A Simple Parallel Algorithm for Natural Joins on Binary Relations

Yufei Tao
Chinese University of Hong Kong, Hong Kong
taoyf@cse.cuhk.edu.hk

Abstract

In PODS'17, Ketsman and Suciu gave an algorithm in the MPC model for computing the result of any natural join where every input relation has two attributes. Achieving an optimal load $O(m/p^{1/\rho})$ – where m is the total size of the input relations, p the number of machines, and ρ the fractional edge covering number of the join – their algorithm requires 7 rounds to finish. This paper presents a simpler algorithm that ensures the same load with 3 rounds (in fact, the second round incurs only a load of $O(p^2)$ to transmit certain statistics to assist machine allocation in the last round). Our algorithm is made possible by a new theorem that provides fresh insight on the structure of the problem, and brings us closer to understanding the intrinsic reason why joins on binary relations can be settled with load $O(m/p^{1/\rho})$.

2012 ACM Subject Classification Theory of computation → Database query processing and optimization (theory)

Keywords and phrases Natural Joins, Conjunctive Queries, MPC Algorithms, Parallel Computing

Digital Object Identifier 10.4230/LIPIcs.ICDT.2020.25

1 Introduction

Understanding the computational hardness of joins has always been a central topic in database theory. Traditionally, research efforts (see [1, 4, 8, 11, 12, 14, 15] and the references therein) have focused on discovering fast algorithms for processing joins in the *random access machine* (RAM) model. Nowadays, massively parallel systems such as Hadoop [6] and Spark [2] (`https://spark.apache.org`) have become the mainstream computing architecture for performing analytical tasks on gigantic volumes of data, where direct implementation of RAM join algorithms rarely gives satisfactory performance. A main reason behind this phenomenon is that, while a RAM algorithm is designed to reduce the CPU time, in systems like Hadoop and Spark it is much more important to minimize the amount of communication across the participating machines, because the overhead of delivering all the necessary messages typically overwhelms the cost of CPU calculation. This has motivated a line of research – which also includes this work – that aims to understand the communication complexities of join problems.

1.1 Problem Definition

In this subsection, we will first give a formal definition of *natural join* – the type of joins studied in this paper – and then elaborate on the computation model assumed.

Natural Joins. Let **att** be a countably infinite set where each element is called an *attribute*. Let **dom** be another countably infinite set. A *tuple* over a set $U \subseteq \mathbf{att}$ is a function $\boldsymbol{u} : U \to \mathbf{dom}$. Given a subset V of U, define $\boldsymbol{u}[V]$ as the tuple $\boldsymbol{v}$ over V such that $\boldsymbol{v}(X) = \boldsymbol{u}(X)$ for every $X \in V$. We say that $\boldsymbol{u}[V]$ is the *projection* of $\boldsymbol{u}$ on V.

A *relation* is defined to be a set R of tuples over the same set U of attributes. We say that R is *over* U, and that the *scheme* of R is U, represented with the notation $scheme(R) = U$.

The *arity* of R, denoted as $arity(R)$, equals $|scheme(R)|$. R is *unary* if $arity(R) = 1$, and *binary* if $arity(R) = 2$.

A *join query* is defined as a set $\mathcal{Q}$ of relations. If we let $attset(\mathcal{Q}) = \bigcup_{R\in\mathcal{Q}} scheme(R)$, the result of the query, denoted as $Join(\mathcal{Q})$, is the following relation over $attset(\mathcal{Q})$

$$\Big\{\text{tuple } \boldsymbol{u} \text{ over } attset(\mathcal{Q}) \;\Big|\; \forall R \in \mathcal{Q},\, \boldsymbol{u}[scheme(R)] \in R\Big\}.$$

If $\mathcal{Q}$ has only two relations R and S, we may also use $R \bowtie S$ as an alternative representation of $Join(Q)$. The integer $m = \sum_{R\in\mathcal{Q}} |R|$ is the *input size* of the query. Concentrating on *data complexity*, we will assume that both $|\mathcal{Q}|$ and $|attset(\mathcal{Q})|$ are constants.

A join query $\mathcal{Q}$ is

- *scheme-clean* if no distinct $R, S \in \mathcal{Q}$ satisfy $scheme(R) = scheme(S)$;
- *simple* if (i) $\mathcal{Q}$ is scheme-clean, and (ii) every $R \in \mathcal{Q}$ is binary.

The primary goal of this paper is to design parallel algorithms for processing simple queries efficiently.

Computation Model. We will assume the *massively parallel computation* (MPC) model which has been widely accepted as a reasonable abstraction of the massively parallel systems that exist today. In this model, there are p machines such that at the beginning the input elements are evenly distributed across these machines. For a join query, this means that each machine stores $\Theta(m/p)$ tuples from the input relations.

An algorithm is executed in *rounds*, each of which has two phases:

- In the first phase, each machine does computation on the data of its local storage.
- In the second phase, the machines communicate by sending messages to each other.

It is important to note that all the messages sent out in the second phase must have already been prepared in the first phase. This prevents a machine from, for example, sending information based on what has been received *during* the second phase. Another round is launched only if the problem has not been solved by the current round. In our context, *solving* a join query means that every tuple in the join result has been produced on at least one machine.

The *load* of a round is defined by the largest number of words that is received by a machine in this round, that is, if machine $i \in [1, p]$ receives x_i words, then the load is $\max_{i=1}^{p} x_i$. The performance of an algorithm is measured by two metrics: (i) the number of rounds, and (ii) the *load* of the algorithm, defined to be the largest load incurred by a round, among all the rounds. CPU computation, which takes place in the first phase of each round, is for free.

The number p of machines is assumed to be significantly less than m, which in this paper means $p^3 \leq m$ specifically. All the algorithms to be mentioned, including those reviewed in the next subsection and the ones we propose, are randomized. Their loads are all bounded in a "high probability manner". Henceforth, whenever we say that an algorithm has load at most L, we mean that its load is bounded by L with probability at least $1 - 1/p^2$. Finally, we consider that every value in **dom** can be encoded in a single word.

1.2 Previous Results

Afrati and Ullman [3] showed that any join query can be solved in a single round with load $\tilde{O}(m/p^{1/\min\{k,|\mathcal{Q}|\}})$ where $k = |attset(\mathcal{Q})|$, and the notation $\tilde{O}$ hides polylogarithmic factors. Improving upon the earlier work [5], Koutris, Beame, and Suciu [10] presented another single-round algorithm that solves a join query with load $\tilde{O}(m/p^{1/\psi})$ where ψ is the *fractional quasi-packing number* of the query. They also proved that $\Omega(m/p^{1/\psi})$ is a lower bound on the load of any one-round algorithm, under certain restrictions on the statistics that the algorithm knows about the input relations.

For algorithms that perform more than one, but still $O(1)$, rounds, $\Omega(m/p^{1/\rho})$ has been shown [10] to be a lower bound on the load, where ρ is the *factional edge covering number* of the query. The value of ρ never exceeds, but can be strictly smaller than, ψ, which implies that multi-round algorithms may achieve significantly lower loads than one-round counterparts, thus providing strong motivation for studying the former.

Though matching the lower bound of $\Omega(m/p^{1/\rho})$ algorithmically still remains open for arbitrary join queries, this has been achieved for several special query classes [3, 7, 9, 10]. In particular, Ketsman and Suciu [9] gave an algorithm, henceforth referred to as "the KS algorithm", that solves a simple query in 7 rounds with load $\tilde{O}(m/p^{1/\rho})$. The class of simple queries bears unique significance due to their relevance to *subgraph enumeration*, which is the problem of finding all occurrences of a subgraph $G' = (V', E')$ in a graph $G = (V, E)$. Regarding E as a relation of two attributes, i.e., source vertex and destination vertex, we can convert subgraph enumeration to a join query on $|E'|$ copies of the "relation" E, and renaming the attributes in each relation to reflect how the vertices of V' are connected in G'; see [12] for an example where G' is a clique of 3 vertices.

1.3 Our Contributions

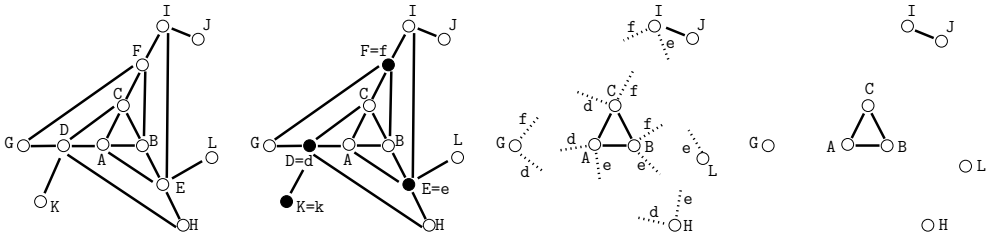
Our main result is that any simple join query can be solved in the MPC model with the optimal load $\tilde{O}(m/p^{1/\rho})$ using 3 rounds. Our algorithm is in fact similar to a subroutine deployed in the KS algorithm, which, however, also demands several other subroutines that entail a larger number of rounds, and are proved to be unnecessary by our solution. The improvement owes to a new theorem that reveals an intrinsic property of the problem, which will be explained shortly with an example. In retrospect, the algorithm of Kestman and Suciu [9] can be regarded as using sophisticated graph-theoretic ideas to compensate for not knowing that property. It is not surprising that their algorithm can be simplified substantially once our understanding on the structure of the problem has been strengthened.

To gain an overview of our techniques, let us consider the join query $\mathcal{Q}$ illustrated by the graph in Figure 1a. An edge connecting vertices X and Y represents a relation $R_{\{X,Y\}}$ with $scheme(R_{\{X,Y\}}) = \{X, Y\}$. $\mathcal{Q}$ is defined by the set of relations represented by the 18 edges in Figure 1a. Notice that $attset(\mathcal{Q}) = \{\mathtt{A}, \mathtt{B}, ..., \mathtt{L}\}$ has a size of 12.

We adopt an idea that is behind nearly all the join algorithms in the MPC model [7, 9, 10], namely, to divide the join result based on "heavy hitters". Let λ be an integer parameter whose choice will be clarified later. A value $x \in \mathbf{dom}$ is *heavy* if an input relation $R \in \mathcal{Q}$ has at least m/λ tuples carrying this value on an attribute $X \in scheme(R)$. The number of heavy values is $O(\lambda)$. A value $x \in \mathbf{dom}$ is *light* if x appears in at least one relation $R \in \mathcal{Q}$ but is not heavy. A tuple in the join result may take a heavy or light value on each of the 12 attributes $\mathtt{A}, ..., \mathtt{K}$. As there are at most $O(\lambda)$ choices on each attribute (namely, light value or one of the $O(\lambda)$ heavy values), there are $O(\lambda^{12})$ "combinations" of choices from all attributes; we will refer to each combination as a *configuration*. If we manage to design an algorithm to find the result tuples under each configuration, executing this algorithm for all configurations solves the query.

Figure 1b illustrates what happens in one of the possible configurations where we constrain attributes `D`, `E`, `F`, and `K` to take heavy values `d`, `e`, `f`, and `k` respectively, and the other attributes to take light values. Accordingly, vertices `D`, `E`, `F`, and `K` are colored black in the figure. This configuration gives rise to a *residue query* $\mathcal{Q}'$ whose input relations are decided as follows:

- For each edge $\{X, Y\}$ with two white vertices, $\mathcal{Q}'$ has a relation $R'_{\{X,Y\}}$ that contains only the tuples in $R_{\{X,Y\}} \in \mathcal{Q}$ using light values on both X and Y;



(a) A join query (b) A residue query (c) After deleting black verticess (d) After semi-join reduction

Figure 1 Processing a join by constraining heavy values.

- For each edge $\{X, Y\}$ with a white vertex X and a black vertex Y, $\mathcal{Q}'$ has a relation $R'_{\{X,Y\}}$ that contains only the tuples in $R_{\{X,Y\}} \in \mathcal{Q}$ each of which uses a light value on X and the constrained heavy value on Y;
- For each edge $\{X, Y\}$ with two black vertices, $\mathcal{Q}'$ has a relation $R'_{\{X,Y\}}$ with only one tuple that takes the constrained heavy values on X and Y, respectively.

For example, in $R'_{\{\texttt{A},\texttt{B}\}}$, a tuple must use light values on both `A` and `B`; in $R'_{\{\texttt{D},\texttt{G}\}}$, a tuple must use value `d` on `D` and a light value on `G`; $R'_{\{\texttt{D},\texttt{K}\}}$ has only a single tuple with value `d` on `D` and `k` on `K`. Finding all result tuples for $\mathcal{Q}$ under the designated configuration amounts to evaluating the residue query $\mathcal{Q}'$.

Since the black attributes have had their values fixed in the configuration, they can be deleted from the residue query, after which some relations in $\mathcal{Q}'$ become unary or even disappear. Relation $R'_{\{\texttt{A},\texttt{D}\}} \in \mathcal{Q}'$, for example, is now regarded as a unary relation over $\{\texttt{A}\}$, with the understanding that every tuple is "piggybacked" the value `d` on `D`. Let us denote this unary relation as $R'_{\{\texttt{A}\}|\texttt{d}}$, which is illustrated in Figure 1c with a dotted edge extending from vertex `A` and carrying the label `d`. The deletion of `D`, `E`, `F`, and `K` results in 13 unary relations (e.g., two of them are over $\{\texttt{A}\}$, namely, $R'_{\{\texttt{A}\}|\texttt{d}}$ and $R'_{\{\texttt{A}\}|\texttt{e}}$). Attributes `G`, `H`, and `L` now become *isolated* because they are not connected to any other vertices by solid edges. Relations $R'_{\{\texttt{A},\texttt{B}\}}$, $R'_{\{\texttt{A},\texttt{C}\}}$, $R'_{\{\texttt{B},\texttt{C}\}}$, and $R'_{\{\texttt{I},\texttt{J}\}}$ still have arity 2 because their schemes do not have black attributes. $R'_{\{\texttt{D},\texttt{K}\}}$, on the other hand, has disappeared.

Our algorithm solves the residue query $\mathcal{Q}'$ of Figure 1c in two steps:

1. Perform a *semi-join reduction* which involves two substeps:
   - For every vertex X in Figure 1c, intersect all the unary relations over $\{X\}$ – if any – into a single list $R''_{\{X\}}$. For example, the two unary relations $R'_{\{\texttt{A}\}|\texttt{d}}$ and $R'_{\{\texttt{A}\}|\texttt{e}}$ of `A` are intersected on `A` to produce $R''_{\{\texttt{A}\}}$. Note that only the values in $R''_{\{\texttt{A}\}}$ can appear in the final join result.
   - For every non-isolated attribute X in Figure 1c, use $R''_{\{X\}}$ to shrink each non-unary relation $R'_{\{X,Y\}}$, for all relevant Y, to kick out those tuples whose X-values do not appear in $R''_{\{X\}}$. This reduces $R'_{\{X,Y\}}$ to a subset $R''_{\{X,Y\}}$. For example, after the shrinking, every tuple in $R''_{\{\texttt{A},\texttt{B}\}}$ uses a value in $R''_{\{\texttt{A}\}}$ on attribute `A`, and a value in $R''_{\{\texttt{B}\}}$ on attribute `B`.
2. Perform a cartesian product. To see how, first observe that the residue query $\mathcal{Q}'$ can now be further simplified into a join query $\mathcal{Q}''$ which includes:
   - The relation $R''_{\{X\}}$ for each isolated attribute X;
   - The relation $R''_{\{X,Y\}}$ for each solid edge in Figure 1c.

   Figure 1d gives a neater view of $\mathcal{Q}''$, from which it is easy to see that $Join(\mathcal{Q}'')$ equals the cartesian product of (i) three unary relations $R''_{\{\texttt{G}\}}$, $R''_{\{\texttt{H}\}}$, and $R''_{\{\texttt{L}\}}$, (ii) a binary relation

$R''_{\{\mathtt{I},\mathtt{J}\}}$, and (iii) the result of the "triangle join" $\{R''_{\{\mathtt{A},\mathtt{B}\}}, R''_{\{\mathtt{A},\mathtt{C}\}}, R''_{\{\mathtt{B},\mathtt{C}\}}\}$. This cartesian product can be generated in one round using the *hypercube algorithm* of [3], leveraging the fact that only light values are present in these relations. An optimal load can be achieved by setting $\lambda = \Theta(p^{1/(2\rho)})$.

The KS algorithm deploys a similar procedure to deal with a primitive type of configurations (see Lemma 14 of [9]). The main difference, however, is that while the KS algorithm resorts to more sophisticated and round-intensive procedures to tackle other configurations (e.g., the one in Figure 1b), we proceed in the same manner for all configurations anyway. This simplification is the side product of a theorem established in this paper, which shows that the cartesian product of all the *unary* relations $R''_{\{X\}}$ – one for each isolated attribute X (in our example, $R''_{\{\mathtt{G}\}}$, $R''_{\{\mathtt{H}\}}$, and $R''_{\{\mathtt{L}\}}$) – is not too large *on average*, over all the possible configurations. This property allows us to *duplicate* such cartesian products onto a large number of machines, which in turn is the key reason why the hypercube algorithm can be invoked to finish Step 2 in one round. In fact, handling those unary relations has been the main challenge in all the algorithms [7, 9, 10] applying the "decomposition by heavy-hitters" idea, because the binary relations obtained from the decomposition are so-called "skew-free", and hence, easy to process. In light of this, our theorem provides deeper insight into the reason why simple join queries can be processed with the optimal load.

It is worth mentioning that while our algorithm performs 3 rounds, the second round, which transmits certain statistics to assist machine allocation in the last round, incurs only a small load that is a polynomial of p and does not depend on m. In other words, the algorithm performs only 2 rounds whose loads are sensitive to m. This brings us very close to finally settling the problem with the optimal load using genuinely only 2 rounds, and leaves open the question: *is the transmission of those statistics absolutely necessary?*

2 Preliminaries

2.1 Hypergraphs

We define a *hypergraph* $\mathcal{G}$ as a pair $(\mathcal{V}, \mathcal{E})$ where:

- $\mathcal{V}$ is a finite set, where each element is called a *vertex*;
- $\mathcal{E}$ is a set of non-empty subsets of $\mathcal{V}$, where each subset is called a *hyperedge*.

Given a vertex $X \in \mathcal{V}$ and a hyperedge $e \in \mathcal{E}$, we say that X and e are *incident* to each other if $X \in e$. A vertex $X \in \mathcal{V}$ is *dangling* if it is not incident on any hyperedge in $\mathcal{E}$. In this paper, we consider only hypergraphs where there are no dangling vertices.

Two distinct vertices $X, Y \in V$ are *adjacent* to each other if there is an $e \in \mathcal{E}$ containing both X and Y. An edge e is *unary* if $|e| = 1$, or *binary* if $|e| = 2$. A *binary* hypergraph is one that has only binary edges. Given a subset $\mathcal{V}'$ of $\mathcal{V}$, we define the subgraph *induced* by $\mathcal{V}'$ as $(\mathcal{V}', \mathcal{E}')$ where $\mathcal{E}' = \{\mathcal{V}' \cap e \mid e \in \mathcal{E} \wedge \mathcal{V}' \cap e \neq \emptyset\}$.

2.2 Fractional Edge Coverings and Packings

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a hypergraph, and W a function mapping $\mathcal{E}$ to real values in $[0, 1]$. We call $W(e)$ the *weight* of the hyperedge e, and $\sum_{e\in\mathcal{E}} W(e)$ the *total weight* of W. Given a vertex $X \in \mathcal{V}$, we refer to $\sum_{e\in\mathcal{E}:X\in e} W(e)$, which is the sum of the weights of the edges incident to X, as the *weight* of X.

W is a *fractional edge covering* of $\mathcal{G}$ if the weight of every vertex $X \in \mathcal{V}$ is at least 1. The *fractional edge covering number* of $\mathcal{G}$, which is denoted as $\rho(\mathcal{G})$, equals the smallest total weight of all the fractional edge coverings. W is a *fractional edge packing* if the weight of

every vertex $X \in \mathcal{V}$ is at most 1. The *fractional edge packing number* of $\mathcal{G}$, which is denoted as $\tau(\mathcal{G})$, equals the largest total weight of all the fractional edge packings. A fractional edge packing W is *tight* if it is simultaneously also a fractional edge covering. Likewise, a fractional edge covering W is *tight* if it is simultaneously also a fractional edge packing. Note that in both cases it must hold that the weight of every vertex $X \in \mathcal{V}$ is exactly 1.

The lemma below lists several useful properties of binary hypergraphs:

▶ **Lemma 1.** *If $\mathcal{G}$ is binary, then:*

- *$\rho(\mathcal{G}) + \tau(\mathcal{G}) = |\mathcal{V}|$ and $\rho(\mathcal{G}) \geq \tau(\mathcal{G})$, where the two equalities hold if and only if $\mathcal{G}$ admits a tight fractional edge packing (or covering).*
- *$\mathcal{G}$ admits a fractional edge packing W of total weight $\tau(\mathcal{G})$ such that*
  1. *The weight of every vertex $X \in \mathcal{V}$ is either 0 or 1.*
  2. *If $\mathcal{Z}$ is the set of vertices in $\mathcal{V}$ with weight 0, then $\rho(\mathcal{G}) - \tau(\mathcal{G}) = |\mathcal{Z}|$.*

Proof. The first bullet is proved in Theorem 2.2.7 of [13]. The fractional edge packing W in Theorem 2.1.5 of [13] satisfies Condition 1 of the second bullet. This W also fulfills Condition 2, as is proved in Lemma 16 of [9]. ◀

Example. Suppose that $\mathcal{G}$ is the binary hypergraph in Figure 1a. It has a fractional edge covering number $\rho(\mathcal{G}) = 6.5$, as is achieved by the function W_1 that maps {G, F}, {D, K}, {I, J}, {E, H}, and {E, L} to 1, {A, B}, {A, C}, and {B, C} to 1/2, and the other edges to 0. Its fractional edge packing number is $\tau(\mathcal{G}) = 5.5$, achieved by the function W_2 which is the same as W_1 except that W_2 maps {E, L} to 0. W_2 also satisfies both conditions of the second bullet (notice that $\mathcal{Z} = \{\text{L}\}$). ◀

2.3 Hypergraph of a Join Query and the AGM Bound

Every join query $\mathcal{Q}$ defines a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \mathit{attset}(\mathcal{Q})$ and $\mathcal{E} = \{\mathit{scheme}(R) \mid R \in \mathcal{Q}\}$. When $\mathcal{Q}$ is scheme-clean, for each hyperedge $e \in \mathcal{E}$ we denote by R_e the input relation $R \in \mathcal{Q}$ with $e = \mathit{scheme}(R)$. Note also that $\mathcal{G}$ must be binary if $\mathcal{Q}$ is simple. The following result is known as the *AGM bound*:

▶ **Lemma 2** ([4]). *Let $\mathcal{Q}$ be a scheme-clean join query, and W be a fractional edge covering of the hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ defined by $\mathcal{Q}$. Then, $|\mathit{Join}(\mathcal{Q})| \leq \prod_{e \in \mathcal{E}} |R_e|^{W(e)}$.*

2.4 MPC Building Blocks

Cartesian Products. Suppose that R and S are relations with disjoint schemes. Their *cartesian product*, denoted as $R \times S$, is a relation over $\mathit{scheme}(R) \cup \mathit{scheme}(S)$ which consists of all the tuples $\boldsymbol{u}$ over $\mathit{scheme}(R) \cup \mathit{scheme}(S)$ such that $\boldsymbol{u}[\mathit{scheme}(R)] \in R$ and $\boldsymbol{u}[\mathit{scheme}(S)] \in S$. Sometimes, we need to compute the *cartesian product* of a set of relations $\mathcal{Q} = \{R_1, R_2, ..., R_t\}$ $(t \geq 2)$ with mutually disjoint schemes. For convenience, define $\mathit{CP}(\mathcal{Q})$ as a short form for $R_1 \times R_2 \times ... \times R_t$. Note that $\mathit{CP}(\mathcal{Q})$ can also be regarded as the result $\mathit{Join}(\mathcal{Q})$ of the join query $\mathcal{Q}$.

Two results regarding cartesian products will be useful:

▶ **Lemma 3** ([3]). *Given $\mathcal{Q} = \{R_1, R_2, ..., R_t\}$, we can compute $\mathit{CP}(\mathcal{Q})$ in one round with load $\tilde{O}(|\mathit{CP}(\mathcal{Q})|^{\frac{1}{t}}/p^{\frac{1}{t}})$ using p machines.*

▶ **Lemma 4** ([9]). *Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two join queries that have input sizes at most m, and satisfy the condition that $attset(\mathcal{Q}_1) \cap attset(\mathcal{Q}_2) = \emptyset$. Suppose that $Join(\mathcal{Q}_1)$ can be computed in one round with load $\tilde{O}(m/p_1^{1/t_1})$ using p_1 machines, and similarly, $Join(\mathcal{Q}_1)$ can be computed in one round with load $\tilde{O}(m/p_2^{1/t_2})$ using p_2 machines. Then $Join(\mathcal{Q}_1) \times Join(\mathcal{Q}_2)$ can be computed in one round with load $\tilde{O}(m/\min\{p^{1/t_1}, p^{1/t_2}\})$ using $p_1 p_2$ machines.*

Skew-Free Queries. Let $\mathcal{Q}$ be a join query on binary relations. Regardless of whether $\mathcal{Q}$ is simple, it can be solved in a single round with a small load if no value appears too often in its input relations. Denote by m the input size of $\mathcal{Q}$. Set $k = |attset(\mathcal{Q})|$, and list out the attributes in $attset(\mathcal{Q})$ as $X_1, ..., X_k$. Let p_i be a positive integer, $i \in [1, k]$, which is referred to as the *share* of X_i. A relation $R \in \mathcal{Q}$ with scheme $\{X_i, X_j\}$ is *skew-free* if every value $x \in \mathbf{dom}$ fulfills both conditions below:

- R has $O(m/p_i)$ tuples $\boldsymbol{u}$ with $\boldsymbol{u}(X_i) = x$;
- R has $O(m/p_j)$ tuples $\boldsymbol{u}$ with $\boldsymbol{u}(X_j) = x$.

Define $share(R) = p_i \cdot p_j$. If every $R \in \mathcal{Q}$ is skew-free, $\mathcal{Q}$ is *skew-free*, and can be solved with the following guarantee:

▶ **Lemma 5** ([5]). *A skew-free query $\mathcal{Q}$ can be answered in one round with load $\tilde{O}(m/\min_{R\in\mathcal{Q}} share(R))$ using $\prod_{i=1}^{k} p_i$ machines.*

One-Attribute Reduction. Let $X \in \mathbf{att}$ be an attribute. We have $a \geq 1$ unary relations $R_1, ..., R_a$ over $\{X\}$, and $b \geq 1$ binary relations $S_1, ..., S_b$ such that S_i $(1 \leq i \leq b)$ is a relation over $\{X, Y_i\}$ where Y_i is an attribute in $\mathbf{att}$ different from X. Here, both a and b are constants. Our objective is to compute $S_i^{\#}$ which includes all tuples $\boldsymbol{u} \in S_i$ satisfying the condition that $\boldsymbol{u}(X) \in \bigcap_{j=1}^{a} R_j$. We will refer to this operation as *one-attribute reduction*. Let $n = \sum_{j=1}^{a} |R_i| + \sum_{i=1}^{b} |S_i|$. A value $x \in \mathbf{dom}$ is a *heavy-hitter* if at least n/p tuples in *some* S_i $(1 \leq i \leq b)$ use x as their X-values, where p is the number of machines assigned to the operation.

▶ **Lemma 6.** *One-attribute reduction can be performed in one round with load $\tilde{O}(p + n/p)$ using p machines, provided that each machine knows all the heavy-hitters.*

Proof. See Appendix A. ◀

It is worth mentioning that the above lemma is an extension of a result in [10]. The p term in the load can actually be eliminated, if the machine knows also additional statistics of the heavy-hitters. We do not need to be bothered with such details because the term is affordable for our purposes.

3 A Taxonomy of the Join Result

Recall that Section 1 outlined a method to partition the join result based on heavy and light values. In this section, we formalize this method and establish some fundamental properties. Denote by $\mathcal{Q}$ a simple join query, by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ the hypergraph defined by $\mathcal{Q}$, by m the input size of $\mathcal{Q}$, and by k the number of attributes in $attset(\mathcal{Q})$.

Heavy and Light Values. Fix an arbitrary integer $\lambda \in [1, m]$. A value $x \in \mathbf{dom}$ is

- *heavy* if there exists a relation $R \in \mathcal{Q}$ and some attribute $X \in \mathit{scheme}(R)$ such that $|\{\boldsymbol{u} \in R \mid \boldsymbol{u}(X) = x\}| \geq m/\lambda$;
- *light* if x is not heavy, but appears in at least one relation $R \in \mathcal{Q}$.

Since each relation has $O(1)$ attributes, the number of heavy values is $O(\lambda)$.

Configurations. Let $\mathcal{H}$ be an arbitrary (possibly empty) subset of $\mathit{attset}(\mathcal{Q})$. A *configuration* of $\mathcal{H}$ is a tuple $\boldsymbol{\eta}$ over $\mathcal{H}$ such that $\boldsymbol{\eta}(X)$ is heavy for every $X \in \mathcal{H}$. Obviously, each $\mathcal{H} \subseteq \mathit{attset}(\mathcal{Q})$ has at most $O(\lambda^{|\mathcal{H}|})$ configurations.

Residue Relations and Residue Queries. Now, let us fix a configuration $\boldsymbol{\eta}$ of $\mathcal{H}$, and aim to produce all the result tuples $\boldsymbol{u} \in \mathit{Join}(\mathcal{Q})$ *consistent* with the configuration, namely, $\boldsymbol{u}$ satisfies

- $\boldsymbol{u}(X) = \boldsymbol{\eta}(X)$ for every $X \in \mathcal{H}$, and
- $\boldsymbol{u}(X)$ is light for every $X \in \mathit{attset}(\mathcal{Q}) \setminus \mathcal{H}$.

We will take a few steps to define what is the *residue query* under $\boldsymbol{\eta}$, which is denoted as $\mathcal{Q}'_{\boldsymbol{\eta}}$, whose result is precisely the set of all the qualifying $\boldsymbol{u}$.

Let e be a hyperedge in $\mathcal{E}$ that is not subsumed by $\mathcal{H}$, i.e., e has at least one attribute outside $\mathcal{H}$. This hyperedge is said to be *active* on $\boldsymbol{\eta}$. Define $e' = e \setminus \mathcal{H}$, namely, the set of attributes in e that are outside $\mathcal{H}$. The relation $R_e \in \mathcal{Q}$ defines a *residue relation* under $\boldsymbol{\eta}$ which

- is over e' and
- consists of every tuple $\boldsymbol{v}$ that is the projection of some tuple $\boldsymbol{w} \in R_e$ "consistent" with $\boldsymbol{\eta}$, namely: (i) $\boldsymbol{w}(X) = \boldsymbol{\eta}(X)$ for every $X \in e \cap \mathcal{H}$, (ii) $\boldsymbol{w}(Y)$ is light for every $Y \in e'$, and (iii) $\boldsymbol{v} = \boldsymbol{w}[e']$.

The residue relation is denoted as $R'_{e'|\boldsymbol{\eta}[e \setminus e']}$, where $\boldsymbol{\eta}[e \setminus e']$ is the projection of $\boldsymbol{\eta}$ on $e \setminus e'$, as was introduced in Section 1.1.

We can now define the *residue query* as

$$\mathcal{Q}'_{\boldsymbol{\eta}} = \left\{ R_{e'|\boldsymbol{\eta}[e \setminus e']} \;\middle|\; e \in \mathcal{E},\ e \text{ active on } \boldsymbol{\eta} \right\}.$$

Example. Suppose that $\mathcal{Q}$ is the query discussed in Section 1.3 with its hypergraph $\mathcal{G}$ given in Figure 1a. Consider the configuration $\boldsymbol{\eta}$ of $\mathcal{H} = \{\mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{K}\}$ where $\boldsymbol{\eta}[\mathtt{D}] = \mathtt{d}$, $\boldsymbol{\eta}[\mathtt{E}] = \mathtt{e}$, $\boldsymbol{\eta}[\mathtt{F}] = \mathtt{f}$, and $\boldsymbol{\eta}[\mathtt{K}] = \mathtt{k}$. If e is the edge $\{\mathtt{A}, \mathtt{D}\}$, then $e' = \{\mathtt{A}\}$ and $\boldsymbol{\eta}[e \setminus e'] = \boldsymbol{\eta}[\{\mathtt{D}\}] = \mathtt{d}$, such that $R'_{e'|\boldsymbol{\eta}[e \setminus e']}$ is the relation $R'_{\{\mathtt{A}\}|\mathtt{d}}$ mentioned in Section 1.3. If e is the edge $\{\mathtt{A}, \mathtt{B}\}$, on the other hand, then $e' = \{\mathtt{A}, \mathtt{B}\}$ and $\boldsymbol{\eta}[e \setminus e'] = \emptyset$, so that $R'_{e'|\boldsymbol{\eta}[e \setminus e']}$ can be written as $R'_{\{\mathtt{A},\mathtt{B}\}|\emptyset}$, and is the relation $R'_{\{\mathtt{A},\mathtt{B}\}}$ in Section 1.3. The residue query $\mathcal{Q}'_{\boldsymbol{\eta}}$ is precisely the query $\mathcal{Q}'$ described in Section 1.3. *(to be continued)*▲

It is rudimentary to verify

$$\mathit{Join}(\mathcal{Q}) = \bigcup_{\mathcal{H}} \Big(\bigcup_{\text{config. } \boldsymbol{\eta} \text{ of } \mathcal{H}} \mathit{Join}(\mathcal{Q}'_{\boldsymbol{\eta}}) \times \{\boldsymbol{\eta}\} \Big). \tag{1}$$

Denote by $m_{\boldsymbol{\eta}}$ the input size of $\mathcal{Q}'_{\boldsymbol{\eta}}$. The next proposition says that total input size of all the residue queries is not too large:

▶ **Proposition 7.** *If $\mathcal{Q}$ is simple, $\sum_{\text{config. } \boldsymbol{\eta} \text{ of } \mathcal{H}} m_{\boldsymbol{\eta}} = O(m \cdot \lambda^{k-2})$ holds for every $\mathcal{H} \subseteq \mathit{attset}(\mathcal{Q})$.*

Proof. See Appendix B. ◀

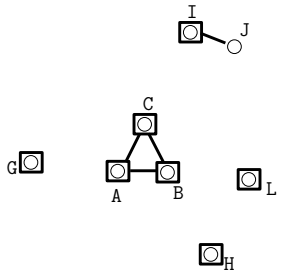


Figure 2 Subgraph induced by $\mathcal{L}$.

4 A Join Computation Framework

Given a simple join query $\mathcal{Q}$, we will concentrate on an arbitrary subset $\mathcal{H}$ of $attset(\mathcal{Q})$ in this section. In Sections 4.1-4.2, we will generalize the strategy illustrated in Section 1.3 into a formal framework for producing

$$\bigcup_{\text{config. } \boldsymbol{\eta} \text{ of } \mathcal{H}} Join(\mathcal{Q}'_{\boldsymbol{\eta}}). \tag{2}$$

Section 4.3 will then establish a theorem on this framework, which is the core of the techniques proposed in this paper.

4.1 Removing the Attributes in $\mathcal{H}$

We will refer to each attribute in $\mathcal{H}$ as a *heavy attribute*. Define $\mathcal{L} = scheme(\mathcal{Q}) \setminus \mathcal{H}$, where each attribute is called a *light attribute*. Denote by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ the hypergraph defined by $\mathcal{Q}$. An edge $e \in \mathcal{E}$ is (i) a *light edge* if e contains two light attributes, or (ii) a *cross edge* if e contains a heavy attribute and a light attribute. A light attribute $X \in \mathcal{L}$ is a *border attribute* if it appears in at least one cross edge e of $\mathcal{G}$; note that this implies $e \setminus \mathcal{H} = \{X\}$. Denote by $\mathcal{G}' = (\mathcal{L}, \mathcal{E}')$ the subgraph of $\mathcal{G}$ induced by $\mathcal{L}$. A vertex $X \in \mathcal{L}$ is *isolated* if $\{X\}$ is the only edge in $\mathcal{E}'$ incident to X. Define $\mathcal{I}$ to be the set of isolated vertices in $\mathcal{G}'$.

Example (cont.). As before, let $\mathcal{Q}$ be the join query whose hypergraph $\mathcal{G}$ is shown in Figure 1a, and set $\mathcal{H} = \{\mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{K}\}$. $\mathcal{L}$ includes all the white vertices in Figure 1b. $\{\mathtt{A}, \mathtt{B}\}$ is a light edge, $\{\mathtt{A}, \mathtt{D}\}$ is a cross edge, while $\{\mathtt{D}, \mathtt{K}\}$ is neither a light edge nor a cross edge. All the vertices in $\mathcal{L}$ except $\mathtt{J}$ are border vertices. Figure 2 shows the subgraph of $\mathcal{G}$ induced by $\mathcal{L}$, where a unary edge is represented by a box and a binary edge by a segment. Notice that no unary edge covers $\mathtt{J}$. Vertices $\mathtt{G}$, $\mathtt{H}$, and $\mathtt{L}$ are the only isolated vertices. *(to be continued)*▲

4.2 Semi-Join Reduction

Recall that every configuration $\boldsymbol{\eta}$ of $\mathcal{H}$ gives rise to a residue query $\mathcal{Q}'_{\boldsymbol{\eta}}$. Next, we will transform $\mathcal{Q}'_{\boldsymbol{\eta}}$ into an alternative join query $\mathcal{Q}''_{\boldsymbol{\eta}}$ which, as shown in the next section, can be processed in a single round in the MPC model.

First of all, observe that the hypergraph defined by $\mathcal{Q}'_{\boldsymbol{\eta}}$ is always $\mathcal{G}' = (\mathcal{L}, \mathcal{E}')$, regardless of $\boldsymbol{\eta}$. Consider a border attribute $X \in \mathcal{L}$, and a cross edge e of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ incident to X. As explained in Section 3, the input relation $R_e \in \mathcal{Q}$ defines a unary residue relation $R'_{e'|\boldsymbol{\eta}[e \setminus e']} \in \mathcal{Q}'_{\boldsymbol{\eta}}$ where $e' = e \setminus \mathcal{H}$. Since $e' = \{X\}$, we can as well write the relation as $R'_{\{X\}|\boldsymbol{\eta}[e \setminus \{X\}]}$. Now that every such relation has the same scheme $\{X\}$, we can define:

$$R''_{\{X\}|\boldsymbol{\eta}} = \bigcap_{\text{cross edge } e \text{ containing } X} R'_{\{X\}|\boldsymbol{\eta}[e \setminus \{X\}]}. \tag{3}$$

Example (cont.). Let $\mathcal{H}$ and $\boldsymbol{\eta}$ be the same as in the earlier description of this example. Set X to the border attribute A. If e is the cross edge $\{\mathtt{A}, \mathtt{D}\}$, the example in Section 3 has shown that $R'_{e'|\boldsymbol{\eta}[e\setminus e']}$ is the relation $R'_{\{\mathtt{A}\}|\mathtt{d}}$ obtained in Section 1.3. Similarly, if e is the cross edge $\{\mathtt{A}, \mathtt{E}\}$, $R'_{e'|\boldsymbol{\eta}[e\setminus e']}$ is the relation $R'_{\{\mathtt{A}\}|\mathtt{e}}$ obtained in Section 1.3. As A is contained only in these two cross edges, $R''_{\{A\}|\boldsymbol{\eta}}$ is the intersection of $R'_{\{\mathtt{A}\}|\mathtt{d}}$ and $R'_{\{\mathtt{A}\}|\mathtt{e}}$, and corresponds to the relation $R''_{\{A\}}$ given in Section 1.3. *(to be continued)*▲

Consider a light edge $e = \{X, Y\}$ in $\mathcal{G}$. Recall that R_e defines a residue relation $R'_{e'|\boldsymbol{\eta}[e\setminus e']} \in \mathcal{Q}'_{\boldsymbol{\eta}}$, which can be written as $R'_{e|\emptyset}$ because $e' = e \setminus \mathcal{H} = e$. We define $R''_{e|\boldsymbol{\eta}}$ as a relation over e which consists of every tuple $\boldsymbol{u} \in R'_{e|\emptyset}$ satisfying both conditions below:

- (applicable only if X is a border attribute) $\boldsymbol{u}(X) \in R''_{X|\boldsymbol{\eta}}$;
- (applicable only if Y is a border attribute) $\boldsymbol{u}(Y) \in R''_{Y|\boldsymbol{\eta}}$.

Note that if neither X nor Y is a border attribute, then $R''_{e|\boldsymbol{\eta}} = R'_{e|\emptyset}$.

Example (cont.). Let us concentrate the light edge $e = \{\mathtt{A}, \mathtt{B}\}$. The example in Section 3 has explained that $R'_{e'|\boldsymbol{\eta}[e\setminus e']} = R'_{\{\mathtt{A},\mathtt{B}\}|\emptyset}$ is the relation $R'_{\{\mathtt{A},\mathtt{B}\}}$ obtained in Section 1.3. As A and B are both border attributes, $R''_{\{\mathtt{A},\mathtt{B}\}|\boldsymbol{\eta}}$ includes all the tuples in $R'_{\{\mathtt{A},\mathtt{B}\}}$ that take a value in $R''_{\{\mathtt{A}\}|\boldsymbol{\eta}}$ on attribute A and a value in $R''_{\{\mathtt{B}\}|\boldsymbol{\eta}}$ on attribute B. Note that $R''_{\{\mathtt{A},\mathtt{B}\}|\boldsymbol{\eta}}$ corresponds to the relation $R''_{\{\mathtt{A},\mathtt{B}\}}$ given in Section 1.3. *(to be continued)*▲

Every vertex $X \in \mathcal{I}$ must be a border attribute, and thus must have $R''_{X|\boldsymbol{\eta}}$ defined. Therefore, we can legally define:

$$\mathcal{Q}''_{light|\boldsymbol{\eta}} = \{R''_{e|\boldsymbol{\eta}} \mid \text{light edge } e \in \mathcal{E}\}$$
$$\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}} = \{R''_{\{X\}|\boldsymbol{\eta}} \mid X \in \mathcal{I}\}$$
$$\mathcal{Q}''_{\boldsymbol{\eta}} = \mathcal{Q}''_{light|\boldsymbol{\eta}} \cup \mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}}.$$

Notice that the join queries $\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}}$, $\mathcal{Q}''_{light|\boldsymbol{\eta}}$, and $\mathcal{Q}''_{\boldsymbol{\eta}}$ are all scheme-clean.

Example (cont.). $\mathcal{Q}''_{light|\boldsymbol{\eta}}$ consists of $R''_{\{\mathtt{A},\mathtt{B}\}}$, $R''_{\{\mathtt{A},\mathtt{C}\}}$, $R_{\{\mathtt{B},\mathtt{C}\}}$, and $R_{\{\mathtt{I},\mathtt{J}\}}$, and $\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}}$ consists of $R''_{\{\mathtt{G}\}}$, $R''_{\{\mathtt{H}\}}$, and $R''_{\{\mathtt{L}\}}$, where all the relation names follow those given in Section 1.3. ◀

▶ **Proposition 8.** $Join(\mathcal{Q}'_{\boldsymbol{\eta}}) = Join(\mathcal{Q}''_{\boldsymbol{\eta}}) = CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}}) \times Join(\mathcal{Q}''_{light|\boldsymbol{\eta}})$.

Proof. See Appendix C. ◀

We will refer to the above process of converting $\mathcal{Q}'_{\boldsymbol{\eta}}$ to $\mathcal{Q}''_{\boldsymbol{\eta}}$ as *semi-join reduction*, and call $\mathcal{Q}''_{\boldsymbol{\eta}}$ the *reduced query* of $\boldsymbol{\eta}$.

4.3 The Isolated Cartesian Product Theorem

We are ready to present the first main result of this paper:

▶ **Theorem 9** (The Isolated Cartesian Product Theorem)**.**

$$\sum_{\text{config. } \boldsymbol{\eta} \text{ of } \mathcal{H}} \left|CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}})\right| \quad = \quad O\Big(\lambda^{2(\rho-|\mathcal{I}|)-|\mathcal{L}\setminus\mathcal{I}|} \cdot m^{|\mathcal{I}|}\Big) \tag{4}$$

where ρ is the fractional edge covering number of $\mathcal{Q}$.

The rest of the section serves as a proof of the above theorem. To start with, define $\mathcal{F}$ to be the set of attributes in $\mathcal{H}$ that are adjacent to at least one isolated vertex in $\mathcal{G}$. The left hand side of (4) can be bounded by looking at only the configurations of $\mathcal{F}$:

▶ **Lemma 10.** $\sum_{\text{config. } \boldsymbol{\eta} \text{ of } \mathcal{H}} \left|CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}})\right| = O\left(\lambda^{|\mathcal{H}|-|\mathcal{F}|}\right) \cdot \sum_{\text{config. } \boldsymbol{\eta}' \text{ of } \mathcal{F}} \left|CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}'})\right|$.

Proof. See Appendix D. ◀

Now, let us take a fractional edge packing W of the hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that obeys the second bullet of Lemma 1. Denote by τ the total weight of W which, by definition of W, is the fractional edge packing number of $\mathcal{G}$. Define:

$$\mathcal{Z} = \left\{X \in \mathcal{V} \mid \sum_{e \in \mathcal{E}: X \in e} W(e) = 0\right\}$$

that is, Z is the set of vertices with weight 0 under W. Set $\mathcal{I}_0 = \mathcal{I} \cap \mathcal{Z}$ and $\mathcal{I}_1 = \mathcal{I} \setminus \mathcal{I}_0$. Since W satisfies Condition 1 of the second bullet in Lemma 1, we know that every vertex in $\mathcal{I}_1$ has weight 1, while every vertex in $\mathcal{I}_0$ has weight 0.

Example. Let $\mathcal{G}$ be the hypergraph in Figure 1a. As explained by the example in Section 2.2, the fractional edge packing number τ of $\mathcal{G}$ is achieved by the function W that maps {G, F}, {D, K}, {I, J}, and {E, H} to 1, {A, B}, {A, C}, and {B, C} to 1/2, and the other edges to 0; $\mathcal{Z}$ contains a single vertex L. Setting $\mathcal{H} = \{\mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{K}\}$ yields $\mathcal{I} = \{\mathtt{G}, \mathtt{H}, \mathtt{L}\}$, $\mathcal{I}_0 = \{\mathtt{L}\}$, and $\mathcal{I}_1 = \{\mathtt{G}, \mathtt{H}\}$. $\mathcal{F} = \{\mathtt{D}, \mathtt{E}, \mathtt{F}\}$, noticing that K is not adjacent to any isolated vertex. *(to be continued)*▲

We now present a crucial lemma which is in fact a stronger version of Theorem 9:

▶ **Lemma 11.** $\sum_{\text{config. } \boldsymbol{\eta}' \text{ of } \mathcal{F}} \left|CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}'})\right| = O\left(\lambda^{|\mathcal{F}|-|\mathcal{I}_1|} \cdot m^{|\mathcal{I}|}\right)$.

Before proving the above lemma, let us first see how it can be used to complete the proof of Theorem 9. By combining Lemmas 10 and 11, we know that the left hand side of (4) is $O(\lambda^{|\mathcal{H}|-|\mathcal{I}_1|} \cdot m^{|\mathcal{I}|})$. Hence, it suffices to prove

$$\begin{aligned}
|\mathcal{H}| - |\mathcal{I}_1| &\leq 2(\rho - |\mathcal{I}|) - |\mathcal{L} \setminus \mathcal{I}| \Leftrightarrow \\
|\mathcal{H}| + |\mathcal{L} \setminus \mathcal{I}| + |\mathcal{I}| - |\mathcal{I}_1| &\leq 2\rho \Leftrightarrow \\
|\mathcal{V}| - \rho + |\mathcal{I}_0| &\leq \rho \text{ (note: } |\mathcal{V}| = |\mathcal{H}| + |\mathcal{L} \setminus \mathcal{I}| + |\mathcal{I}|) \Leftrightarrow \\
\tau + |\mathcal{I}_0| &\leq \rho \text{ (note: } \rho + \tau = |\mathcal{V}| \text{ by Lemma 1)}
\end{aligned}$$

which is true because $\rho - \tau = |\mathcal{Z}|$ by Condition 2 of the second bullet in Lemma 1, and $\mathcal{I}_0 \subseteq \mathcal{Z}$.

Proof of Lemma 11. Our idea is to construct a set $\mathcal{Q}^*$ of relations such that $Join(\mathcal{Q}^*)$ has a result size at least the left hand side of the inequality in Lemma 11. Then, we will prove that the hypergraph of $\mathcal{Q}^*$ has a certain fractional edge covering which, together with the AGM bound, yields an upper bound on $|Join(\mathcal{Q}^*)|$ which happens to be the right hand side of the inequality.

We construct $\mathcal{Q}^*$ as follows. Initially, set $\mathcal{Q}^*$ to $\emptyset$. For every cross edge $e \in \mathcal{E}$ incident to an isolated vertex, add to $\mathcal{Q}^*$ a relation $R^*_e = R_e$. For every $X \in \mathcal{F}$, add a unary relation $R^*_{\{X\}}$ to $\mathcal{Q}^*$ which consists of all the heavy values on attribute X. Note that $R^*_{\{X\}}$ has $O(\lambda)$ tuples. Finally, for every $Y \in \mathcal{I}_0$, add a unary relation $R^*_{\{Y\}}$ to $\mathcal{Q}^*$ which contains all the heavy and light values on attribute Y.

Define $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$ as the hypergraph defined by $\mathcal{Q}^*$. Note that $\mathcal{V}^* = \mathcal{I} \cup \mathcal{F}$, while $\mathcal{E}^*$ consists of all the cross edges in $\mathcal{G}$, $|\mathcal{F}|$ unary edges $\{X\}$ for every $X \in \mathcal{F}$, and $|\mathcal{I}_0|$ unary edges $\{Y\}$ for every $Y \in \mathcal{I}_0$.

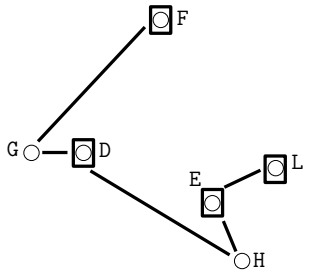


Figure 3 Illustration of $\mathcal{Q}^*$.

Example (cont.). Figure 3 shows the hypergraph of the $\mathcal{Q}^*$ constructed, where as before a box and a segment represent a unary and a binary edge, respectively. *(to be continued)*▲

▶ **Lemma 12.** $\sum_{config.\ \boldsymbol{\eta}'\ of\ \mathcal{F}} \left| CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}'}) \right| \leq |Join(\mathcal{Q}^*)|$.

Proof. See Appendix E. ◀

▶ **Lemma 13.** *$\mathcal{G}^*$ admits a tight fractional edge covering $\mathcal{W}^*$ satisfying $\sum_{X\in F} W^*(\{X\}) = |\mathcal{F}| - |\mathcal{I}_1|$.*

Proof. Recall that our proof of Theorem 9 began with a fractional edge packing W of $\mathcal{G}$. We construct a desired function W^* from W as follows. First, for every cross edge $e \in \mathcal{E}$, set $W^*(e) = W(e)$. Observe that every edge in $\mathcal{E}$ incident to $Y \in \mathcal{I}$ must be a cross edge. Hence, $\sum_{\text{binary } e\in\mathcal{E}^*:Y\in e} W^*(e)$ is precisely the weight of Y under W. By definition of W, we thus have ensured $\sum_{\text{binary } e\in\mathcal{E}^*:Y\in e} W^*(e) = 1$ for each $Y \in \mathcal{I}_1$, and $\sum_{\text{binary } e\in\mathcal{E}^*:Y\in e} W^*(e) = 0$ for each $Y \in \mathcal{I}_0$. As a second step, we set $W^*(\{Y\}) = 1$ for each $Y \in \mathcal{I}_0$ so that the edges in $\mathcal{E}^*$ containing Y have a total weight of 1.

It remains to make sure that each attribute $X \in \mathcal{F}$ has a weight 1 under W^*. Since W is a fractional edge packing of $\mathcal{G}$, it must hold that $\sum_{\text{binary } e\in\mathcal{E}^*:X\in e} W(e) \leq 1$. This permits us to assign the following weight to the unary edge $\{X\}$:

$$W^*(\{X\}) \quad = \quad 1 - \sum_{\text{binary } e\in\mathcal{E}^*:X\in e} W(e).$$

This finishes the design of W^* which is now a tight fractional edge covering of $\mathcal{G}^*$. Clearly:

$$\sum_{X\in\mathcal{F}} W^*(\{X\}) \quad = \quad |\mathcal{F}| - \sum_{X\in\mathcal{F}} \sum_{\text{binary } e\in\mathcal{E}^*:X\in e} W(e). \tag{5}$$

Every binary edge $e \in \mathcal{E}^*$ contains a vertex in $\mathcal{F}$ and a vertex in $\mathcal{I}$. Therefore:

$$\sum_{X\in\mathcal{F}} \sum_{\text{binary } e\in\mathcal{E}^*:X\in e} W(e) \quad = \quad \sum_{Y\in\mathcal{I}} \sum_{\text{binary } e\in\mathcal{E}^*:Y\in e} W(e) = |\mathcal{I}_1|.$$

Putting together the above equation with (5) completes the proof. ◀

Example (cont.). For the $\mathcal{G}^*$ in Figure 3, a fractional edge covering in Lemma 13 is given by the function W^* that maps {G, F}, {E, H}, {D}, and {L} to 1, and the other edges to 0. Note that $\sum_{X\in F} W^*(\{X\}) = W^*(\{\mathtt{D}\}) = 1$, same as $|\mathcal{F}| - |\mathcal{I}_1| = 3 - 2 = 1$. ◀

The AGM bound in Lemma 2 tells us that

$$\begin{aligned} \mathit{Join}(\mathcal{Q}^*) &\leq \prod_{e\in\mathcal{E}^*} |R^*_e|^{W^*(e)} = \Big(\prod_{X\in\mathcal{F}} |R^*_{\{X\}}|^{W^*(\{X\})}\Big)\Big(\prod_{Y\in\mathcal{I}} \prod_{e\in\mathcal{E}^*:Y\in e} |R^*_e|^{W^*(e)}\Big) \\ &= \Big(\prod_{X\in\mathcal{F}} (O(\lambda))^{W^*(\{X\})}\Big)\Big(\prod_{Y\in\mathcal{I}} \prod_{e\in\mathcal{E}^*:Y\in e} m^{W^*(e)}\Big) \\ \text{(by Lemma 13)} &= O(\lambda^{|\mathcal{F}|-|\mathcal{I}_1|}) \cdot m^{|\mathcal{I}|} \end{aligned}$$

which completes the proof of Lemma 11.

5 A 5-Round MPC Algorithm

We now proceed to implement the strategy discussed in the previous section under the MPC model. Our objective in this section is to explain how the isolated cartesian product theorem can be utilized to answer a simple join query $\mathcal{Q}$ with the optimal load $O(m/p^{1/\rho})$ in a rather straightforward manner. Hence, we intentionally leave out the optimization tricks to reduce the number of rounds, but even so, our algorithm finishes in only 5 rounds. Those tricks are the topic of the next section.

A *statistical record* is defined as a tuple (R, X, x, cnt), where R is a relation in $\mathcal{Q}$, X an attribute in $scheme(R)$, x a value in **dom**, and cnt the number of tuples $\boldsymbol{u} \in R$ with $\boldsymbol{u}(X) = x$. Specially, $(R, \emptyset, nil, cnt)$ is also regarded as a statistical record where cnt gives the number of tuples in R that use only light values. A *histogram* is defined as the set of statistical records for all possible R, X, and x satisfying (i) $cnt = \Omega(m/p^{1/\rho})$, or (ii) $X = \emptyset$ (and, hence $x = nil$); note that there are only $O(p^{1/\rho})$ such records. We assume that every machine has a local copy of the histogram. It is worth mentioning that all existing join algorithms [5,9], which strive to finish in a *specifically* small – rather than just asymptotically constant – number of rounds, demand that each machine should be preloaded with $p^{O(1)}$ statistical records.

Henceforth, the value of λ will be fixed to $\Theta(p^{1/(2\rho)})$. We focus on explaining how to compute (2) for an arbitrary subset $\mathcal{H}$ of $attset(\mathcal{Q})$. Set $k = |attset(\mathcal{Q})|$. As $attset(\mathcal{Q})$ has $2^k = O(1)$ subsets, processing all of them in parallel increases the load only by a constant factor, and definitely discovers the entire $\mathit{Join}(\mathcal{Q})$, as is guaranteed by (1). Our algorithm produces (2) in three steps:

1. Generate the input relations of the residue query $\mathcal{Q}'_{\boldsymbol{\eta}}$ of every configuration $\boldsymbol{\eta}$ of $\mathcal{H}$.
2. Generate the input relations of the reduced query $\mathcal{Q}''_{\boldsymbol{\eta}}$ of every $\boldsymbol{\eta}$.
3. Evaluate $\mathcal{Q}''_{\boldsymbol{\eta}}$ for every $\boldsymbol{\eta}$.

The number of configurations of $\mathcal{H}$ is $O(\lambda^{|\mathcal{H}|}) = O(\lambda^k) = O(p^{k/(2\rho)})$, which is $O(p)$ because $\rho \geq k/2$ by the first bullet of Lemma 1. Next, we elaborate on the details of each step.

Step 1. Proposition 7 tells us that the input relations of all the residue queries have $O(m \cdot \lambda^{k-2})$ tuples in total. We allocate $p'_{\boldsymbol{\eta}} = \lceil p \cdot \frac{m_{\boldsymbol{\eta}}}{\Theta(m\cdot\lambda^{k-2})} \rceil$ machines to store the relations of $\mathcal{Q}'_{\boldsymbol{\eta}}$, so that each machine assigned to $Q'_{\boldsymbol{\eta}}$ keeps on average $O(m_{\boldsymbol{\eta}}/p'_{\boldsymbol{\eta}}) = O(m \cdot \lambda^{k-2}/p) = O(m/p^{1/\rho})$ tuples, where the last equality used $\rho \geq k/2$. Since each machine $i \in [1, p]$ can use the histogram to calculate $m_{\boldsymbol{\eta}}$ precisely for each $\boldsymbol{\eta}$, it can compute locally the id range of the $m_{\boldsymbol{\eta}}$ machines responsible for $Q'_{\boldsymbol{\eta}}$. If a tuple $\boldsymbol{u}$ in the local storage of machine i belongs

to $Q'_{\boldsymbol{\eta}}$, the machine sends $\boldsymbol{u}$ to a random machine within that id range. Standard analysis shows that each of the $m_{\boldsymbol{\eta}}$ machines receives roughly the same number of tuples, such that this step can be done in a single round with load $\tilde{O}(m/p^{1/\rho})$.

Step 2. Now that all the input relations of each $\mathcal{Q}'_{\boldsymbol{\eta}}$ have been stored on $p'_{\boldsymbol{\eta}}$ machines, the semi-join reduction that converts $\mathcal{Q}'_{\boldsymbol{\eta}}$ to $\mathcal{Q}''_{\boldsymbol{\eta}}$ becomes a standard process [9] that can be accomplished in 2 rounds with load $\tilde{O}(m_{\boldsymbol{\eta}}/p'_{\boldsymbol{\eta}}) = \tilde{O}(m/p^{1/\rho})$.

Step 3. This step starts by letting each machine know about the value of $|CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}})|$ for every $\boldsymbol{\eta}$. For this purpose, each machine can broadcast to all machines how many tuples it has in $R''_{\{X\}|\boldsymbol{\eta}}$ for every $X \in \mathcal{I}$ and every $\boldsymbol{\eta}$. Since there are $O(p)$ different $\boldsymbol{\eta}$, at most $O(p)$ numbers are broadcast by each machine, such that the load of this round is $O(p^2)$. With all these numbers, each machine can figure out independently the values of all $|CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}})|$. We will call this round the *statistical round* henceforth.

We allocate

$$p''_{\boldsymbol{\eta}} = \left\lceil p \cdot \frac{|CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}})|}{\Theta(\lambda^{2(\rho-|\mathcal{I}|)-|\mathcal{L}\setminus\mathcal{I}|} \cdot m^{|\mathcal{I}|})} \right\rceil \tag{6}$$

machines to computing $\mathcal{Q}''_{\boldsymbol{\eta}}$. Theorem 9 guarantees that the total number of machines needed by all the configurations is at most p. We complete the algorithm with the lemma below:

▶ **Lemma 14.** *$\mathcal{Q}''_{\boldsymbol{\eta}}$ can be answered in one round with load $O(m/p^{1/\rho})$ using $p''_{\boldsymbol{\eta}}$ machines.*

Proof. $Join(\mathcal{Q}''_{\boldsymbol{\eta}})$ is the cartesian product of $CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}})$ and $Join(\mathcal{Q}''_{light|\boldsymbol{\eta}})$, as shown Proposition 8. By Lemma 3, if we deploy $p''_{\boldsymbol{\eta}}/\lambda^{\mathcal{L}\setminus\mathcal{I}}$ machines to compute $CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}})$ in one round, the load is

$$\tilde{O}\left(\frac{CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}})^{1/|\mathcal{I}|}}{\left(\frac{p''_{\boldsymbol{\eta}}}{\lambda^{\mathcal{L}\setminus\mathcal{I}}}\right)^{1/|\mathcal{I}|}}\right) = \tilde{O}\left(\frac{m \cdot \lambda^{\frac{2(\rho-|\mathcal{I}|)}{|\mathcal{I}|}}}{p^{1/|\mathcal{I}|}}\right) = \tilde{O}\left(\frac{m \cdot p^{\frac{2(\rho-|\mathcal{I}|)}{2\rho|\mathcal{I}|}}}{p^{1/|\mathcal{I}|}}\right) = \tilde{O}\left(\frac{m}{p^{1/\rho}}\right).$$

Regarding $\mathcal{Q}''_{light|\boldsymbol{\eta}}$, first verify that $attset(\mathcal{Q}''_{light|\boldsymbol{\eta}}) = \mathcal{L}\setminus\mathcal{I}$. Recall that the input relations of $\mathcal{Q}''_{light|\boldsymbol{\eta}}$ contain only light values. Hence, this join query is skew-free if we assign a share of λ to each attribute in $\mathcal{L}\setminus\mathcal{I}$. By Lemma 5, we can solve it in one round with load $\tilde{O}(m/\lambda^2) = \tilde{O}(m/p^{1/\rho})$ using $\lambda^{|\mathcal{L}\setminus\mathcal{I}|}$ machines.

Lemma 4 now tells us that $Join(\mathcal{Q}''_{\boldsymbol{\eta}})$ can be computed in one round with load $\tilde{O}(m/p^{1/\rho})$ using $(p''_{\boldsymbol{\eta}}/\lambda^{|\mathcal{L}\setminus\mathcal{I}|}) \cdot \lambda^{|\mathcal{L}\setminus\mathcal{I}|} = p''_{\boldsymbol{\eta}}$ machines. ◀

6 A 3-Round MPC Algorithm

Next, we will improve the algorithm in Section 5 by reducing the number of rounds to 3.

6.1 A New Approach to Handle Light Edges

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the hypergraph defined by the simple join query $\mathcal{Q}$ given. Fix an arbitrary subset $\mathcal{H}$ of $attset(\mathcal{Q})$. Recall that, for every light edge $e \in \mathcal{E}$, our 5-round algorithm needed to generate $R''_{e|\boldsymbol{\eta}}$ for every configuration $\boldsymbol{\eta}$ of $\mathcal{H}$. Next, we will describe an alternative approach to perform the join without explicitly computing $R''_{e|\boldsymbol{\eta}}$, which is crucial for obtaining a 3-round algorithm (to be presented in the next subsection).

Fix a configuration $\boldsymbol{\eta}$ of $\mathcal{H}$. Consider a light edge e of $\mathcal{G}$, and an attribute $X \in e$. Define $R^{\#X}_{e|\boldsymbol{\eta}}$ as follows:

- if X is not a border attribute, $R^{\#X}_{e|\boldsymbol{\eta}} = R'_{e|\emptyset}$;
- otherwise, $R^{\#X}_{e|\boldsymbol{\eta}}$ is a relation over e that consists of every tuple $\boldsymbol{u} \in R'_{e|\emptyset}$ satisfying $\boldsymbol{u}(X) \in R''_{X|\boldsymbol{\eta}}$.

▶ **Proposition 15.** $R''_{e|\boldsymbol{\eta}} = R^{\#X}_{e|\boldsymbol{\eta}} \bowtie R^{\#Y}_{e|\boldsymbol{\eta}}$.

Proof. See Appendix F. ◀

Example. Returning to the query $\mathcal{Q}$ in Figure 1a, consider again $\mathcal{H} = \{\mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{K}\}$, and the configuration $\boldsymbol{\eta}$ of $\mathcal{H}$ with $\boldsymbol{\eta}(\mathtt{D}) = \mathtt{d}$, $\boldsymbol{\eta}(\mathtt{E}) = \mathtt{e}$, $\boldsymbol{\eta}(\mathtt{F}) = \mathtt{f}$, and $\boldsymbol{\eta}(\mathtt{K}) = \mathtt{k}$. We will illustrate the above definitions by showing how to avoid explicitly computing $R''_{\{\mathtt{A},\mathtt{B}\}|\boldsymbol{\eta}}$ and $R''_{\{\mathtt{I},\mathtt{J}\}|\boldsymbol{\eta}}$ (namely, $R''_{\{\mathtt{A},\mathtt{B}\}}$ and $R''_{\{\mathtt{I},\mathtt{J}\}}$ in the description of Section 1.3). We will generate $R^{\#\mathtt{A}}_{\{\mathtt{A},\mathtt{B}\}|\boldsymbol{\eta}}$, $R^{\#\mathtt{B}}_{\{\mathtt{A},\mathtt{B}\}|\boldsymbol{\eta}}$, $R^{\#\mathtt{I}}_{\{\mathtt{I},\mathtt{J}\}|\boldsymbol{\eta}}$, and $R^{\#\mathtt{J}}_{\{\mathtt{I},\mathtt{J}\}|\boldsymbol{\eta}}$ such that $R''_{\{\mathtt{A},\mathtt{B}\}|\boldsymbol{\eta}} = R^{\#\mathtt{A}}_{\{\mathtt{A},\mathtt{B}\}|\boldsymbol{\eta}} \bowtie R^{\#\mathtt{B}}_{\{\mathtt{A},\mathtt{B}\}|\boldsymbol{\eta}}$ and $R''_{\{\mathtt{I},\mathtt{J}\}|\boldsymbol{\eta}} = R^{\#\mathtt{I}}_{\{\mathtt{I},\mathtt{J}\}|\boldsymbol{\eta}} \bowtie R^{\#\mathtt{J}}_{\{\mathtt{I},\mathtt{J}\}|\boldsymbol{\eta}}$.

Among the four relations to compute, $R^{\#\mathtt{J}}_{\{\mathtt{I},\mathtt{J}\}|\boldsymbol{\eta}}$ is the simplest because $\mathtt{J}$ is not a border attribute; hence, $R^{\#\mathtt{J}}_{\{\mathtt{I},\mathtt{J}\}|\boldsymbol{\eta}}$ equals $R'_{\{\mathtt{I},\mathtt{J}\}|\emptyset}$ (i.e., $R'_{\{\mathtt{I},\mathtt{J}\}}$ in Section 1.3). Regarding the other three relations, we will elaborate only on the generation of $R^{\#\mathtt{A}}_{\{\mathtt{A},\mathtt{B}\}}$ because the same ideas apply to $R^{\#\mathtt{B}}_{\{\mathtt{A},\mathtt{B}\}}$ and $R^{\#\mathtt{I}}_{\{\mathtt{I},\mathtt{J}\}}$.

Under $\boldsymbol{\eta}$, there are two unary residue relations defined over $\{\mathtt{A}\}$, namely, $R'_{\{\mathtt{A}\}|\mathtt{d}}$ and $R'_{\{\mathtt{A}\}|\mathtt{e}}$. The intersection of those two relations yields the unary relation $R''_{\{\mathtt{A}\}|\boldsymbol{\eta}}$ (i.e., $R''_{\{\mathtt{A}\}}$ in Section 1.3). Then, $R^{\#\mathtt{A}}_{\{\mathtt{A},\mathtt{B}\}}$ consists of every tuple $\boldsymbol{u}$ in the residue relation $R'_{\{\mathtt{A},\mathtt{B}\}|\emptyset}$ (i.e., $R'_{\{\mathtt{A},\mathtt{B}\}}$ in Section 1.3) whose $\boldsymbol{u}(A)$ appears in $R''_{\{\mathtt{A}\}|\boldsymbol{\eta}}$. ◀

Define:

$$\mathcal{Q}^{\#}_{light|\boldsymbol{\eta}} = \{R^{\#X}_{e|\boldsymbol{\eta}} \mid \text{every light edge } e \in \mathcal{E}, \text{every attribute } X \in e\}$$
$$\mathcal{Q}^{\#}_{\boldsymbol{\eta}} = \mathcal{Q}^{\#}_{light|\boldsymbol{\eta}} \cup \mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}}.$$

Proposition 15 immediately implies $Join(Q''_{\boldsymbol{\eta}}) = Join(Q^{\#}_{\boldsymbol{\eta}}) = CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}}) \times Join(\mathcal{Q}^{\#}_{light|\boldsymbol{\eta}})$.

6.2 The Algorithm

We are now ready to clarify how to solve $\mathcal{Q}$ in 3 rounds, concentrating on a specific subset $\mathcal{H}$ of $attset(\mathcal{Q})$ (set $k = |attset(\mathcal{Q})|$):

- Round 1: Generate the input relations of $\mathcal{Q}^{\#}_{\boldsymbol{\eta}}$ for every configuration $\boldsymbol{\eta}$ of $\mathcal{H}$.
- Round 2: Same as the statistical round in Section 5.
- Round 3: Evaluate $\mathcal{Q}^{\#}_{\boldsymbol{\eta}}$ for every $\boldsymbol{\eta}$.

It remains to elaborate on the details of Round 1 and 3.

Round 1. Allocate $p'_{\boldsymbol{\eta}} = \lceil p \cdot \frac{m_{\boldsymbol{\eta}}}{\Theta(m \cdot \lambda^{k-2})} \rceil$ machines to computing the relations of $\mathcal{Q}^{\#}_{\boldsymbol{\eta}}$. Let us focus on a specific configuration $\boldsymbol{\eta}$. To generate the relations in $\mathcal{Q}^{\#}_{light|\boldsymbol{\eta}}$, we carry out one-attribute reduction (see Section 2.4) for every border attribute $X \in \mathcal{L}$. Specifically, this operation is performed on

- the unary relations $R'_{\{X\}|\boldsymbol{\eta}[e\setminus\{X\}]}$ for all cross edges e of $\mathcal{G}$ incident to X, and
- the binary relations $R'_{e|\emptyset}$ for all light edges e of $\mathcal{G}$ incident to X.

It generates the $R^{\#X}_{e|\boldsymbol{\eta}}$ for every light edge e incident to X. Note that a value $x \in \mathbf{dom}$ is a heavy-hitter for this operation only if x appears in some input relation of $\mathcal{Q}$ $m_{\boldsymbol{\eta}}/p'_{\boldsymbol{\eta}} = \Omega(\frac{m \cdot \lambda^{k-2}}{p}) = \Omega(m/p^{1/\rho})$ times. Therefore, every machine can independently figure out the heavy-hitters from its histogram, and send each tuple in its local storage directly to the corresponding machines where the tuple is needed to perform one-attribute reductions. By Lemma 6, all the one-attribute reductions entail an overall load of $\tilde{O}(p + m_{\boldsymbol{\eta}}/p'_{\boldsymbol{\eta}}) = \tilde{O}(p + m/p^{1/\rho})$.

The relations in $\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}}$ can be easily produced by set intersection. Specifically, for every isolated attribute $X \in \mathcal{I}$, we obtain $R''_{\{X\}|\boldsymbol{\eta}}$ as the intersection of all the unary relations $R'_{\{X\}|\boldsymbol{\eta}[e\setminus\{X\}]}$, where e ranges over all cross edges of $\mathcal{G}$ incident to X. This can be done by standard hashing in one round with load $\tilde{O}(m_{\boldsymbol{\eta}}/p'_{\boldsymbol{\eta}}) = \tilde{O}(m/p^{1/\rho})$.

Round 3. Allocating $p''_{\boldsymbol{\eta}}$, as is given in (6), machines to each configuration $\boldsymbol{\eta}$ of $\mathcal{H}$, we compute $\mathcal{Q}^{\#}_{\boldsymbol{\eta}}$ in exactly the same way Lemma 14 computes $\mathcal{Q}''_{\boldsymbol{\eta}}$. In fact, the statement of Lemma 14, as well as the proof, holds verbatim by replacing every $\mathcal{Q}''_{\boldsymbol{\eta}}$ with $\mathcal{Q}^{\#}_{\boldsymbol{\eta}}$ and every $\mathcal{Q}''_{light|\boldsymbol{\eta}}$ with $\mathcal{Q}^{\#}_{light|\boldsymbol{\eta}}$.

We thus have obtained a 3-round algorithm for answering a simple join query with load $\tilde{O}(p^2 + m/p^{1/\rho})$ which is $\tilde{O}(m/p^{1/\rho})$ under our assumption $m \geq p^3$. This establishes the second main result of this paper:

▶ **Theorem 16.** *Given a simple join query with input size m and a fractional edge covering number ρ, we can answer it in the MPC model using p machines in two rounds with load $O(m/p^{1/\rho})$, assuming that $m \geq p^3$, and that each machine has been preloaded with a histogram as is prescribed in Section 5.*

It is worth mentioning that Round 2 of our algorithm (i.e., the statistical round) has a load of $O(p^2)$ such that only the first and third rounds of the algorithm entail a load sensitive to m.

References

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

2 Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel J. Abadi, Alexander Rasin, and Avi Silberschatz. HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. *Proceedings of the VLDB Endowment (PVLDB)*, 2(1):922–933, 2009.

3 Foto N. Afrati and Jeffrey D. Ullman. Optimizing Multiway Joins in a Map-Reduce Environment. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 23(9):1282–1298, 2011.

4 Albert Atserias, Martin Grohe, and Daniel Marx. Size Bounds and Query Plans for Relational Joins. *SIAM J. Comput.*, 42(4):1737–1767, 2013.

5 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication Steps for Parallel Query Processing. *Journal of the ACM (JACM)*, 64(6):40:1–40:58, 2017.

6 Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 137–150, 2004.

7 Xiao Hu, Paraschos Koutris, and Ke Yi. An External-Memory Work-Depth Model and Its Applications to Massively Parallel Join Algorithms. *Manuscript*, 2018.

8 Xiaocheng Hu, Miao Qiao, and Yufei Tao. I/O-efficient join dependency testing, loomis-whitney join, and triangle enumeration. *Journal of Computer and System Sciences (JCSS)*, 82(8):1300–1315, 2016.

9 Bas Ketsman and Dan Suciu. A Worst-Case Optimal Multi-Round Algorithm for Parallel Computation of Conjunctive Queries. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 417–428, 2017.

10 Paraschos Koutris, Paul Beame, and Dan Suciu. Worst-Case Optimal Algorithms for Parallel Query Processing. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 8:1–8:18, 2016.

11 Hung Q. Ngo, Ely Porat, Christopher Re, and Atri Rudra. Worst-case Optimal Join Algorithms. *Journal of the ACM (JACM)*, 65(3):16:1–16:40, 2018.

12 Rasmus Pagh and Francesco Silvestri. The input/output complexity of triangle enumeration. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 224–233, 2014.

13 Edward R. Scheinerman and Daniel H. Ullman. *Fractional Graph Theory: A Rational Approach to the Theory of Graphs*. Wiley, New York, 1997.

14 Todd L. Veldhuizen. Triejoin: A Simple, Worst-Case Optimal Join Algorithm. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 96–106, 2014.

15 Mihalis Yannakakis. Algorithms for Acyclic Database Schemes. In *Very Large Data Bases, 7th International Conference, September 9-11, 1981, Cannes, France, Proceedings*, pages 82–94, 1981.

A Proof of Lemma 6

For each $i \in [1, b]$, divide S_i into (i) S_i^1, which includes the tuples of $\boldsymbol{u} \in S_i$ where $\boldsymbol{u}(X)$ is a heavy-hitter, and (ii) $S_i^2 = S_i \setminus S_i^1$. Accordingly, divide $S_i^{\#}$ into (i) $S_i^{\#1}$, which includes the tuples of $\boldsymbol{u} \in S_i^{\#}$ where $\boldsymbol{u}(X)$ is a heavy-hitter, and (ii) $S_i^{\#2} = S_i^{\#} \setminus S_i^{\#1}$. We will compute $S_i^{\#1}$ and $S_i^{\#2}$, separately.

The computation of $S_1^{\#1}, ..., S_b^{\#1}$ is trivial. Since there are at most p heavy-hitters, each machine storing a heavy-hitter x in some R_j ($j \in [1, a]$) simply broadcasts the pair (x, j) to all machines. This takes one round with load $O(p)$. A machine holding a tuple $\boldsymbol{u}$ with $\boldsymbol{u}(X) = x$ in some S_i ($i \in [1, a]$) adds $\boldsymbol{u}$ to $S_i^{\#1}$ only if it has received (x, j) for all $j \in [1, a]$.

$S_1^{\#2}, ..., S_b^{\#2}$, on the other hand, can be produced using Lemma 5. Assign a share of p to X and a share of 1 to every other attribute. By definition, the join query $\{R_1, ..., R_a, S_1^2, ..., S_b^2\}$ is skew-free, and therefore, can be solved in round round with load $\tilde{O}(n/p)$. For each $i \in [1, b]$, $S_i^{\#2}$ can then be easily obtained from the result of this query.

B Proof of Proposition 7

Let us first introduce a definition. Suppose that $\mathcal{S}$ is a subset of $\mathcal{H}$. We say that a configuration $\boldsymbol{\eta}_{\mathcal{H}}$ of $\mathcal{H}$ *extends* a configuration $\boldsymbol{\eta}_{\mathcal{S}}$ of $\mathcal{S}$ if $\boldsymbol{\eta}_{\mathcal{S}} = \boldsymbol{\eta}_{\mathcal{H}}[\mathcal{S}]$. $O(\lambda^{|\mathcal{H}|-|\mathcal{S}|})$ configurations $\boldsymbol{\eta}_{\mathcal{H}}$ of $\mathcal{H}$ can extend $\boldsymbol{\eta}_{\mathcal{S}}$, because every attribute in $\mathcal{H} \setminus \mathcal{S}$ has $O(\lambda)$ heavy values.

Returning to the proof of the proposition, Let $\boldsymbol{\eta}$ be an arbitrary configuration of $\mathcal{H}$, and $e \in \mathcal{E}$ an arbitrary hyperedge that is active on $\boldsymbol{\eta}$. Define $\mathcal{H}' = e \cap \mathcal{H}$ and $e' = e \setminus \mathcal{H}$. A tuple $\boldsymbol{u} \in R_e$ belongs to $R_{e|\boldsymbol{\eta}[e \setminus e']}$ only if $\boldsymbol{\eta}$ extends the configuration $\boldsymbol{u}[\mathcal{H}']$ of $\mathcal{H}'$. There are $O(\lambda^{|\mathcal{H}|-|\mathcal{H}'|})$ such $\boldsymbol{\eta}$. As $|\mathcal{E}| = O(1)$, $\boldsymbol{u}$ can contribute $O(1)$ to the term $m_{\boldsymbol{\eta}}$ for at most $O(\lambda^{|\mathcal{H}|-|\mathcal{H}'|})$ different $\boldsymbol{\eta}$.

It remains to prove that $|\mathcal{H}| - |\mathcal{H}'| \leq k - 2$. Observe that $|\mathcal{H}| - |\mathcal{H}'|$ is the number of attributes in $\mathcal{H}$ that do *not* belong to e. This number is at most $k - 2$ because e has two attributes.

C Proof of Proposition 8

The second equality follows directly from the fact that the scheme of each relation in $\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}}$ is disjoint with the scheme of any other relation in $\mathcal{Q}''_{\boldsymbol{\eta}}$. Next we focus on proving the first equality.

We first show $Join(\mathcal{Q}'_{\boldsymbol{\eta}}) \subseteq Join(\mathcal{Q}''_{\boldsymbol{\eta}})$. Consider an arbitrary tuple $\boldsymbol{u} \in Join(\mathcal{Q}'_{\boldsymbol{\eta}})$. For any attribute $X \in \mathcal{L}$ and any cross edge e of $\mathcal{G}$ containing X, since $\boldsymbol{u}(X) \in R_{\{X\}|\boldsymbol{\eta}[e\setminus\{X\}]}$, it must hold that $\boldsymbol{u}(X) \in R''_{\{X\}|\boldsymbol{\eta}}$. For any light edge $e = \{X, Y\} \in \mathcal{E}$, since $\boldsymbol{u}[e] \in R'_{e|\emptyset}$, it must hold that $\boldsymbol{u}[e] \in R''_{e|\boldsymbol{\eta}}$. It thus follows that $\boldsymbol{u} \in Join(\mathcal{Q}''_{\boldsymbol{\eta}})$.

Next, we show $Join(\mathcal{Q}''_{\boldsymbol{\eta}}) \subseteq Join(\mathcal{Q}'_{\boldsymbol{\eta}})$. Consider an arbitrary tuple $\boldsymbol{u} \in Join(\mathcal{Q}''_{\boldsymbol{\eta}})$. For any attribute $X \in \mathcal{L}$, since $\boldsymbol{u}(X) \in R''_{\{X\}|\boldsymbol{\eta}}$, it must hold that $\boldsymbol{u}(X) \in R_{\{X\}|\boldsymbol{\eta}[e\setminus\{X\}]}$ for any cross edge e of $\mathcal{G}$ containing X. For any light edge $e = \{X, Y\} \in \mathcal{E}$, since $\boldsymbol{u}[e] \in R''_{e|\boldsymbol{\eta}}$, it must hold that $\boldsymbol{u}[e] \in R'_{e|\emptyset}$. It thus follows that $\boldsymbol{u} \in Join(\mathcal{Q}'_{\boldsymbol{\eta}})$.

D Proof of Lemma 10

Consider any $R''_{\{X\}|\boldsymbol{\eta}} \in \mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta}}$. Observe that the content of $R''_{\{X\}|\boldsymbol{\eta}}$ does not depend on $\boldsymbol{\eta}(Y)$ for any $Y \in \mathcal{H} \setminus \mathcal{F}$. In other words, if we set $\boldsymbol{\eta'} = \boldsymbol{\eta}[\mathcal{F}]$, then $R''_{\{X\}|\boldsymbol{\eta}}$ is precisely the same as $R''_{\{X\}|\boldsymbol{\eta'}}$. Notice that $\boldsymbol{\eta'}$ is a configuration of $\mathcal{F}$ that is extended by $\boldsymbol{\eta}$ (see the proof of Proposition 7 for the definition of *extension*). The lemma follows from the fact that a configuration $\boldsymbol{\eta'}$ of $\mathcal{F}$ can be extended by $O(\lambda^{|\mathcal{H}|-|\mathcal{F}|})$ configurations $\boldsymbol{\eta}$ of $\mathcal{H}$.

E Proof of Lemma 12

We will prove

$$\bigcup_{\text{config. } \boldsymbol{\eta'} \text{ of } \mathcal{F}} CP(\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta'}}) \times \{\boldsymbol{\eta'}\} \quad \subseteq \quad Join(\mathcal{Q}^*). \tag{7}$$

from which the lemma follows.

Take a tuple $\boldsymbol{u}$ from the left hand side of (7), and set $\boldsymbol{\eta'} = \boldsymbol{u}[\mathcal{F}]$. Based on the definition of $\mathcal{Q}''_{\mathcal{I}|\boldsymbol{\eta'}}$, it is easy to verify that $\boldsymbol{u}[e] \in R_e$ for every cross edge $e \in \mathcal{E}$, and hence, $\boldsymbol{u}[e] \in R^*_e$. Furthermore, $\boldsymbol{u}(X) \in R^*_{\{X\}}$ for every $X \in \mathcal{F}$ because $\boldsymbol{u}(X) = \boldsymbol{\eta'}(X)$ is a heavy value. Finally, obviously $\boldsymbol{u}(Y) \in R^*_{\{Y\}}$ for every $Y \in \mathcal{I}_0$. All these facts together ensure that $\boldsymbol{u} \in Join(\mathcal{Q}^*)$.

F Proof of Proposition 15

Consider first the case where X and Y are both border attributes. We have

$$\begin{aligned} R''_{e|\boldsymbol{\eta}} &= R''_{X|\boldsymbol{\eta}} \bowtie R'_{e|\emptyset} \bowtie R''_{Y|\boldsymbol{\eta}} \\ &= (R''_{X|\boldsymbol{\eta}} \bowtie R'_{e|\emptyset}) \bowtie (R'_{e|\emptyset} \bowtie R''_{Y|\boldsymbol{\eta}}) \\ &= R^{\#X}_{e|\boldsymbol{\eta}} \bowtie R^{\#Y}_{e|\boldsymbol{\eta}}. \end{aligned}$$

If X is a border attribute but Y is not, then:

$$\begin{aligned} R''_{e|\boldsymbol{\eta}} &= R''_{X|\boldsymbol{\eta}} \bowtie R'_{e|\emptyset} \\ &= (R''_{X|\boldsymbol{\eta}} \bowtie R'_{e|\emptyset}) \bowtie R'_{e|\emptyset} \\ &= R^{\#X}_{e|\boldsymbol{\eta}} \bowtie R^{\#Y}_{e|\boldsymbol{\eta}}. \end{aligned}$$

If neither X nor Y is a border attribute, the proposition is trivial.