the tight fractional edge packing for $\mathcal{Q}$ is also a tight fractional edge packing for $\mathcal{Q}[H]$ *and* the fact that $\mathcal{Q}_v$ is simple, which implies $\rho^*(\mathcal{Q}[H]) = \rho^*(\mathcal{Q}_v) = |L|/2$. But when $\mathcal{Q}_v$ is not simple, then $\rho^*(\mathcal{Q}_v)$ can be as high as $\rho^*(\mathcal{Q})$ (for example $\mathcal{Q} = H(\boldsymbol{x})$:- $R(x,y), S(y,z), T(z,u), U(u,w), W(w,z)$ has the tight packing $f(R) = 1$, $f(T) = f(U) = f(W) = 1/2$, $f(S) = 0$ and $\rho^*(\mathcal{Q}) = 5/2$, but when $H = \{y\}$ then $\mathcal{Q}[H] = R(x), S(z), T(z,u), U(u,w), W(w,z)$ and $\mathcal{Q}_v = R(x), T(z,u), U(u,w), W(w,z)$ thus $\rho^*(\mathcal{Q}[H]) = 5/2$). Notice that also the technique from Lemma 12 does not help for the example query. To overcome this issue a more rigorous share-allocation technique is needed. Here, we need to assume that the tight fractional edge packing is half integral, and distinguish two cases: when $vars(\mathcal{C}_f(\mathcal{Q})) \cap H$ is $= \emptyset$ or $\neq \emptyset$. (recall that $\mathcal{C}_f(\mathcal{Q})$ is the set of atoms with weight 1/2).

Lemma 14. *Let $\mathcal{Q}$ be a simple CQ. Assume that $\mathcal{Q}$ admits a tight, half-integral edge packing f, and $\Psi = (H, 1/|vars(\mathcal{Q})|)$ is a heavy hitter configuration, where $vars(\mathcal{C}_f(\mathcal{Q})) \cap H = \emptyset$ and $\mathcal{Q}[H]$ has at least one binary atom. ($\mathcal{Q}[H]$ may have isolated variables.) Then $\mathcal{Q}$ can be computed over the Ψ-compatible instance by a $\rho^*(\mathcal{Q})$-load algorithm in three rounds.*

[3] Notice that both x_1 and x_4 are incident to two atoms, thus each have at most $2p^{1/6}$ many heavy-hitters.

Proof. Let I be the Ψ-compatible instance. The definition of L and $\mathcal{Q}[H]$ are as before. Now let $L_1 = \{x_1, \ldots, x_k\}$ be the set of isolated variables in $\mathcal{Q}[H]$. Recall that $vars(\mathcal{Q}[H]) = L$, thus $L_1 \subseteq L$. Let $L_2 = L \setminus L_1$. We have $|L_2| \geq 2$, because $\mathcal{Q}[H]$ has at least one binary atom.

By the assumption that all variables in $\mathcal{C}_f(\mathcal{Q})$ are light it follows that $vars(\mathcal{C}_f(\mathcal{Q})) \subseteq L_2$, and particularly that every variable $x_i \in L_1$ is incident to an atom $R_i(x_i, y_i)$ of $\mathcal{Q}$ with weight 1 under f. Denote $H_1 = \{y_i \mid f(R_i(x_i, y_i)) = 1 \text{ and } x_i \in L_1\}$: thus, each $x_i \in L_1$ chooses a unique $y_i \in H_1$ based on the matching f. $H_1 \subseteq H$ (since x_i became isolated) and $|H_1| = |L_1|$ (because of the matching). Further, define $H_2 = H \setminus H_1$. Denote:

$$p' = c \cdot p^{|L_2|/|vars(\mathcal{Q})|}, \text{ and}$$
$$p_{i,h} = p^{2/|vars(\mathcal{Q})|} \cdot \deg_{R_i}(h, y_i)/m, \text{ for } i \in [k],$$

with $y_i \in H_1$, h a heavy-hitter value for y_i, and $c = 1/d(\mathcal{Q})^{|H_2|}$ a constant. Notice that $\sum_h p_{i,h} \leq p^{2/|vars(\mathcal{Q})|}$. Further, denote $p_{\boldsymbol{h}} = p' \cdot \prod_{i \in [k]} p_{i,h_i}$, with $\boldsymbol{h} = (h_1, \ldots, h_k)$.

As in Lemma 13, we compute $\mathcal{Q}$ for every heavy-hitter tuple for variables H separately, using item 2 of Lemma 5. Only now we make a distinction between the heavy-hitter tuples $\boldsymbol{h}$ for H_1 and $\boldsymbol{g}$ for H_2. To compute $\mathcal{Q}(I|_{\boldsymbol{h},\boldsymbol{g}})$, we need to compute the residual query $\mathcal{Q}[H]$, whose semi-join decomposition we denote $\mathcal{Q}_v$: we will compute this query using $p_{\boldsymbol{h}}$ exclusive servers (note that this number depends on $\boldsymbol{h}$ but not on $\boldsymbol{g}$). We do not exceed budget p, because

$$\sum_{\boldsymbol{h}} p_{\boldsymbol{h}} = p' \prod_{i \in [k]} (\sum_{h_i} p_{i,h_i}) \leq p' \cdot p^{2|H_1|/|vars(\mathcal{Q})|}$$
$$= c \cdot p^{(|H_1|+|L_1|+|L_2|)/|vars(\mathcal{Q})|},$$

there are at most $p^{|H_2|/|vars(\mathcal{Q})|}/c$ heavy-hitter tuples $\boldsymbol{g}$, and $|L_1| + |H_1| + |H_2| + |L_2| = |vars(\mathcal{Q})|$.

We can compute the semi-join reductions for $\mathcal{V}$ over the assigned $p_{\boldsymbol{h}}$ servers using Lemma 4. The latter is with desired load because $p' \geq p^{2/|vars(\mathcal{Q})|}$ (due to $|L_2| \geq 2$). It remains to show how to compute $\mathcal{Q}_v$ using $p_{\boldsymbol{h}}$ servers. The query $\mathcal{Q}_v$ consists of the cross product:

$$\mathcal{Q}_v = \mathcal{Q}'_v \times \overline{\mathrm{R}}_1^v(x_1) \times \cdots \times \overline{\mathrm{R}}_k^v(x_k) \quad (6)$$

where $\mathcal{Q}'_v$ is a semi-join decomposition for $\mathcal{Q}[H \cup L_1]$, and $\overline{\mathrm{R}}_i^v(x_i)$, $i \in [k]$ are the isolated unary relations obtained from the binary relations $R_i(x_i, y_i)$ with $x_i \in L_1$ and $y_i \in H_1$; notice that the size of $\overline{\mathrm{R}}_i^v(x_i)$ is $\deg_{R_i}(h_i, y_i)$. To compute this cross product, we organize the $p_{\boldsymbol{h}}$ servers into a $(k+1)$-dimensional hypercube, of sizes $p' \cdot \prod_{i \in [k]} p_{i,h_i}$: we compute the query $\mathcal{Q}'_v$ using the p' servers of the first dimension, and hash-partition $\overline{\mathrm{R}}_i^v(x_i)$ to the p_{i,h_i} servers in its dimension: the load is as desired because $\tilde{O}\left(|\overline{\mathrm{R}}_i^{vI_{\boldsymbol{h},\boldsymbol{g}}}|/p_{i,h_i}\right) = \tilde{O}\left(\deg_{R_i}(h_i, y_i)/p_{i,h_i}\right) \leq \tilde{O}\left(m/p^{2/|vars(\mathcal{Q})|}\right)$.

Notice that the query $\mathcal{Q}'_v$ is computed repeatedly, namely $\prod_{i \in [k]} p_{i,h_i}$ times, while each relation $\overline{\mathrm{R}}_i^v(x_i)$ is replicated $p' \cdot \prod_{j \in [k], j \neq i} p_{j,h_j}$ times.

It remains to show how to compute $\mathcal{Q}'_v$ using p' servers. $\mathcal{Q}_{\mathcal{V}'}$ is a simple query (since it is obtained by removing isolated unary relations from $\mathcal{Q}_v$) and all tuples in $I_{\boldsymbol{h},H}$ over the modified schema of $\mathcal{Q}_{\mathcal{V}'}$ are "light", thus we can use the algorithm of Lemma 11. Particularly, the load of the HC algorithm given by Lemma 11 is $m/(p')^{2/|vars(\mathcal{Q}_{\mathcal{V}'})|} = m/(p')^{2/|L_2|} = m/p^{2/|vars(\mathcal{Q})|} = m/p^{1/\rho^*(\mathcal{Q})}$, proving that this is a $\rho^*(\mathcal{Q})$-load algorithm.

To summarize, the algorithm is the following. For each heavy-hitter tuple $\boldsymbol{h}$ for H_1 and $\boldsymbol{g}$ for H_2, use $p_{\boldsymbol{h}}$ exclusive servers to compute $\mathcal{Q}(I|_{\boldsymbol{h},\boldsymbol{g}})$ as follows:

Step 1: Compute the semi-join reductions for $\mathcal{V}$ in two rounds over instance $I|_{\boldsymbol{h},\boldsymbol{g}}$ as in Lemma 4.

Step 2: Compute the cross product in Eq.(6) by organizing the $p_{\boldsymbol{h}}$ servers into a $(k+1)$-dimensional hypercube. Along the first dimension, compute the query $\mathcal{Q}'_v$ (the result of the semi-join reduction in the first step) using p' servers using the algorithm of Lemma 11, by assigning a weight $f(x) = 1/2$ to each variable. Along all other dimensions, hash partition $\overline{\mathrm{R}}_i^v(x_i)$ (the result of the semi-join reduction in the first step). □

Lemma 14 provides an alternative to the strategy in [11] to compute even-length cycles.

Example 7. *For an example consider the even-length cycle $C_8 = \{R_i(x_i, x_{(i \bmod 8)+1}) \mid i \in [8]\}$ and heavy-hitter configuration $\Psi = (H, 1/8)$, and $H = \{x_1, x_2, x_3, x_6, x_7\}$. There is one isolated variable, $L_1 = \{x_8\}$ and, assuming $f(R_i) = (i \bmod 2)$, we pair x_8 with $H_1 = \{x_7\}$. Partition the p available servers in groups of size $p_{\boldsymbol{h}} = p^{4/8} \cdot deg_{R_7}(h, x_7)/m$, one group for each heavy-hitter tuple $\boldsymbol{h}$ over H, where h denotes the value for variable $x_7 \in H$ in $\boldsymbol{h}$. Let semi-join decomposition for $\mathcal{Q}[H]$ be:*

$$\overline{\mathrm{R}}_7^v(x_8) \text{ :- } \overline{\mathrm{R}}_8(x_8), \overline{\mathrm{R}}_7(x_8).$$
$$\overline{\mathrm{R}}_4^v(x_4, x_5) \text{ :- } \overline{\mathrm{R}}_4(x_4, x_5), \overline{\mathrm{R}}_3(x_4), \overline{\mathrm{R}}_5(x_5).$$

Query $\mathcal{Q}_v$ takes the form:

$$\mathrm{H}^v(x_4, x_5, x_8) \text{ :- } \overline{\mathrm{R}}_4^v(x_4, x_5), \overline{\mathrm{R}}_7^v(x_8).$$

Compute for each heavy-hitter tuple $\boldsymbol{h}$ the respective semi-joins in $\mathcal{V}$ for $I|_{\boldsymbol{h}}$ over its fragment of $p_{\boldsymbol{h}}$ servers. Hereafter, hash-partition the tuples in R_4^v over a fragment of $p^{2/8}$ servers and hash-partition the tuples in $\overline{R'}^v_7$ over the remaining fragment of $p^{2/8} \cdot deg_{R_7}(h, x_7)/m$ servers. Further broadcast all heavy-hitter tuples in $I|_{\boldsymbol{h}}$. Then by computing $\mathcal{Q}_v$ locally on every server and taking the cross product with the locally computed output for $\mathcal{Q}[vars(C_8) \setminus H]$ over the broadcast heavy-hitter tuples, we obtain the output for $\mathcal{Q}$.

Finally, we generalize Lemma 14 to the case where some variables in $\mathcal{C}_f(\mathcal{Q})$ may be heavy.

Lemma 15. *Let $\mathcal{Q}$ be a simple CQ. Assume $\mathcal{Q}$ admits a tight, half-integral edge packing f, and $\Psi = (H, 1/|vars(\mathcal{Q})|)$ is a heavy hitter-configuration, where $\mathcal{Q}[H]$ has at least one binary atom. ($vars(\mathcal{C}_f(\mathcal{Q})) \cap H$ may be $\neq \emptyset$ and $\mathcal{Q}[H]$ may have isolated variables.) Then $\mathcal{Q}$ can be computed over the Ψ-compatible instance by a $\rho^*(\mathcal{Q})$-load algorithm using five rounds.*

Proof. Let $C_1, \ldots, C_\ell$ be the odd-length cycles in $\mathcal{C}_f(\mathcal{Q})$ with heavy variables. For every cycle C_i we (arbitrarily) choose one of its heavy variables and call it x_i. Let H' be the set of selected variables and let $\mathcal{Q}[H']$ be the residual query obtained by removing these variables from $\mathcal{Q}$. Let $L' = vars(\mathcal{Q}) \setminus H' = vars(\mathcal{Q}[H'])$.

To compute $\mathcal{Q}$ we again make use of item 2 of Lemma 5, by computing $\mathcal{Q}$ for every heavy-hitter tuple $\boldsymbol{h}$ for H' separately over an exclusive group of $p' = c \cdot p^{|L'|/|vars(\mathcal{Q})|}$ servers, with p the total number of available servers and $c = 1/d(\mathcal{Q})^{|H'|}$ a constant. There are at most $p^{|H'|/|vars(\mathcal{Q})|}/c$ many heavy tuples over H', and $p' \cdot p^{|H'|/|vars(\mathcal{Q})|}/c = p$, thus the number of used servers is within budget.

Interestingly, for the semi-join decomposition $\mathcal{V}$ of $\mathcal{Q}[H']$, query $\mathcal{Q}_v$ satisfies the conditions of Lemma 14. For this to see let f' be the fractional edge packing obtained from f by updating all interrupted 1/2-weighted cycles in $\mathcal{Q}_v$ to paths with alternating weights 1 and 0. More formally, for each cycle $C_i = a_1, \ldots, a_k$ where a_1 and a_k denote the atoms involving variable x_i, define $f'(a_i^v) = 0$ if i is odd, and $f'(a_i^v) = 1$ if i is even. For all other atoms $a^v \in \mathcal{Q}_v$, let $f'(a^v) = f(a)$. Notice that all atoms a_i^v are in $\mathcal{Q}_v$ except for a_1^v and a_k^v. From $\{a \mid a^v \in atoms(\mathcal{Q}_v)\} \subseteq atoms(\mathcal{Q})$ it follows that f' describes a fractional edge packing for $\mathcal{Q}_v$. By construction, f' is tight for $\mathcal{Q}_v$ and thus $\tau^*(\mathcal{Q}_v) = \rho^*(\mathcal{Q}_v) = |vars(\mathcal{Q}_v)|/2 = \rho^*(\mathcal{Q}) - |H'|/2$.

The computation of $\mathcal{Q}[H']$ now proceeds by first computing the semi-join reductions for $\mathcal{V}$ over relations where $\mathcal{Q}[H]$ is based on heavy-hitter tuple $\boldsymbol{h}$ and then computing $\mathcal{Q}_v$ over the semi-join reduced relations for configuration $(H', 1/|L'|)$ using the algorithm from Lemma 14 and 6 over the assigned p' servers. Notice that the semi-join reductions can be computed with desired load, as $p' \geq p^{1/\rho^*(\mathcal{Q})}$ (because there is at least one cycle C_i, with $|vars(C_i)| \geq 3$, and only one of its variables is removed). Further notice that $m/(p')^{2/|L'|} = m/p^{2/|vars(\mathcal{Q})|}$. □

Example 8. *For an example consider the query $\mathcal{Q}$,*

$$\mathrm{H}(\boldsymbol{x}) \text{ :- } \mathrm{R}_1(x_1, x_2), \mathrm{R}_2(x_2, x_3), \mathrm{R}_3(x_3, x_4), \mathrm{R}_4(x_4, x_5), \mathrm{R}_5(x_5, x_3),$$

and heavy-hitter configuration $\Psi = (H, \delta)$ with $H = \{x_4\}$ and $\delta = 1/5$. To compute $\mathcal{Q}$ assign to every heavy-hitter value h a unique fraction of $p' = p^{4/5}$ servers. As there at most $p^{1/5}$ many values h, [4] we do not exceed budget p. To compute $\mathcal{Q}$ first consider the residual query $\mathcal{Q}[H]$ and semi-join reduce the relations accordingly, with $\mathcal{Q}[H]$ as below:

$$\overline{\mathrm{H}}(\boldsymbol{x'}) \text{ :- } \overline{\mathrm{R}}_1(x_1, x_2), \overline{\mathrm{R}}_2(x_2, x_3), \overline{\mathrm{R}}_3(x_3), \overline{\mathrm{R}}_4(x_5), \overline{\mathrm{R}}_5(x_5, x_3),$$

which is simplified to query $\mathcal{Q}_v$ by use of a semi-join decomposition. Here $\mathcal{Q}_v$ takes the form:

$$\mathrm{H}^v(\boldsymbol{x'}) \text{ :- } \overline{\mathrm{R}}_1^v(x_1, x_2), \overline{\mathrm{R}}_2^v(x_2, x_3), \overline{\mathrm{R}}_5^v(x_5, x_3).$$

Now we have $\rho^*(\mathcal{Q}) = 5/2$ and $\rho^*(\mathcal{Q}_v) = 2$. For $\mathcal{Q}_v$ we know how to compute it with load $m/p^{1/2}$, thus over a fraction p' of the servers we obtain load $m/p^{2/5}$ as desired.

[4] Technically there can be $2 \cdot p^{1/5}$ heavy values, which we ignore, as this influences the load only by a constant factor.

Combining the above Lemmas we obtain:

Theorem 1. *Let $\mathcal{Q}$ be a simple CQ that admits a tight, half-integral fractional edge packing. Then $\mathcal{Q}$ can be computed by a $\rho^*(\mathcal{Q})$-load algorithm using five rounds.*

Proof. To compute query $\mathcal{Q}$ over arbitrary instance I, one can simply compute $\mathcal{Q}$ for each heavy-hitter configuration $\Psi = (H, 1/|vars(\mathcal{Q})|)$ separately, with $H \subseteq vars(\mathcal{Q})$, over instance I_Ψ following the algorithms in Lemma 12 and 15. Correctness follows from Lemma 5. Particularly notice that the load increases only by constant $2^{|vars(\mathcal{Q})|}$, which corresponds to the number of possible heavy-hitter configurations. □

5.4 Non-Tight Edge Packings

For CQ $\mathcal{Q}$, let $\mathcal{N}_f(\mathcal{Q})$ denote the set of variables in $\mathcal{Q}$ incident to only 0-weighted predicates under fractional edge packing f. Given that f is optimal and maximizes the number of 0-weighted predicates, $\mathcal{N}_f(\mathcal{Q})$ denotes exactly those variables witnessing non-tightness of f.

Example 9. *For an example consider query $\mathcal{Q}$,*

$$\mathrm{H}(\boldsymbol{x}) \text{ :- } \mathrm{R}_1(x_1, x_2), \mathrm{R}_2(x_2, x_3), \mathrm{R}_3(x_3, x_4), \mathrm{R}_4(x_4, x_5), \mathrm{R}_5(x_5, x_3), \mathrm{R}_6(x_6, x_2),$$

which is essentially the same query as in Example 8 except for additional atom R_6. Query $\mathcal{Q}$ has no tight fractional edge packing due to predicates R_1 and R_6. Here, $\tau^*(\mathcal{Q}) = 5/2$, and $\rho^*(\mathcal{Q}) = 7/2$. We divide the p servers in shares $p_1 = p^{2/7}$ and $p_2 = p^{5/7}$. Now hash-partition the tuples for R_6 over share p_1 and proceed the computation of $\mathcal{Q}$ without predicate R_6 over share p_2. Notice that the latter query now has a tight edge packing, we can thus proceed as in Example 8.

We first show the case where none of the variables in $\mathcal{N}_f(\mathcal{Q})$ are heavy. Notice that we now consider a different threshold value.

Lemma 16. *Let $\mathcal{Q}$ be a simple CQ $\mathcal{Q}$ with non-tight half-integral edge packing f, and heavy-hitter configuration $\Psi = (H, 1/\rho^*(\mathcal{Q}))$, satisfying $\mathcal{N}_f(\mathcal{Q}) \cap H = \emptyset$. Then, $\mathcal{Q}$ can be computed over the Ψ-compatible instance by a $\rho^*(\mathcal{Q})$-load algorithm using five rounds.*

Proof. As before, let $L = vars(\mathcal{Q}) \setminus H$. By assumption, $\mathcal{N}_f(\mathcal{Q}) \subseteq L$ and $\mathcal{N}_f(\mathcal{Q}) \neq \emptyset$, as otherwise f is tight.

Notice that all atoms in $\mathcal{Q}$ have at least some variable not in $\mathcal{N}_f(\mathcal{Q})$, as otherwise f cannot be optimal. Indeed, then one can obtain a fractional edge packing with higher value by simply assigning weight 1 to the selected predicate.

We partition the atoms of $\mathcal{Q}$ in two sets A and B. Set A contains all the atoms from $\mathcal{Q}$ where variables in $\mathcal{N}_f(\mathcal{Q})$ are incident to, and $B = atoms(\mathcal{Q}) \setminus A$.

Now compute $\mathcal{Q}$ by computing the cross product $\mathcal{Q}' = \mathcal{Q}_A \times \mathcal{Q}_B$, and then post process the answer by selecting only those tuples where values for variables of $\mathcal{Q}_A$ equal the values for corresponding variables in $\mathcal{Q}_B$. Observe that this indeed provides us the desired output result.

To compute $\mathcal{Q}_A$ and $\mathcal{Q}_B$, consider share definitions:

$$p_1 = p^{\tau^*(\mathcal{Q}_1)/\rho^*(\mathcal{Q})}$$
$$p_2 = p^{\rho^*(\mathcal{Q}_2)/\rho^*(\mathcal{Q})}.$$

We first argue $p_1 \cdot p_2 = p$. For this, notice that $\tau^*(\mathcal{Q}_A) = |\mathcal{N}_f(\mathcal{Q})|$, which is because $\mathcal{N}_f(\mathcal{Q})$ defines an (integral) vertex cover for $\mathcal{Q}_A$, and one can easily obtain an edge packing with same value by selecting for each variable in $\mathcal{N}_f(\mathcal{Q})$ exactly one incident atom.[5] Further notice that $\tau^*(\mathcal{Q}) = |\mathcal{C}_f(\mathcal{Q})|/2 + (|vars(\mathcal{Q})| - |\mathcal{C}_f(\mathcal{Q})| - |\mathcal{N}_f(\mathcal{Q})|)/2$, thus from Lemma 10 we obtain $\rho^*(\mathcal{Q}) = |vars(\mathcal{Q})|/2 + |\mathcal{N}_f(\mathcal{Q})|/2$. The result now follows by the additional observation that $\rho^*(\mathcal{Q}_B) = vars(\mathcal{Q}_B)/2$ due to tightness of f for $\mathcal{Q}_B$, where $vars(\mathcal{Q}_B) = vars(\mathcal{Q}) \setminus \mathcal{N}_f(\mathcal{Q})$.

We compute $\mathcal{Q}_A$ over a share of p_1 servers by allocating to all the variables in $\mathcal{N}_f(\mathcal{Q})$ a weight of 1 and to all others a weight 0. This strategy works as desired because I_Ψ is light for all variables in $\mathcal{N}_f(\mathcal{Q})$, and the meaning of light is not affected by considering only p_1 servers. Indeed,

$$m/p^{1/\rho^*(\mathcal{Q})} = m/p_1^{1/\tau^*(\mathcal{Q}_1)}$$

and $\tau^*(\mathcal{Q}_A) = |\mathcal{N}_f(\mathcal{Q})|$. Following Lemma 2 the load is as desired and requires only one round.

The computation of $\mathcal{Q}_B$ over I_Ψ relies on Theorem 1 using a share of p_2 servers. This algorithm uses at most five rounds, and by choice of p_2 the load is as desired. □

Finally, in Lemma 17 we deal with the case where variables in $\mathcal{N}_f(\mathcal{Q})$ are heavy:

Lemma 17. *Let $\mathcal{Q}$ be a simple CQ with heavy-hitter configuration $\Psi = (H, 1/\rho^*(\mathcal{Q}))$. Let f be a non-tight half-integral edge packing for $\mathcal{Q}$ with $\mathcal{N}_f(\mathcal{Q}) \cap H \neq \emptyset$. Then $\mathcal{Q}$ can be computed over the Ψ-compatible instance by a $\rho^*(\mathcal{Q})$-load algorithm using seven rounds.*

[5] This strategy works because A consists of disjoint star-queries, one for each variable in $\mathcal{N}_f(\mathcal{Q})$.

Proof. Let I be an arbitrary Ψ-compatible instance. Let $H' = \mathcal{N}_f(\mathcal{Q}) \cap H$, and $\mathcal{Q}[H']$ be the residual query obtained by removing from $\mathcal{Q}$ the variables in H', and $L' = vars(\mathcal{Q}) \setminus H'$.

Let $\mathcal{V}$ be a semi-join decomposition for $\mathcal{Q}[H']$. Fractional edge packing f describes a fractional edge packing for $\mathcal{Q}_v$ when ignoring the annotation of relation symbols. Particularly, then f is also maximal for $\mathcal{Q}_v$, as only 0-weight atoms are removed from $\mathcal{Q}$ to obtain $\mathcal{Q}_v$, and non-optimality would thus contradict the assumed optimality of f for $\mathcal{Q}$. Therefore $\tau^*(\mathcal{Q}_v) = \tau^*(\mathcal{Q})$.

Similar as before we compute $\mathcal{Q}$ for every heavy hitter tuple $\boldsymbol{h}$ over H' separately over a unique set of p' servers, where $p' = c \cdot p^{\rho^*(\mathcal{Q}_v)/\rho^*(\mathcal{Q})}$, and $c = 1/d(\mathcal{Q})^{|H'|}$ a constant. As there are only $p^{|vars(H')|/\rho^*(\mathcal{Q})}/c$ many such tuples we remain within budget. Indeed, $\rho^*(\mathcal{Q}) = |vars(\mathcal{Q})| - \tau^*(\mathcal{Q}_v) = \rho^*(\mathcal{Q}_v) + |H'|$, following Lemma 10, $|vars(\mathcal{Q})| = |vars(\mathcal{Q}_v)| + |H'|$, and the earlier made observation $\tau^*(\mathcal{Q}) = \tau^*(\mathcal{Q}_v)$. The computation then proceeds in two steps:

Step 1: Compute the semi-join reductions defined by $\mathcal{V}$ over instance $I|_{\boldsymbol{h},H'}$ using the strategy in Lemma 4.

Step 2: Reshuffle the semi-join reduced relations from the previous step to compute $\mathcal{Q}_v$ by following the algorithm in Lemma 16. Then broadcast heavy-hitter tuples in $I|_{\boldsymbol{h},H'}$ over the schema of $\mathcal{Q}[L']$, and compute locally on every server the crossproduct $\mathcal{Q}[L'] \times \mathcal{Q}_v$ to obtain the desired output.

Completeness of the output follows from Lemma 16 and Lemma 5. Step 1 is done in two rounds with desired load because $p' \geq c \cdot p^{1/\rho^*(\mathcal{Q})}$. The latter particularly follows from the fact that $\mathcal{Q}_v$ does not contain isolated variables and cannot be empty, as both would contradict the assumed optimality of f for $\mathcal{Q}$.

For Step 2, notice that $\mathcal{Q}_v$ and f indeed satisfy the conditions of Lemma 16 and thus the obtained load using at most five rounds is $\tilde{O}\left(m/p^{1/\rho^*(\mathcal{Q})}\right)$. □

Similar to Theorem 1, for conjunctive queries with non-tight half-integral edge packing we now obtain the following result:

Theorem 2. *Let $\mathcal{Q}$ be a simple CQ and f a non-tight half-integral edge packing. Then $\mathcal{Q}$ can be computed by a $\rho^*(\mathcal{Q})$-load algorithm using seven rounds.*

Proof. for every heavy-hitter configuration $(H, 1/\rho^*(\mathcal{Q}))$ in parallel, using Lemma 16 if $vars(\mathcal{N}_f(\mathcal{Q})) \cap H = \emptyset$, and the algorithm in Lemma 17 otherwise. □

5.5 Wrap-Up

By combining the two theorems we obtain a general algorithm for computing conjunctive queries over relations with arity at most two: first simplify the query by considering a semi-join decomposition $\mathcal{V}$ for $\mathcal{Q}$.

Step 1: Compute a semi-join decomposition $\mathcal{Q}_v$ using the algorithm in Lemma 8.

Step 2: Compute a half-integral fractional edge packing f for $\mathcal{Q}_v$. If f is tight then use the algorithm in Theorem 1; if f is non-tight use the algorithm in Theorem 2.

Hence our main result:

Theorem 3. *Every CQ $\mathcal{Q}$ over relations with arity at most two can be computed with a $\rho^*(\mathcal{Q})$-load algorithm using at most nine rounds.*

As our algorithm takes a fractional edge packing as parameter, notice that, in contrast to finding a maximum vertex cover for a given graph, which is well-known to be NP-complete, the problem of finding maximum (integral) edge packings (i.e., maximum matchings), is solvable in polynomial time. Interestingly, finding optimal half-integral edge packings for a graph (or simple query) with the properties of Lemma 9 is comparably easy. For a polynomial time algorithm based on iteratively finding augmenting paths (i.e., paths with alternating assignment of weight 0 and 1 to its edges) and blossoms (i.e., odd-length cycles) we refer to [7].

When a query has multiple optimal fractional edge packings satisfying the conditions of Lemma 9, then one of the packings may yield a preferred evaluation strategy over the other:

Example 10. *For an example consider query $\mathcal{Q}$,*

$$\begin{aligned}\mathrm{H}(\bar{x}) \text{ :- } & \mathrm{R}_1(x_1,x_2),\mathrm{R}_2(x_2,x_3),\mathrm{R}_3(x_3,x_1),\mathrm{R}_4(x_3,x_4),\\ & \mathrm{R}_5(x_4,x_5),\mathrm{R}_6(x_5,x_6),\mathrm{R}_7(x_6,x_4),\end{aligned}$$

which represents two triangles connected by an edge. Here $\rho^*(\mathcal{Q}) = \tau^*(\mathcal{Q}) = 3$. We can consider the fractional edge packing assigning weight $1/2$ to predicates R_1, R_2, R_3, R_5, R_6, R_7 and 0 to predicate R_4, or alternatively the edge packing assigning weight 1 to predicates R_1, R_4, R_6, and 0 to all others. The latter seems beneficial as Lemma 14 guarantees three rounds in contrast to five for Lemma 15, particularly if certain variables are heavy.

Notice that disallowing multiple occurrences of the same variable in a single atom is without loss of generality. Indeed, given a query that violates this condition, we can simply ignore duplicates variable occurrences by implementing a preprocessing step in which each server autonomously discards those tuples not satisfying the by the query implied equality types.

Finally, notice that our algorithm can be generalized to conjunctive queries with existential quantification (projections) and self-joins in the usual way by duplicating the self-joined relations and implementing projections as a post processing step. Then, however, we inevitably lose optimality as the AGM bound is no longer the right notion to express the worst-case output size [8].

5.6 Higher-Arity Relations

Unfortunately it is unclear how to generalize our techniques to higher-arity relations, as the graph properties we rely on do not generalize to hypergraphs. More specifically, they fail already for ternary relations. First, in general the inequality $\tau^*(\mathcal{Q}) \leq \rho^*(\mathcal{Q})$ does not hold.

Example 11. *For an example of a query $\mathcal{Q}$, where $\tau^*(\mathcal{Q}) > \rho^*(\mathcal{Q})$, consider*

$$\begin{aligned}\mathrm{H}(\boldsymbol{x}) \text{ :- } & \mathrm{R}(x_1,y_1,z_1),\mathrm{T}(x_2,y_2,z_2),\\ & \mathrm{S}_1(x_1,x_2),\mathrm{S}_2(y_1,y_2),\mathrm{S}_3(z_1,z_2).\end{aligned}$$

Here, $\tau^*(\mathcal{Q}) = 3$, which is easy to see as all three predicates S_1, S_2, S_3 need to be covered, and do not share variables. On the other hand, $\rho^*(\mathcal{Q}) = 2$ is witnessed by the cover containing predicates R and T.

Therefore, taking the HC algorithm as a building block seems to not work in general for higher-arity relations. Particularly for query $\mathcal{Q}$ from Example 11, it is even unclear how to compute $\mathcal{Q}$ with $\rho^*(\mathcal{Q})$-based load over instances without skew. Unless, for example, in the special case where all columns of relations are keys, as then intermediate join outputs can grow only by a constant factor with respect to the input (the factor is caused by the arity of the output relation). In fact, the only class of queries over higher-arity relations that is currently known to be computable with $\rho^*(\mathcal{Q})$-load algorithm are those having a guard atom, as in Lemma 4.

Second, $\rho^*(\mathcal{Q}) + \tau^*(\mathcal{Q}) = |vars(\mathcal{Q})|$ in general does not hold for hypergraphs either. Again, this can be observed by considering query $\mathcal{Q}$ from Example 11. There, $\rho^*(\mathcal{Q}) + \tau^*(\mathcal{Q}) = 5 < |vars(\mathcal{Q})|$.

6 External Memory Model

Recently, much attention went also to worst-case I/O-optimal algorithms for computing join queries in the external memory model [9, 10]. The external memory model is a sequential computation model in which a distinction is made between space for data storing (external memory) and processing (internal memory). The former is unbounded, the latter is bounded by parameter w, which expresses its size in number of disc blocks: a disc block is the amount of data, denoted B, moved from external to internal memory (or vice versa) in a single I/O operation. For a more comprehensive description of the model we refer to [2].

In [11] it was observed that algorithms for the MPC model can be simulated straightforwardly in the external memory model, given that they are tuple-based, and that the internal memory is at least as large as the maximal

amount of space needed by any of the individual servers (i.e., $w \geq r \cdot L/B$).

The tuple-based MPC model is the MPC model, as described in Section 3, where additionally only tuples from subqueries of $\mathcal{Q}$ are communicated, and the communication depends only on the data statistics that are initially available to the algorithm. As the algorithm from Section 5 satisfies these conditions, the following Corollary follows steadily from the results in [11].

Corollary 1. *For every CQ $\mathcal{Q}$ over relations with arity at most two, an external memory algorithm exists that, for any $w > m^{\rho^*(\mathcal{Q})/(\rho^*(\mathcal{Q})+1)}$, computes Q with $\tilde{O}\left(\frac{m^{\rho^*(\mathcal{Q})}}{w^{\rho^*(\mathcal{Q})-1} \cdot B}\right)$ I/O cost, by simulating the algorithm from Section 5 with $p = (m/w)^{\rho^*(\mathcal{Q})}$.*

7 Conclusion

In this paper we introduced an algorithm for computing conjunctive queries over binary atoms in parallel shared-nothing settings. Our algorithm runs with load $m/p^{1/\rho^*(\mathcal{Q})}$, and therefore obtains the worst-case optimal load when relations have the same size m. In combination with the lower bound given in [11], it shows that the load in this setting is given by the fractional edge covering number $\rho^*(\mathcal{Q})$ rather than the fractional vertex covering number $\tau^*(\mathcal{Q})$, at least when relations are binary.

Our techniques build upon, and extend in a non-trivial way, techniques introduced in [11] computing chain queries and cycle queries in multiple rounds with worst-case optimal load. Our result critically relies on key properties from graphs, which in general do not hold for hypergraphs, already for arity three. Therefore we leave open the optimal communication cost for queries over relations of arbitrary arity. Particularly it is unclear what the bound might be for queries where $\tau^*(\mathcal{Q}) > \rho^*(\mathcal{Q})$, as in general, the best algorithm we know to compute queries over skew-free database instances is the one-round HC algorithm with load $m/p^{1/\tau^*(\mathcal{Q})}$. On the other hand, the best multi-round lower bound we know follows from the AGM bound and is $m/p^{1/\rho^*(\mathcal{Q})}$. To close the gap one needs to either design a new multi-round algorithm for skew-free databases, or prove a new multi-round bound that is stronger than that implied by the AGM bound.

References

[1] F. N. Afrati and J. D. Ullman. Optimizing joins in a map-reduce environment. In *EDBT*, pages 99–110, 2010.

[2] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Commun. ACM*, 31(9):1116–1127, 1988.

[3] T. J. Ameloot, G. Geck, B. Ketsman, F. Neven, and T. Schwentick. Parallel-correctness and transferability for conjunctive queries. In *PODS*, pages 47–58, 2015.

[4] A. Atserias, M. Grohe, and D. Marx. Size bounds and query plans for relational joins. *SIAM J. Comput.*, 42(4):1737–1767, 2013.

[5] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. In *PODS*, pages 273–284, 2013.

[6] P. Beame, P. Koutris, and D. Suciu. Skew in parallel query processing. In *PODS*, pages 212–223, 2014.

[7] J. Bourjolly and W. R. Pulleyblank. König-egerváry graphs, 2-bicritical graphs and fractional matchings. *Discrete Applied Mathematics*, 24(1-3):63–82, 1989.

[8] G. Gottlob, S. Tien Lee, G. Valiant, and P. Valiant. Size and treewidth bounds for conjunctive queries. *J. ACM*, 59(3):16, 2012.

[9] X. Hu, Y. Tao, and C. Chung. Massive graph triangulation. In *SIGMOD*, pages 325–336, 2013.

[10] X. Hu and K. Yi. Towards a worst-case I/O-optimal algorithm for acyclic joins. In *PODS*, pages 135–150, 2016.

[11] P. Koutris, P. Beame, and D. Suciu. Worst-case optimal algorithms for parallel query processing. In *ICDT*, pages 8:1–8:18, 2016.

[12] P. Koutris and D. Suciu. Parallel evaluation of conjunctive queries. In *PODS*, pages 223–234, 2011.

[13] H. Q. Ngo, E. Porat, C. Ré, and A. Rudra. Worst-case optimal join algorithms: [extended abstract]. In *PODS*, pages 37–48, 2012.

[14] H. Q. Ngo, C. Ré, and A. Rudra. Skew strikes back: new developments in the theory of join algorithms. *SIGMOD Record*, 42(4):5–16, 2013.

[15] E. R. Scheinerman and D. H. Ullman. *Fractional graph theory: a rational approach to the theory of graphs*. Wiley, New York, 1997.

[16] T. L. Veldhuizen. Triejoin: A simple, worst-case optimal join algorithm. In *ICDT*, pages 96–106, 2014.