Cover or Pack: New Upper and Lower Bounds for Massively Parallel Joins

Xiao Hu
Duke University
xh102@cs.duke.edu

Abstract
This paper considers the worst-case complexity of multi-round join evaluation in the Massively Parallel Computation (MPC) model. Unlike the sequential RAM model, in which there is a unified optimal algorithm based on the AGM bound for all join queries, worst-case optimal algorithms have been achieved on a very restrictive class of joins in the MPC model. The only known lower bound is still derived from the AGM bound, in terms of optimal fractional edge covering number of the query.

In this work, we make efforts towards bridging this gap. We first design an instance-dependent algorithm for the class of acyclic join queries. In particular, when the maximum size of input relations is bounded, this complexity has a closed form in terms of the optimal fractional edge covering number, which is worst-case optimal. Beyond acyclic joins, we surprisingly find that optimal fractional edge covering number doesn’t lead to a tight lower bound any more. More specifically, we prove for a class of cyclic joins a better lower bound in terms of optimal fractional edge packing number of the query, which is also tight as matched by existing algorithms. This new result displays a significant distinction not only between acyclic and cyclic joins, but also between the fine-grained RAM and coarse-grained MPC model.

ACM Reference Format:

1 Introduction
Evaluating join queries is one of the most central problems in relational databases, both in theory and practice. The worst-case complexity of join evaluation started to be unraveled, largely thanks to the work of Atserias, Grohe, and Marx [4], who gave a worst-case bound on the join size, known as AGM bound. More specifically, the maximum possible join size is always bounded by $O(N^\rho)$, where $N$ is the maximum size of input relations and $\rho$ is the optimal fractional edge covering number of the join query, which is also tight with an instance outputting $\Theta(N^\rho)$ join results. This then led to worst-case optimal join algorithms [22, 26] in the RAM model. All joins display a unified form of worst-case complexity of $O(N^\rho)$.

Ngo, Ré, and Rudra [23] presented a nice survey of these results, and also gave a simpler and unified proof for both the AGM bound and the running time of the algorithm.

Meanwhile, massively parallel algorithm has received much more attention in recent years due to the rapid development of massively parallel systems such as MapReduce [9] and Spark [28]. Join evaluation in the massively computational model are quite different from the RAM model, where an efficient algorithm should make best use of data locality, i.e., the algorithm tries to send as many tuples that can be joined as possible to one machine so that it can produce their join results. Intriguing questions arise, can we also achieve a unified worst-case optimal join algorithm in the MPC model? Is the worst-case complexity only related to the optimal fractional edge covering number; if not, what other query-dependent quantities? This work will answer these two questions.

1.1 Join Query
A (natural) join is defined as a hypergraph $Q = (V, E)$, where the vertices $V = \{x_1, \ldots, x_n\}$ model the attributes and the hyperedges $E = \{e_1, \ldots, e_m\} \subseteq 2^V$ model the relations [1]. Let $\text{dom}(x)$ be the domain of attribute $x \in V$. An instance of $Q$ is a set of relations $R = \{R(e) : e \in E\}$, where $R(e)$ is a set of tuples, where each tuple is an assignment that assigns a value from $\text{dom}(x)$ to $x$ for every $x \in e$. We use $N = \max_{e \in E}|R(e)|$ to denote the maximum size of input relations. The join results of $Q$ on $R$, denoted as $Q(R)$, consist of all combinations of tuples, one from each $R(e)$, such that they share common values on their common attributes. People usually study the data complexity of join algorithms, i.e., we assume that the query size, namely $n$ and $m$, are constants. Hence, the total number of input tuples, denoted as input size, is always $O(N)$.

For a join query $Q = (V, E)$, two query-related quantities as edge covering and edge packing will be commonly used throughout this paper. Let $f$ be a mapping from $E$ to $[0, +\infty)$. Note that $f$ is a fractional edge covering if

$$\sum_{e \in E} f(e) \geq 1, \text{ for all } v \in V$$

and a fractional edge packing if

$$\sum_{e \in E} f(e) \leq 1, \text{ for all } v \in V$$

The quantity $\sum_e f(e)$ is noted as the number of $f$, where the optimal fractional edge covering denoted as $\rho^*$ is the one with minimum number, and the optimal fractional edge packing denoted as $\tau^*$ is the one with maximum number. Generally, there is no clear relation between $\tau^*$ and $\rho^*$, except for some specific joins.

1.2 The model of computation
We consider the MPC model [2, 3, 6, 7, 18–20], which has become the standard model of computation for studying massively parallel algorithms, especially for join algorithms. In the MPC model, data is initially distributed evenly over $p$ servers with each server holding $O(\frac{N}{p})$ tuples. Computation proceeds in rounds. In each round, each server first sends messages to other servers, receives messages from other servers, and then does some local computation. The complexity of the algorithm is measured by the number of rounds.
In this work, we will focus on worst-case optimality, which is the most commonly used measurement for algorithm design, providing theoretical guarantees for the performance of algorithms in the worst case. More specifically, the entire space of input instances is divided into classes, where instances in the same class share the same input size N. An algorithm is worst-case optimal if its complexity is optimal on the worst instance for each class. Further subdividing the instance space leads to more refined analyses, for example, output-optimal algorithms take both input size N and output size OUT as parameters to divide the instance space, and display optimality on the worst instance for each class. In extreme cases where each class contains just one instance, we obtain instance-optimal algorithms. Note that by definition, an instance-optimal algorithm must be output-optimal, and an output-optimal algorithm must be worst-case optimal, but the reserve direction may not be true. We refer interested readers to [15] for a summary of fine-grained join algorithms in the MPC model.

In the RAM model, there is a unified worst-case optimal algorithm [23] for computing all joins in O(N_p^N) time. The situation becomes much more interesting in the MPC model. All results to be reviewed for the MPC model are put into Table 1 and the relationship between join queries mentioned is clarified in Figure 1.

Previous efforts have been put to understand what can be done in a single round in the MPC model. Initially, a one-round hashing-based algorithm [3, 6], named as hypercube, was proposed for computing all joins on non-skewed input instances with load $\tilde{O}(N_p^N)$. Later, an improved algorithm built upon hypercube has been proposed by the same researchers [19], for computing all joins with arbitrary input instances. This algorithm still runs in a single round, but incurs a higher load of $\tilde{O}(N_p^N)$, where $\psi^*$ is the optimal fractional edge quasi-packing number of the query. This result has been proved to be optimal (up to a polylog factor) for single-round computation with arbitrary input instances. It should be noted that $\psi^* \geq r^*$ [19].

Meanwhile, people found that even allowing a constant number of rounds may bring a significant (polynomially) reduction in the overall cost. Consider an example join query $Q = (\mathcal{V}, E)$, the edge quasi-packing number is defined as follows. Let $x \subseteq \mathcal{V}$ be any subset of vertices of $\mathcal{V}$. Define the residual hypergraph after removing attributes $x$ as $Q_x = (\mathcal{V}_x, E_x)$, where $\mathcal{V}_x = \mathcal{V} - x$ and $E_x = \{e - x : e \in E\}$. The edge quasi-packing number of $Q$ is the maximum optimal fractional edge packing number over all $Q_x$'s, i.e., $\psi^* = \max_{x \subseteq \mathcal{V}} r^*(Q_x)$.

Table 1: Worst-case complexity of join evaluation in the MPC model. $N$ is the maximum size of input relations. $p$ is the number of servers. $\psi^*$ is the optimal fractional edge quasi-packing number. $r^*$ is the optimal fractional edge covering number. $\rho^*$ is the optimal fractional edge covering number. It is known that $\psi^* \geq \max\{\rho^*, r^*\}$ [19].

<table>
<thead>
<tr>
<th>Joins</th>
<th>one-round</th>
<th>multi-round</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$-acyclic</td>
<td>$O\left(\frac{N}{p^{1+\epsilon}}\right)$ is achieved for r-hierarchical join [15] and all $\alpha$-acyclic joins [Theorem 4]</td>
<td>$\Omega\left(\frac{N}{p^{1+\epsilon}}\right)$ for $\Box$-join [Theorem 5] and some degree-two joins [Theorem 6]</td>
</tr>
<tr>
<td>cyclic</td>
<td>$\tilde{O}\left(\frac{N}{p^{1+\epsilon}}\right)$ is achieved for binary-relation join [18, 19, 25] and the Loosin-Witney join [19]</td>
<td>$\Omega\left(\frac{N}{p^{1+\epsilon}}\right)$ for $\Box$-join [Theorem 5] and some degree-two joins [Theorem 6]</td>
</tr>
</tbody>
</table>

Figure 1: Classification of join queries.
where $n$ relations contain one distinct attribute and one additional relation contains all attributes. This opens up new opportunities for how to compute multi-way join in the MPC model.

The goal of a multi-round worst-case optimal algorithm in the MPC model is believed to achieve a load of $O\left(\frac{N}{p^{\rho}}\right)$. The reason why this is a reasonable lower bound can be argued by the counting argument: Each server can only produce $O(L^p)$ join results in a constant number rounds with the load limited to $L$ (also implied by the AGM bound [4]), so all the $p$ servers can produce at most $O(p \cdot L^p)$ join results. Thus, producing $N^p$ join results needs at least a load of $L = \Omega\left(\frac{N}{p^{\rho}}\right)$. So far, this bound (up to polylog factors) has been achieved on some specific classes of joins [18, 19, 25], such as binary-join where each relation has at most two attributes, and loomis-whitney join. All these algorithms resort to the heavy-light decomposition technique for tackling data skew, and then invoke the hypercube algorithm as primitives for handling non-skewed instances. However, whether this bound can be achieved for arbitrary joins, or even just $a$-acyclic joins, is still open.

Besides, several output-optimal (or output-sensitive) algorithms have been proposed for join queries in the MPC model. For the class of $r$-hierarchical joins, an output-optimal algorithm has been proposed in [15], which must also be worst-case optimal. Note that $r$-hierarchical join is a very restrictive class of join; for example, the simplest line-3 join query $R_1(A, B) \bowtie R_2(B, C) \bowtie R_3(C, D)$ is not $r$-hierarchical. Meanwhile, the classical Yannakakis algorithm can be as large as $\Omega(\frac{N}{p^{\rho}})$ I/Os, shadowing the previous work [11].

It is remarkable that a reduction from the MPC model to the external memory (EM) model has been established in [19] in a cost-preserving way, such that any MPC algorithm running in $r$ rounds with load $L(N, p)$ can be converted to an external memory algorithm incurring $O\left(\frac{N}{p} + \rho p\frac{M}{\rho}\right)$ I/Os, where $\rho = \min_p\{L(N, p) \leq M/r\}$. Implied by this reduction, worst-case optimal algorithms can be automatically obtained for LW join and binary-join in the EM model. It is worth mentioning that a worst-case optimal algorithm has been proposed [14] for $a$-acyclic join in the EM model using $O\left(\frac{N}{p^{\rho}} \cdot \frac{M}{\rho}\right)$ I/Os, without a counterpart in the MPC model. However, $a$-acyclic join is a very restrictive sub-class of $a$-acyclic join; for example, a simple join query $R_0(A, B, C) \bowtie R_1(A, B, D) \bowtie R_2(B, C, E) \bowtie R_3(A, C, F)$ is $a$-acyclic but not $a$-acyclic, thus cannot be handled by the algorithm in [14]. On the other hand, there is no result showing any conversion from the sequential EM model to the parallel MPC model. We won’t pursue this dimension further.

### 1.4 Our Results

\[ A \text{ join query } Q = \langle V, E \rangle \text{ is a loomis-whitney (LW) join if } E = \{ \langle x \rangle : \forall x \in V \}. \]

Moreover, it has $\rho^* = \tau^* = n/(n - 1)$, where $n = |V|$. As it is a very restrict class of joins with highly symmetric structures, we omit it in the following discussion.

\[ A \text{ worst-case I/O-optimal algorithm for LW join was proposed in [12] independently.} \]

Our main results are also summarized in Table 1, which can be split into two parts: new upper bound for $a$-acyclic joins and new lower bound for some cyclic joins. We also include a brief connection of results from these two sides.

### New Upper Bound.

The primary class of join queries we target in this work is the $a$-acyclic join [8], which is the most commonly studied class of acyclic joins in database theory. Formally, a join query $Q = \langle V, E \rangle$ is $a$-acyclic if there exists an undirected tree $T$ whose nodes are in one-to-one correspondence with the edges in $E$ such that for any vertex $v \in V$, all nodes containing $v$ form a connected subtree. Such a tree $T$ is called the join tree of $Q$. An example of an acyclic join is illustrated in Figure 4.

We propose an instance-dependent algorithm for computing any acyclic join $Q$ in the MPC model, whose load complexity is in terms of subjoins (the formal definition is given in Section 3). Subjoins are proposed to relate the complexity of our algorithm to the lower bound on each sub-query of $Q$. When all relations contain at most $N$ tuples, this complexity has a closed form of $O\left(\frac{N}{p^{\rho}}\right)$, matching the lower bound $O\left(\frac{N}{p^{\rho}}\right)$.

This result has reduced the complexity of acyclic joins evaluation from $O\left(\frac{N}{p^{\rho}}\right)$ to $O\left(\frac{N}{p^{\rho}}\right)$ since $\rho^* \geq \rho^*$ [19], only increasing the number of rounds from 1 to a constant. This improvement could be significant because of the possibly huge gap between $\rho^*$ and $\rho^*$, as we have seen on the example in Section 1.3. In general, we notice several important sub-classes of acyclic joins on which this gap can be as large as $\Theta(m + n)$, in terms of the query size, including path join and star-dual join. We refer interested reader to [19] for details. Moreover, by the MPC-EM reduction, this result implies an EM algorithm for computing all acyclic joins with $O\left(\frac{N\rho}{p} \cdot \frac{M^\rho}{\rho^*}\right)$ I/Os, shadowing the previous work [11].

### New Lower Bound.

Finally, we turn to cyclic joins. Surprisingly, we find that $O\left(\frac{N}{p^{\rho}}\right)$ is not necessarily a correct target for multi-round worst-case optimal join algorithms, since the existing lower bound $O\left(\frac{N}{p^{\rho}}\right)$ is not tight any more. We start by answering an open question posed in [18]: On the $\exists$-join $Q_{\exists} = \{ R_1(A, B, C) \bowtie R_2(D, E, F) \bowtie R_3(A, D) \bowtie R_4(B, E) \bowtie R_5(C, F) \}$, whether there exists a better upper bound than $O\left(\frac{N}{p^{\rho}}\right)$, or a better lower bound than $O\left(\frac{N}{p^{\rho}}\right)$? As shown in Figure 2, $Q_{\exists}$ has $\rho^* = 2$ by choosing $\{R_1, R_2\}$ in the fractional edge cover and $\tau^* = 3$ by choosing $\{R_3, R_4, R_5\}$ in the fractional edge packing. We show a probabilistic hard instance on which any MPC algorithm computing it in $O(1)$ rounds must incur a load of $O\left(\frac{N}{p^{\rho^*}}\right)$. The intuition is that such an instance has "dense" join results, which is indeed as large as the AGM bound, but each server cannot achieve high efficiency in emitting the join results, no matter which combinations of input tuples it receives. Any attempts in lowering this bound further would break the counting

---

1 Other notions of acyclicity have been proposed, including berge-acyclicity, $\gamma$-acyclicity and $\beta$-acyclicity. Moreover, berge-acyclicity implies $\gamma$-acyclicity which implies $\beta$-acyclicity which implies acyclicity.

2 In the following of this work, "acyclic" always means "$a$-acyclic" if not specified.

3 A path join $Q = \langle V, E \rangle$ is defined as $V = \{x_1, x_2, \ldots, x_n\}$ and $E = \{e_i = \{x_i, x_{i+1}\} : i \in \{1, 2, \ldots, n - 1\}\}$.

4 A star-dual join $Q = \langle V, E \rangle$ is defined as $V = \{x_1, x_2, \ldots, x_n\}$ and $E = \{e_i = \{x_i\} : i \in \{1, 2, \ldots, n\}\}$. 

5 As shown in Figure 2, $Q_{\exists}$ has $\rho^* = 2$ by choosing $\{R_1, R_2\}$ in the fractional edge cover and $\tau^* = 3$ by choosing $\{R_3, R_4, R_5\}$ in the fractional edge packing. We show a probabilistic hard instance on which any MPC algorithm computing it in $O(1)$ rounds must incur a load of $O\left(\frac{N}{p^{\rho^*}}\right)$. The intuition is that such an instance has "dense" join results, which is indeed as large as the AGM bound, but each server cannot achieve high efficiency in emitting the join results, no matter which combinations of input tuples it receives. Any attempts in lowering this bound further would break the counting
argument that every join result must be emitted at least once, thus violating the correctness of join algorithms. Meanwhile, the existing algorithm [19] can compute it in a single round with load \(\tilde{O}(\frac{N}{\rho^{1/\epsilon}})^9\), which is already optimal implied by our new lower bound.

This framework of lower bound proof for \(Q_E\) can be extended to a larger class of cyclic join queries, noted as degree-two joins, in which each attribute appears in two relations. Observe that the dual\(^1\) of a degree-two join is a binary-relation join; hence, degree-two joins enjoy very nice properties, implied by those of binary-relation joins [24]. More specifically, we characterize the edge-packing-provable condition, under which there exists a better (at least not worse) lower bound in terms of \(\Omega(\frac{N}{\rho^{1/\epsilon}})\) for computing any degree-two join in the MPC model.

Cover or Pack. From the lower bound side, we know that the worst-case complexity of join evaluation displays \(\Omega(\frac{N}{\rho^{1/\epsilon}})\) for binary-relation join, LW join and acyclic joins, and \(\Omega(\frac{N}{\rho^{1/\epsilon}})\) for some class of cyclic joins, including the \(\square\)-join. However, there is no clear distinction on the relative ordering between \(\rho^*\) and \(\tau^*\), at least for acyclic joins and binary-relation joins. A natural question arises: cover or pack, which one is the correct measurement for the worst-case complexity of join evaluation.

To clarify this question, we first introduce the notion of reduced join. A reduce procedure on a hypergraph \((\mathcal{V}, E)\) is to remove an edge \(e \in E\) if there exists another edge \(e' \in E\) such that \(e \subseteq e'\). We can repeatedly apply the reduce procedure until no more edge can be reduced, and the resulting hypergraph is said to be reduced. From the upper bound side, a join query can be reduced in \(O(1)\) rounds with linear load (see Section 2), thus the hardness of multi-round computation comes from the reduced join.

Now, we can draw some distinction for the reduced join queries: (1) \(\tau^* \leq \rho^*\) holds for reduced berge-acyclic joins (see Lemma A.3) and reduced\(^2\) binary-relation joins [18]; and (2) \(\rho^* \leq \tau^*\) holds for reduced degree-two joins (see Lemma 6.3), as shown in Figure 3. Moreover, we conjectured that \(\tau^* \leq \rho^*\) holds for reduced \(\alpha\)-acyclic joins, but the formal proof is open. Existing results indicate that the worst-case complexity of multi-round join evaluation is determined by the maximum of cover and pack on the reduced join; however, this conjecture is still open for general cyclic joins.

1.5 Outline

This paper is organized as follows. In Section 2, we review some basic primitives that will be commonly used in our MPC algorithm.

2 MPC PRELIMINARIES

We mention the following deterministic primitives in the MPC model, which can be computed with load \(\tilde{O}(\frac{N}{\rho^{1/\epsilon}})\) in \(O(1)\) rounds. Assume \(N > \rho^{1/\epsilon}\) where \(\epsilon > 0\) is any small constant.

Reduce-by-key [13]. Given \(N\) pairs in terms of (key, value), compute the "sum" of values for each key, where the "sum" is defined by any associative operator.

This primitive will also be frequently used to compute data statistics, for example the degree information. The degree of value \(a \in \text{dom}(v)\) in relation \(R(e)\) is defined as the number of tuples in \(R(e)\) having this value in attribute \(v\), i.e., \(|\sigma_{=a}(R(e))|\). Each tuple \(t \in R(e)\) is considered to have "key" \(\pi_t(e)\) and "value" 1.

Semi-Join [15]. Given two relations \(R_1\) and \(R_2\) with a common attribute \(v\), the semi-join \(R_1 \times R_2\) returns all the tuples in \(R_1\) whose value on \(v\) matches that of at least one tuple in \(R_2\).

For any acyclic join, all dangling tuples, i.e., those that will not participate in the full join results, can be removed by a series of semi-joins [27].

Parallel-packing [15]. Given \(N\) numbers \(x_1, x_2, \ldots, x_N\) where \(0 < x_i \leq 1\) for \(i \in [N]\), group them into \(m\) sets \(Y_1, Y_2, \ldots, Y_m\) such that \(\sum_{i \in Y_j} x_i \leq 1\) for all \(j\), and \(\sum_{i \in Y_j} x_i \geq \frac{1}{2}\) for all but one \(j\). Initially, the \(N\) numbers are distributed arbitrarily across all servers, and the algorithm should produce all pairs \((i, j)\) if \(i \in Y_j\) when done. Note that \(m \leq 1 + 2 \sum_i x_i\).

3 GENERIC ACYCLIC JOIN ALGORITHM

In this section, we study how to compute the class of acyclic joins efficiently in the MPC model. Before dividing into the algorithmic details, we first define the following concepts to simplify our description. Recall that in an acyclic join, its relations can be organized into a join tree \(T\) such that for each attribute, the nodes containing this attribute form a connected subtree in \(T\). An example is illustrated in Figure 4. It should be noted that if there exists some relation \(e \in E\) which doesn’t share any common attributes with other relations, then we just add it to any node \(e' \in E\) in the join tree as its child; moreover add a dummy attribute to both \(e\) and \(e'\).
so that all tuples in $R(e)$ and $R(e')$ share the same value on this dummy attribute. Moreover, an attribute is unique if it only appears in one relation.

![Join Tree Example](image)

**Figure 4:** A join tree $T$ of join query $Q = \langle V, E \rangle$, where $\mathcal{V} = \{A, B, C, D, E, F, G, H, I, J, K\}$ and $\mathcal{E} = \{e_0(ABCH), e_1(ABD), e_2(BCE), e_3(ACF), e_4(ABHJ), e_5(AHI), e_6(AIK), e_7(AIG)\}$.

A nice property of this join tree will be used by our generic algorithm is stated by Lemma 3.1. In plain language, for a leaf $e_1$ and its parent $e_0$ after removing all common attributes between $e_1$ and $e_0$, $e_1$ will become isolated from the remaining ones.

**Lemma 3.1.** For an acyclic join $Q = \langle V, E \rangle$ with its join tree $T$, consider an arbitrary leaf node $e_1$ and its parent $e_0$. For any $e \in E - \{e_0, e_1\}$, $(e \cap e_1) - e_0 = \emptyset$.

**Proof.** By contradiction, assume $v \in (e \cap e_1) - e_0$. Implied by the definition of join tree $T$, all edges containing $v$ form a connected subtree of $T$. As $e_1$ is only connected to $e_0$, $v \in e_0$ if $v \in e \cap e_1$, coming to a contradiction.

### 3.1 Subjoin

To better describe our algorithm, we introduce the notion of subjoin, defined on the subset of relations in the join query $Q = \langle V, E \rangle$, with join tree $T$ and input instance $R$. Take an arbitrary subset of relations $S \subseteq E$. We define $T[S]$ as the set of maximally connected components of $S$ on the join tree $T$. It should be carefully distinguished from the notion of maximally connected components of $S$.

**Example 3.2.** Let’s take two examples in Figure 4 for illustration. On $S_1 = \{e_1, e_3, e_7\}$, $S_1$ itself is a single connected component since all edges share the common attribute $A$; however, all edges are not directly connected on the join tree $T$, so $T[S_1] = \{(e_1), (e_3), (e_7)\}$. Adding one more edge $e_0$ to $S_1$, denoted as $S_2 = \{e_0, e_1, e_3, e_7\}$, does not change the fact that $S_2$ is a single connected component, but $e_0, e_1, e_3$ form a connected component on the join tree $T$, so $T[S_2] = \{(e_0, e_1, e_3), (e_7)\}$.

**Definition 1 [Subjoin].** For a join query $Q = \langle V, E \rangle$ with join tree $T$, and an instance $R$, the subjoin of $S \subseteq E$ is defined as

$$(\pi_{T[S]} R) \times_{S_1 \in T[S]} \pi_{e \in S} R(e)$$

i.e., the Cartesian product of the join results over all maximally connected components in $T[S]$.

**Example 3.3.** Continue the example in Example 3.2. The subjoin$^{12}$ of $S_1$ is defined as $\pi_{T[S_1]} R = R_1 \times R_3 \times R_7$ and that of $S_2$ is defined as $\pi_{T[S_2]} R = (R_0 \times R_1 \times R_3) \times R_7$.

From Example 3.3, we can see the difference between subjoin of $S$ and the join result of relations in $S$, or even the projection of final join results on attributes appearing in any relation of $S$, i.e.,

$$\pi_S Q(R) \subseteq \pi_{e \in S} R(e) \subseteq \pi(T, R, S)$$

In our following algorithm, subjoin is frequently used in server allocation for achieving maximum parallelism. In some sense, subjoin indicates the hardness of computing the join induced on a subset of relations by our generic algorithm.

### 3.2 Algorithm

We describe our generic algorithm for computing the result $Q(R)$ on the input join tree $T$. For a clean presentation, we delay its analysis to the next section, and now only focus on the algorithmic details and correctness.

Our algorithm chooses a fixed threshold $L$, whose value will be determined later. At first, we remove dangling tuples and reduce the join instance. If any relation $R(e)$ has input size smaller than $L$, we just broadcast $R(e)$ to all servers and discard $e$ from the query. The high-level idea is to recursively decompose the join into multiple subqueries, and apply different join strategy for each subquery.

**Base Case.** When there is only one relation, say $E = \{e\}$, just let all servers emit all tuples in $R(e)$ directly.

**General Case.** We start with an arbitrary leaf node $e_1$ in $T$ and denote its parent node as $e_0$. Let $x = e_1 \cap e_0$ be the set of join attributes between $e_1$ and $e_0$.

**Step 1: Compute data statistics.** For each assignment $a$ over attributes $x$, we compute its degrees in $R(e)$ and $R(e_0)$ respectively using the reduce-by-key primitive. An assignment $a$ over attributes $x$ is heavy if its degree in $R(e)$ is greater than $L$ and light otherwise. Denote the set of heavy assignments as

$$H(x, e_1) = \{a \in \text{dom}(x) : |\sigma_a \times_{e \in E - \{e_0, e_1\}} R(e)| \geq L\}$$

Tuples in $R(e_1)$ and $R(e_0)$ can be identified as heavy or light correspondingly, depending on their values on $x$ as heavy or light.

Moreover, for all light assignments over attributes $x$, we run the parallel-packing primitive to put them into $k = O\left(\frac{|R(e)|}{L}\right)$ groups $I_1, I_2, \cdots, I_k$, where the assignments in each group have total degree of $O(L)$ in $R(e_1)$.

**Step 2: Decompose the join query.** In this way, we can decompose the original join query into multiple subqueries:

$$Q_1(R) = \sigma_{x \in x} R(e) \times \sigma_{e \in E - \{e_0, e_1\}} \pi_{e \in S} R(e)$$

where $\sigma_{x = a}$ is for some $a \in H(x, e_1)$ or $x \in I_j$ for some $j \in \{1, 2, \cdots, k\}$. The subquery with predicate $x = a$ is denoted as heavy and $x \in I_j$ as light. Note that there are $O\left(\frac{|R(e)|}{L}\right)$ such subqueries. For simplicity, we just drop $x$ from $Q_j(R)$ and distinguish different subqueries by the instance $R$.

After removing dangling tuples, the heavy subquery derived by an assignment $a \in H(x, e_1)$ has input instance

$$R_a = \{x' : x' = x \cap e, a' = \pi_x a, e \in E\}$$

Note that after removing all attributes in $x = e_1 \cap e_0$, edge $e_1$ becomes isolated in the residual hypergraph, implied by Lemma 3.1.
In this way, computing the join result \( Q(R_a) \) is equivalent to computing the Cartesian product of tuples in \( \sigma_{e \in \mathcal{E}} R(e_1) \) and join results in the remaining relations, i.e.,
\[
Q(R_a) = (\sigma_{e \in \mathcal{E}} R(e_1)) \times (\sigma_{e \in \mathcal{E} - \{e_1\}} \sigma_{e \in \mathcal{E}} R(e))
\]
Similarly, the light subquery derived by group \( I_j \) has input instance
\[
R_j = (\sigma_{e \in \mathcal{E}_j} R(e) : e \in \{e_1, e_0\}) \cup \{R(e) : e \in \mathcal{E} - \{e_1, e_0\}\}
\]
whose join result is
\[
Q(R_j) = (\sigma_{e \in \mathcal{E}_j} R(e_0)) \Rightarrow (\sigma_{e \in \mathcal{E} - \{e_1, e_0\}} R(e))
\]
Note that all subqueries have disjoint results and their union is exactly the result of original join, i.e.,
\[
Q(\mathcal{R}) = \bigcup_{e \in \mathcal{E}} Q(R_e) \cup \bigcup_{j \in \{1, 2, \ldots, k\}} Q(R_j)
\]
thus the completeness is guaranteed.

**Step 3: Allocate servers.** We introduce a residual query \( Q_e = (V_e, E_y, \mathcal{E}_y) \) by removing all attributes in leaf \( e_1 \), where \( V_e = V - e_1 \) and \( E_y = \{e - e_1 : e \in \mathcal{E}\} \). For each heavy subquery with input instance \( \mathcal{R}_a \), we allocate
\[
p_a = \left[ \frac{|\sigma_{e \in \mathcal{E}_a} R(e_1)|}{L} \max_{S \subseteq \mathcal{E}_a} \left[ \frac{|T(R, \mathcal{R}_a, S)|}{L[S]} \right] \right]
\]
servers.

Let \( y = e_1 - e_0 \). We introduce a residual query \( Q_y = (V_y, E_y, \mathcal{E}_y) \) by removing all attributes in \( y \), where \( V_y = V - y \) and \( E_y = \{e - y : e \in \mathcal{E} - \{e_1\}\} \). Note that \( E_y \) only differs on the leaf \( e_1 \). Later in Step 4, we will see why leaf \( e_1 \) does not go into the recursion of residual query. For each light subquery with input instance \( \mathcal{R}_j \), we allocate
\[
p_j = \left[ \frac{|\sigma_{e \in \mathcal{E}_j} R(e_1)|}{L} \max_{S \subseteq \mathcal{E}_j} \left[ \frac{|T(R, \mathcal{R}_j, S)|}{L[S]} \right] \right]
\]
servers.

How to compute these \( p_a \)'s or \( p_j \)'s efficiently in parallel? Both boil down to computing a set of subjoin. For each \( S \subseteq \mathcal{E}_a \) or \( S \subseteq \mathcal{E}_j \), we use the optimization \( \max_{S \subseteq \mathcal{E}_a} \left[ \frac{|T(R, \mathcal{R}_a, S)|}{L[S]} \right] \) for \( \mathcal{R}_a \) over all heavy assignments or \( \max_{S \subseteq \mathcal{E}_j} \left[ \frac{|T(R, \mathcal{R}_j, S)|}{L[S]} \right] \) for \( \mathcal{R}_j \) over all light groups. By definition, computing these statistics can be captured by a free-connex join-aggregate query
\[
\sum_{V \subseteq \mathcal{E}} X_{S \subseteq \mathcal{E}} (T_S, S) \Rightarrow_{e \in S} R(e),
\]
where each tuple has weight annotation as 1. We invoke the algorithm in [16] to compute this query in \( O(1) \) rounds, whose result is in forms of \( t, w(t) \) for each assignment \( t \) in \( \text{dom}(x) \), with size bounded by \( O(|R(e_1)|) \). If \( a \in H(x, e_1) \), then \( |T(R, \mathcal{R}_a)| = w(a) \); otherwise, we run the reduce-by-key primitive to compute \( |T(R, \mathcal{R}_j, S)| = \sum_{e \in \mathcal{E}_j} w(a) \) for all light groups. As there are \( O(1) \) subsets of relations, this step can be done in \( O(1) \) rounds.

---

13The definitions of join-aggregate query and free-connex query are provided in Appendix A.4. In short, for join-aggregate query \( \sum Q \), if \( Q \) is acyclic and \( V - x \) is contained by one relation, this query is free-connex.

**Step 4: Compute all subqueries in parallel.** For heavy subquery, our strategy of computing \( Q(R_a) \) is to arrange \( p_a \) servers into a \( d_1 \times d_2 \) grid, where
\[
d_1 = \left[ \frac{|\sigma_{e \in \mathcal{E}_a} R(e_1)|}{L} \right], \quad d_2 = \max_{S \subseteq \mathcal{E}_a} \left[ \frac{|T(R, \mathcal{R}_a, S)|}{L[S]} \right]
\]
Each server is associated with a pair of coordinates \( (i, j) \) where \( i \in [d_1], j \in [d_2] \). For every \( j \in [d_2] \), the \( d_1 \) servers with coordinates \( (i, j) \) form a group to load all tuples in \( \sigma_{e \in \mathcal{E}_a} R(e_1) \) in a balanced way. Similarly, for every \( i \in [d_1] \), the \( d_2 \) servers with coordinates \( (i, j) \) form a group to compute \( Q_{e_i}(R_a) \) by running this whole algorithm recursively. At last, each server just emits the combination \( (1, t_1, t_2) \) for each input tuple \( t_1 \in \sigma_{e \in \mathcal{E}_a} R(e_1) \) and each join result \( t \in Q_{e_i}(R_a) \) by local computation.

**4 ANALYSIS OF ACYCLIC JOIN ALGORITHM**

To clarify the analysis for load complexity, we define the following quantity for each subset of relations \( S \subseteq \mathcal{E} \):
\[
\Psi(T, \mathcal{R}, S) = \frac{|T(R, \mathcal{R}_a)|}{L[S]}
\]
Intuitively, \( \Psi(T, \mathcal{R}, S) \) can be interpreted as the minimum number of servers required for computing the join of relations in \( S \) with load complexity \( O(L) \). We will prove the following result for our generic algorithm in Section 3:

**Theorem 1.** For a join query \( Q = (V, E) \) with join tree \( T \), an instance \( R \) and a pre-determined parameter \( L \), the result \( Q(R) \) can be computed using \( O(\sum_{S \subseteq \mathcal{E}} \Psi(T, \mathcal{R}, S)) \) servers in \( O(1) \) rounds with load complexity \( O(L) \).

**Proof.** We will prove it by induction on the size of \( Q \).

**Base Case.** When there is only one relation, say \( E = \{e\} \), emitting all tuples in \( R(e) \) with \( \frac{|R(e)|}{L} \) servers achieves a load of \( O(L) \), matching the bound in Theorem 1, since \( \Psi(T, \mathcal{R}, \{e\}) = \frac{|R(e)|}{L} \).

**General Case.** First, we know that all primitives can be computed using \( O(\max_{e \in \mathcal{E}} \frac{|R(e)|}{L}) \) servers in \( O(1) \) rounds with load complexity \( O(L) \). Moreover, server allocation in Step 4 can be computed using \( O(\max_{S \subseteq \mathcal{E}} \frac{|R(e_1)|}{L[S]}) \) servers in \( O(1) \) rounds with load \( O(L) \), implied by Theorem 6.2 in [16]. As
\[
\max_{e \in \mathcal{E}} \frac{|R(e)|}{L} = \max_{S \subseteq \mathcal{E} : |S| = 1} \Psi(T, \mathcal{R}, S),
\]
the complexity of these primitives matches the bound in Theorem 1.

Next we will analyze the complexity for computing heavy and light subqueries respectively.

**Complexity of computing heavy subqueries.** Below, we give an assumption made on a smaller join query \( Q_{e_1} \), which will be used by our inductive proof as hypothesis.

**Hypothesis 1.** For a join query \( Q_{e_1} = (V_{e_1}, E_{e_1}) \) with join tree \( T \), an input instance \( R_a \), and a pre-determined parameter \( L \), the
We distinguish each $O$ with load complexity $R$ below, we give an number of servers allocated to all groups is (big-Oh of)

$$L$$

Hypothesis 2. For a join query $Q_R$, used in this step. Note that the number of servers allocated over all $tuples in computing $Q$ can be computed using Cover or Pack: New Upper and Lower Bounds for Massively Parallel Joins

Recall that $e_1 \notin E$ by definition. We distinguish each $S \subseteq E$ into two cases: $e_0 \in S$ and $e_0 \notin S$.

If $e_0 \notin S$, this term induced on $S$ can be bounded by

$$\sum_{j \in J} \Psi(T, R_j, S) \leq \Psi(T, R, S) \cdot \frac{|R(e)|}{L} = \Psi(T, R, S \cup \{e_1\})$$

where the first inequality is implied by the fact that there are at most $O(\frac{|R(e)|}{L})$ light groups and the second one is implied by the fact that $e_1$ forms a single connected component in $T[S \cup \{e_1\}]$ when $e_0 \notin S$. Otherwise, $e_0 \in S$. This term induced on $S$ can be directly bounded by

$$\sum_{j \in J} \Psi(T, R_j, S) \leq \Psi(T, R, S).$$

Together, the number of servers allocated for handling light sub-queries is at most $O(\sum_{S \subseteq E} \Psi(T, R, S))$ implied by the following facts: (i) $\{e_1\} \in 2^E$; (ii) $\cap_{S \subseteq E} \{e_1\} \in 2^E$ for each $S \subseteq E$ but $e_0 \notin S$; (iii) $S \in 2^E$ for each $S \subseteq E$ and $e_0 \in S$, thus completing the induction proof for light subqueries.

Over all subqueries, the total number of servers allocated in total can be bounded by

$$\sum_{S \subseteq E} \Psi(T, R, S \cup \{e_1\}) + \sum_{S \subseteq E, e_0 \notin S} \Psi(T, R, S \cup \{e_1\})$$

$$+ \sum_{S \subseteq E, e_0 \in S} \Psi(T, R, S) + \Psi(T, R, \{e_1\}) \leq 4 \sum_{S \subseteq E} \Psi(T, R, S)$$

where the last inequality is implied by the following facts:

- $S \cup \{e_1\} \subseteq E$ for each $S \subseteq E$;
- $\{e_1\} \subseteq E$;
- $S \cup \{e_1\} \subseteq E$ for each $S \subseteq E$ but $e_0 \notin S$;
- $S \subseteq E$ for each $S \subseteq E$ and $e_0 \in S$;

thus completing the induction proof.

4.1 Choosing $L$

Theorem 1 displays a full trade-off between the number of servers available and the load complexity. A natural question arises, if we are only given $p$ servers, what’s the smallest load complexity that can be achieved for computing an acyclic join query in $O(1)$ rounds. We choose the value of $L$ as below:

$$L = \max_{S \subseteq E} \left(\frac{\Theta(T, R, S)}{p}\right)^{\frac{1}{n}},$$

where $S$ is taken over all subsets of $E$. It can be easily checked that for each $S \subseteq E$, $\Psi(T, R, S) \leq p$ holds, thus this is feasible.

As shown in Step 3 of our generic algorithm, such a value of $L$ can be computed through a join-aggregate query similarly using $p$ servers in $O(1)$ rounds with load complexity $O(1/p)$. Observe that

$$\sum_{e \in E} \frac{|R(e)|}{p} \leq |E| \cdot \max_{S \subseteq E: |S|=1} \left(\frac{\Theta(T, R, S)}{p}\right)^{\frac{1}{n}} = O(L)$$

Together, we come to the following result directly.
Theorem 2. For an acyclic join query \( Q = (V, E) \) with a join tree \( T \), and an instance \( R \), the result \( Q(R) \) can be computed using \( p \) servers in \( O(1) \) rounds with load \( O \left( \frac{\sum_{S \subseteq E} |S|}{p} \right) \).

4.2 Gap from worst-case optimality

So far we have obtained an algorithm for computing any acyclic joins running in \( O(1) \) rounds, whose load complexity is in terms of subjoins. However, we observe a gap between the result in Theorem 2 and our target \( O(N/p^{1/\rho}) \).

Example 4.1. Let's use the example query in Figure 4 for illustration. This join query has \( \rho = 6 \) by choosing \( \{e_1, e_2, e_3, e_4, e_5, e_7\} \) in the fractional edge covering, thus our target load complexity is \( O(N/p^{1/6}) \). Consider a hard instance constructed as below. There are \( N \) distinct values in the domain of attributes \( D, E, F, H, J, K, G \) and a single value in the domain of remaining attributes. Relation \( R_4(ABHJ) \) is a one-to-one mapping over attributes \( H, J \), and every remaining relation is a Cartesian product over its all attributes, containing \( N \) tuples in total. Also, this instance has its join size matching the AGM bound as \( O(N^5) \). On \( S_0 = \{e_1, e_2, e_3, e_5, e_6, e_7\} \), its subjoin has size as large as \( N^5 \) since \( |R_1 \bowtie R_2 \bowtie R_3 \bowtie R_0| = N^5 \) and \( |R_2 \bowtie R_0| = N^3 \). Thus, our generic algorithm compute the example join query on this hard instance with load complexity \( O(N/p^{1/\rho}) \), which is worse than the optimal by a factor of \( O(p^{1/\rho}) \).

Careful inspection reveals that not every subset \( \subseteq E \) appears in the cost formula of Theorem 2, depending on which recursive choices the algorithm makes while running. A key observation is that if there exists a pair of edges \( e, e' \in E \) such that \( e \subseteq e' \), relation \( R(e) \) can be reduced through a semi-join \( R(e') \bowtie R(e) \). In this way, \( e' \) won't go into the recursion, appearing together with other relations in the subjoins. For example, if the algorithm chooses to peel off \( e_1, e_2, e_3 \) sequentially, relation \( e_0 \) could be reduced since \( e_0 - e_1 \cup e_2 \cup e_3 \subseteq e_4 \). In this way, \( e_0 \) won’t appear together with any remaining relation of \( e_5, e_6, e_7 \), and \( S_3 \) does not contributes to the cost formula of Theorem 1.

This motivates us to adopt a fine-grained analysis of this generic algorithm, in terms of its recursive choices or peeling ordering. It is hopeful to achieve the worst-case optimality through some specific choices. We next give a characterization of such choices.

5 WORST-CASE OPTIMALITY OF ACYCLIC JOIN ALGORITHM

As mentioned, Theorem 2 gives a upper bound on the complexity of the algorithm proposed in Section 3, no matter which choice it makes in each recursive step. Next, we lower its cost further by restricting the algorithm’s choices with some rules, as shown in Definition 2. At last, we are able to show that any run of this non-deterministic algorithm is worst-case optimal, as long as it makes choices following these rules.

5.1 A good run

We first define \( S(E) \subseteq 2^E \) as the set of subsets of \( E \). The high-level idea of decreasing the complexity of Theorem 2 is by limiting the set of subjoins from \( 2^E \) to \( S(E) \). Abusing notations, we redefine the cross product between two sets of sets as \( X \times Y = \{S_1 \cup S_2 \subseteq S_1 \times S_2 \subseteq Y\} \).
where the first inequality is implied by the fact that \( e_0 \in S(E - \{e_1\}) \) and \( \{e_1\} \in S(E) \) from Lemma 5.1.

**Rule (3).** If there exists a leaf node \( e_k \in E \) whose parent node contains some unique attribute, the analysis is similar to the proof of Theorem 1, where hypothesis 1 and hypothesis 2 are replaced by hypothesis 3. The total number of servers required is \( \text{(big-Oh of)} \)

\[
\sum_{a} \sum_{S \subseteq S(E - \{e_1\})} \frac{|S|}{|S'|} \cdot \Psi(T, R_a, S)
\]

where the second inequality follows that the number of heavy assignments as well as the light group sizes are \( O(\frac{|S|}{|S'|}) \).

**Rule (4).** If there is an internal node \( e_0 \) without any unique attributes, and its children \( \Gamma(e_0) = \{e_1, e_2, \ldots, e_k\} \) are all leaves, we will prove by induction on \( k \). Let \( e' \) be the parent of \( e_0 \) in \( E \). As \( e_0 \) doesn’t contain any unique attribute, \( e_0 - e_1 \cup e_2 \cup \cdots \cup e_k \subseteq e' - e_1 \cup e_2 \cup \cdots \cup e_k \). In this way, \( S(E - \Gamma(e_0)) = S(E - \Gamma(e_0) - \{e_0\}) \cup \{e_0\}. \)

(4.1): \( k = 1 \). In this case, we first observe that \( S(E) \) in rule (4) can be simplified as

\[
S(E) = 2^{\{e_1\}} \cup \{\{e_1\} \cup S(E - \{e_1, e_0\}) \cup S(E - \{e_1\}) \}
\]

Following the proof of Theorem 1 by replacing hypothesis 1 and 2 are replaced with hypothesis 3, the total number of servers allocated for computing heavy subqueries is

\[
(2) = \sum_{a} \frac{|S_a| \cdot R(e_1)}{L} \cdot \Psi(T, R_a, S)
\]

It can be easily checked that for every \( S \in S(E - \{e_1\}) \),

\[
\sum_{a} \frac{|S_a| \cdot R(e_1)}{L} \cdot \Psi(T, R_a, S) \leq \Psi(T, R, S \cup \{e_1\})
\]

Recall that \( S(E - \{e_1\}) = \{\{e_1\}\} \cup S(E - \{e_1, e_0\}) \). We can expand \( \{\{e_1\}\} \times S(E - \{e_1\}) \) as

\[
\{\{e_1\}\} \times \{\{e_1\}\} \times S(E - \{e_1, e_0\}) \subseteq S(E)
\]

implied by (4.3) above. Thus, (2) \( \leq \sum_{S \subseteq S(E)} \Psi(T, R, S) \). Similarly, the number of servers allocated for computing light subqueries is

\[
(3) = \sum_{j} \sum_{S \subseteq S(E - \{e_1\})} \Psi(T, R_j, S)
\]

where each \( S \in S(E - \{e_0\}) \) is distinguished into two cases. With (4.3), both terms are captured by \( \sum_{S \subseteq S(E)} \Psi(T, R, S) \).

(4.2): \( k > 1 \). Next, we hold the following hypothesis for \( k - 1 \). Let \( E' = E - \{e_k\} \) and \( \Gamma'(e_0) = \{e_1, e_2, \ldots, e_{k-1}\} \).

\[
S(E') = 2^{\Gamma'(e_0)} \cup 2^{\Gamma'(e_0)} \times S(E' - \Gamma'(e_0)) - \{e_0\}
\]

In this case, we peel \( e_k \) off first and then handle the remaining query by applying rule (4) recursively.

Let \( a \) be the heavy assignment over attributes \( x = e_{k+1} \cap e_0 \). The number of servers allocated for heavy subqueries is

\[
\sum_{a} \frac{|S_a| \cdot R(e_k)}{L} \cdot \Psi(T, R_a, S)
\]

It can be easily checked that for any \( S \in S(E - \{e_k\}) \),

\[
\sum_{a} \frac{|S_a| \cdot R(e_k)}{L} \cdot \Psi(T, R_a, S) \leq \Psi(T, R, S \cup \{e_k\})
\]

Moreover, \( S \cup \{e_k\} \in S(E) \) implied by the following facts,

\[
\{\{e_k\}\} \times 2^{\Gamma'(e_0)} \cup \{e_0\} = 2^{\Gamma'(e_0)} \cup \{e_0\}.
\]

\[
\{\{e_k\}\} \times 2^{\Gamma'(e_0)} \times S(E' - \Gamma'(e_0)) - \{e_0\} = 2^{\Gamma'(e_0)} \times S(E' - \Gamma'(e_0)) - \{e_0\}
\]

Thus, the number of servers required is \( O(\sum_{S \subseteq S(E)} \Psi(T, R, S)) \).

Let \( l_1, l_2, \ldots, l_m \) be the light group defined over attributes \( x = e_{k+1} \cap e_0 \). The number of servers allocated for light subqueries is

\[
\sum_{j} \sum_{S \subseteq S(E - \{e_k\})} \Psi(T, R_j, S) \leq \sum_{S \subseteq S(E - \{e_k\}) \cup \{e_k\}} \Psi(T, R, S)
\]

Similarly, we distinguish every \( S \in S(E - \{e_k\}) \) into two cases. For any \( S \in S(E - \{e_k\}) \), there is \( S \in S(E) \). Moreover, for any \( S \in S(E - \{e_k\}) \) but \( e_0 \notin S \), there is \( S \cup \{e_k\} \in S(E) \) implied by the same facts as the heavy case. Thus, both two terms are captured by \( \sum_{S \subseteq S(E)} \Psi(T, R, S) \).

\[\square\]

### 5.2 Worst-case Optimality

We run the generic algorithm by making choices according to Definition 2, but using a different value of \( L \) defined as below:

\[
L = \max_{S \subseteq S(E)} \left[ \frac{|\Theta(T, R, S)|}{L|S|} \right]
\]

Since the query has constant complexity, the \( S(E) \) can be computed by a single server in \( O(1) \) rounds with \( O(L) \) load.

**Theorem 3.** For an acyclic join \( Q \) with a join tree \( T \) and an instance \( R \), the result \( Q(R) \) can be computed using \( p \) servers in \( O(1) \) rounds with load \( O \left( \frac{\Theta(T, R, S)}{p} \right) \).

Note that we have obtained an instance-dependent algorithm. On any acyclic join and an instance where each relation has at most \( N \) input tuples, such a complicated form has a clean form as stated in Theorem 4. Its proof is related to an important property of acyclic join that it always admits an integral solution for the optimal fractional edge covering number, with more details in Appendix A.1.

**Theorem 4.** For an \( \alpha \)-acyclic join query \( Q \) and an instance \( R \) where each relation contains at most \( N \) tuples, there is an algorithm computing \( Q(R) \) in \( O(1) \) rounds with load \( O \left( \frac{\Theta(T, R, S)}{p} \right) \), where \( p \) is the optimal fractional edge covering number of \( Q \), which is worst-optimal.
The proof of Theorem 4 is given in Appendix A.2. Note that the optimality is shown through a hard instance implied by the AGM bound, where each relation contains Θ(N) input tuples and the join size is as large as Θ(N^3). On the other hand, a more fine-grained analysis of the optimality of Theorem 3 in terms of arbitrary input sizes N(ε)'s would be interesting and challenging question. A similar fine-grained lower bound has been established in [15]. We won’t go into this direction further.

6 LOWER BOUNDS FOR CYCLIC JOINS

In this section, we first prove a lower bound as Ω(N̄(𝑝̄̂)) for the query Q on, as stated in Theorem 5, which is matched by the existing upper bound O(N̄(𝑝̄̂)) [19], thus being optimal. We then identify the class of degree-two joins, as well as the edge-packing-provable conditions, such that the edge-packing-dependent lower bound of Ω(N̄(𝑝̄̂)) can be proved for any degree-two join, as long as it satisfies the edge-packing-provable conditions, as stated in Theorem 6.

Theorem 5. For any N/log⁶ N ≥ p̄, there exists an instance R for Q on, where each relation has input tuples such that any tuple-based algorithm computing Q on in O(1) rounds must incur a load of Ω(N̄(𝑝̄̂)), which is optimal.

Theorem 6. On any edge-packing-provable degree-two join Q, for any N/log⁶ N ≥ p̄ with some constant c, there exists an instance R for Q on each relation has input tuples such that any tuple-based algorithm computing Q on in O(1) rounds must incur a load of Ω(N̄(𝑝̄̂)), where p̄ is the optimal fractional edge packing number.

The high-level idea of our lower bound proof is to show that with positive probability, an instance R for Q on can be constructed with bounded J(L), the maximum number of join results a server can produce, if it loads at most L tuples from each relation. We again resort to the counting argument that each join result must be emitted by at least one server. Setting p = J(L) = Ω(|Q on|) yields a lower bound on L.

6.1 □-Join

We next prove Theorem 5. Assume N ≥ p̄. Note that L ≥ N/p ≥ N̄(𝑝̄̂)/3 in this case. Our hard instance R is constructed as follows.

Hard Instance. Each one of the attributes A, B, C has N̄(𝑝̄̂)/3 distinct values and each of the attributes D, E, F has N̄(𝑝̄̂)/3 distinct values. Relations R1(A, B, C), R2(A, D), R4(B, E) and R5(C, F) are Cartesian products, each with exactly N̄(𝑝̄̂) tuples. Relation R3(D, E, F) is constructed in a probabilistic way. For R3(D, E, F), each combination (d, e, f) ∈ dom(D) × dom(E) × dom(F) has a probability of 1/N to form a tuple in R3(D, E, F). In this way, relation R3(D, E, F) have N̄(𝑝̄̂) tuples in expectation. The join result of this instance can be represented as the Cartesian product of R1(A, B, C) × R2(D, E, F). So this instance has input size 5N̄(𝑝̄̂) and output size N̄(𝑝̄̂)/3 in expectation. By the Chernoff bound, the probability that the input size and output size deviate from their expectation by more than a constant factor is at most exp(-Ω(N)).

Step 1: Make a reasonable restriction on loading tuples.

To bound J(L), we first argue that on any instance constructed as above, we can limit the choice of the L tuples loaded from R1(A, B, C), R3(A, D), R4(B, E) and R5(C, F) by any server in the form of LA×LB×LC, LA×LD, LB×LD, and LC×LF for LA ⊆ dom(A), LB ⊆ dom(B), LC ⊆ dom(C), LF ⊆ dom(E) and LF ⊆ dom(F), i.e., the algorithm should load tuples from R1(A, B, C), R3(A, D), R4(B, E) and R5(C, F) in the form of Cartesian product. More precisely, we first prove this result for attributes A as stated in Lemma 6.1. The similar argument can be applied for attributes B and C.

Lemma 6.1. Restricting loading tuples from R3(A, D) in a form of LA×LD and those from R1(A, B, C) in a form of LA×LB where LB are the assignments over attributes B, C, will not make J(L) smaller by more than a constant factor.

Proof. Suppose a server has loaded L tuples from R3(D, E, F), R4(B, E) and R5(C, F). Then the server needs to decide which L tuples from R1(A, B, C) and R3(A, D) to load to maximize the number of join results produced. This is a combinatorial optimization problem that can be formulated as an integer program (IP). Introduce a variable xabc for each triple a ∈ dom(A), b ∈ dom(B), c ∈ dom(C), yad for each pair a ∈ dom(A), d ∈ dom(D). Let Idef = 1 if tuple (d, e, f) ∈ R3(D, E, F) is loaded by the server, and 0 otherwise. The similar definition applies for Ibe and If. Then IP1 below defines this optimization problem, where a always ranges over dom(A), b over dom(B), c over dom(C), d over dom(D), e over dom(E), f over dom(F) unless specified otherwise.

(IP1) \[
\begin{align*}
\text{max} & \sum_{a,b,c,d,e,f} I_{def} \cdot I_{be} \cdot I_{cf} \cdot x_{abc} \cdot y_{ad} \\
\text{subject to} & \sum_{abc} x_{abc} \cdot y_{ad} \leq L \\
& I_{def}, I_{be}, I_{cf} \cdot x_{abc} \cdot y_{ad} \in \{0, 1\}, \forall a, b, c, d, e, f
\end{align*}
\]

However, it seems very difficult to dig out any structural property of the optimal solution of IP1. Instead, we introduce a relaxed version of IP1, shown as IP3 below.

(IP3) \[
\begin{align*}
\text{max} & \sum_a \Lambda(w_a) \\
\text{subject to} & \sum_a w_a \leq 2L \\
& w_a \in \{1, 2, \cdots, L\}, \forall a
\end{align*}
\]

Note that IP3 uses a function \(\Lambda(w)\), which denotes the optimal solution of IP2 defined as below:

(IP2) \[
\begin{align*}
\text{max} & \sum_{b,c,d,e,f} I_{def} \cdot I_{be} \cdot I_{cf} \cdot x_{abc} \cdot y_{ad} \\
\text{subject to} & \sum_{bc} x_{abc} \cdot y_{ad} \leq w \\
& I_{def}, I_{be}, I_{cf} \cdot x_{abc} \cdot y_{ad} \in \{0, 1\}, \forall a, b, c, d, e, f
\end{align*}
\]

IP2 is parameterized by w and a, which finds the maximum number of join results that can be formed by tuples loaded from R3(D, E, F), R4(B, E) and R5(C, F), subject to the constraint that at most w tuples containing value a are loaded from R1(A, B, C) and R3(A, D).

Since all values in the domain of attribute A are structurally equivalent, the optimal solution of IP2 does not depend on the
particular choice of $a$, which is why we write the optimal solution of $IP_2$ as $\Delta(w)$. Also, it is obvious that $\Delta(.)$ is a non-decreasing function. Then, $IP_3$ tries to find the optimal allocation of the $L$ tuples to different values $a \in \text{dom}(A)$ so as to maximize the total number of join results formed. Let the optimal solutions of $IP_1$, $IP_3$ be $OPT_1$, $OPT_3$, respectively. Because $IP_3$ only restricts the server to load at most $2L$ tuples from $R_1(A, B, C)$ and $R_3(A, D)$ in total, any feasible solution to $IP_3$ is also a feasible solution to $IP_3$, so $OPT_1 \leq OPT_3$. Next we construct a feasible solution of $IP_3$ with the Cartesian product restriction above, and show that it is within a constant factor from $OPT_3$, hence $OPT_1$.

Set $w^* = \arg \max_{L \leq w \leq \frac{L}{\alpha}} \frac{L}{w} \cdot \Delta(w)$. We choose $\frac{L}{w^*}$ distinct values arbitrarily from $\text{dom}(A)$ and allocate $w^*$ tuples to each such $a$. For each $a$, we use the optimal solution of $IP_2$ to load the $w^*$ tuples to load from $R_1(A, B, C)$ and $R_3(A, D)$. Note that the optimal solution is the same for every $a$, so each $a$ choose the same sets of $(b, c)$'s and $d$'s. Thus, this feasible solution loads tuples from $R_1(A, B, C)$ and $R_3(A, D)$ in the form of Cartesian products. The number of join results that can be produced is $W = \frac{L}{w^*} \cdot \Delta(w^*)$. We show that $W$ is a constant-factor approximation of $OPT_3$, as below.

**Lemma 6.2.** $W \geq \frac{1}{2}OPT_3 \geq \frac{1}{2}OPT_1$.

The proof of Lemma 6.2 is deferred to Appendix A.3, thus completing our whole proof. □

Note that Lemma 6.1 implies that $L_{AB} = L_A \times L_B$ and $L_{AC} = L_A \times L_C$ in relation $R_1(A, B, C)$. Applying a similar argument to attribute $B$, we get $L_{BC} = L_B \times L_C$. Together, we come to $L_{ABC} = L_A \times L_B \times L_C$, i.e., tuples in relation $R_1(A, B, C)$ should be loaded in form of Cartesian product over all attributes.

**Step 2: Prove a upper bound on $f(L)$.**

Next, we show that with positive probability (actually high probability), we obtain an instance on which $f(L)$ is bounded, no matter which $L$ tuples are loaded. By the analysis above, we need only to consider the case where tuples from $R_1(A, B, C)$, $R_2(A, D)$, $R_2(B, E)$, $R_3(C, F)$ are loaded form of Cartesian products. Denote the number of distinct values in $\text{dom}(A)$, $\text{dom}(B)$ loaded by the server as $\alpha$, $\beta$ respectively. The number of distinct values in $\text{dom}(C)$, $\text{dom}(D)$, $\text{dom}(E)$, $\text{dom}(F)$ loaded by the server are $\frac{L}{\alpha \beta}$, $\frac{L}{\alpha}$, $\frac{L}{\beta}$, $\frac{L}{\alpha \beta \alpha}$. Moreover, $\frac{L}{\alpha \beta} \leq \alpha, \beta \leq N^{1/3}$.

There are $L^3$ combinations in terms of $(a, b, c, d, e, f)$ in total. Each one is a valid join result if and only if $(d, e, f) \in R_2(D, E, F)$, which happens with probability $\frac{L^3}{N^3}$. By the linearity of expectation, the expected number of join results can be produced by the $L$ tuples is $\frac{L^3}{N^3}$. More careful inspection reveals that the $L^3$ combinations are not independent; instead we can divide them into $L^2$ independent groups where each one is associated to one distinct triple $(d, e, f)$. Implied by the Chernoff bound, the probability that this server produces more than $2 \cdot \frac{L^3}{N^3}$ join results is at most $\exp(-\Omega(\frac{L^3}{N^3}))$.

For attribute $A$, there are $\binom{N^{1/3}}{L^{1/3}} = O(N^{-\alpha})$ choices of loading $\alpha$ distinct values from $\text{dom}(A)$. Similar argument can be applied to attributes $B, C, D, E, F$. Over all values of $\alpha, \beta$, the number of choices in total is

$$\sum_{\alpha=\frac{L}{N^{1/3}}}^{N^{1/3}} \sum_{\beta=\frac{L}{N^{1/3}}}^{N^{1/3}} \delta^{\frac{1}{3}} + \frac{4}{5} \cdot \frac{2}{3} \cdot \frac{1}{2} + \frac{1}{2} = \exp\left(\frac{L}{\alpha} + \frac{L}{\beta} \right) = \exp\left(\frac{L}{\alpha} + \frac{L}{\beta} \right)$$

By the union bound, the probability that any of the choice produces more than $2 \cdot \frac{L^3}{N^3}$ join results is at most

$$\exp(-\Omega(\frac{L^3}{N^3}))$$

which is exponentially small if

$$\frac{L^2}{N} \geq c_1 \cdot N^{\frac{1}{3}} \cdot \log N$$

for some sufficiently large constant $c_1$. Rearranging, we get

$$L^2 \geq c_1 \cdot N^{\frac{1}{3}} \cdot \log N$$

We know that $L = \Omega(\frac{N^3}{p})$, so this is true as long as

$$\frac{N}{c_1} \geq L^2 \geq N^{\frac{1}{3}} \cdot \log N$$

for some sufficiently large constant $c_1$, or $N^{\log 6} N \geq c_2 \cdot p^6$.

**Step 3: Apply counting argument.**

So far, we have shown that with exponentially high probability each server produces no more than $2 \cdot \frac{L^3}{N^3}$ join results in each round. Over $p$ servers, the total number of join results that can be produced in $O(1)$ rounds is $O(\frac{L^3}{N^3})$. Each of the $N^2$ join results must be emitted at least once, so we will have $p \cdot \frac{L^3}{N^3} \geq N^2$, i.e., $L \geq N/p^{1/3}$. We have completed the whole proof for Theorem 5.

**6.2 Degree-two Joins**

Our lower bound proof for the $\sqcap$-join can be extended to a larger class of join queries, named as degree-two join, where every vertex appears in exactly two edges. As mentioned, degree-two joins enjoy several nice properties, as stated in Lemma 6.3. The proof of Lemma 6.3 is given in Appendix A.6.

**Lemma 6.3.** For any reduced degree-two join $Q = (\mathcal{V}, E)$, the following holds: (1) $\tau^* \geq \frac{|E|}{2} \geq \rho^*$; (2) $\tau^* + \rho^* = |E|$; (3) The optimal fractional edge packing/covering admits half-integral solution; (4) if there exists no odd-length cycle\(^{14}\), the optimal fractional edge packing/covering admits integral solution.

However, not all degree-two joins fit for the lower bound framework, two additional conditions are captured in Definition 6.4. Before describing the conditions, we introduce some terminologies and notions first. In a hypergraph $Q = (\mathcal{V}, E)$, let $\Gamma(e)$ be the set of neighbors of edge $e \in E$, i.e., $\Gamma(e) = \{f \in E : e \cap f \neq \emptyset\}$. A fractional vertex covering for $Q = (\mathcal{V}, E)$ is a mapping $x$ from $\mathcal{V}$ to $[0, +\infty)$ such that $\sum_{x \in \mathcal{V}} x_v \geq 1$ holds for each edge $e \in E$; and the optimal solution is to minimize the quantity $\sum_{x \in \mathcal{V}} x_v$. In addition, a vertex covering $x$ is constant-small if $\max_x x_v \leq 1 - \epsilon$ for some constant $0 < \epsilon < 1$.

\(^{14}\)A cycle $(\mathcal{V}, E)$ is defined as $\mathcal{V} = \{v_1, v_2, \cdots, v_n\} \subseteq \mathcal{V}$ and $\mathcal{E} = \{e_i = (v_{i}, v_{i+1 \mod n}) : i \in [1, 2, \cdots, n]\}$. The length of a cycle $(\mathcal{V}, E)$ is defined as $|\mathcal{E}|$. 
Definition 6.4. A degree-two join $Q = (V, E)$ is edge-packing-provable if (1) it is reduced; (2) there is no odd-length cycle; (3) there exists an optimal fractional constant-small vertex covering $x$, such that $|\Gamma(e) \cap E'| \leq 1$ for every $e \in E$, where $E' = \{ e \in E : \sum_{\alpha \in E} x_\alpha > 1 \}$. Note that $Q_\exists$ is an edge-packing-provable degree-two join. Obviously, there is no odd-length cycle, and a valid vertex covering $x$ is $x_A = x_B = x_C = \frac{3}{2}$ and $x_D = x_E = x_F = \frac{1}{2}$, which is also used in the lower bound proof of Theorem 5. Some other examples of edge-packing-provable degree-two joins are given in Figure 5.

Figure 5: Examples of edge-packing-provable joins.

The detailed proof of Theorem 6 is deferred to Appendix A.7. Here, we only give some intuition why the three conditions can be put together for generalizing this framework to degree-two joins.

In Definition 6.4, if there is no odd-length cycle in $Q$, it admits integral optimal edge packing $\tau^*$ and covering $\rho^*$, implied by Lemma 6.3. More specifically, there exists a partition $(E_\alpha, E_\beta)$ of $E$: $E_\alpha = \{ e \in E : \rho^*(e) = 1 \}$ and $E_\beta = \{ e \in E : \rho^*(e) = 0 \}$, for example, $E_\alpha = \{ e_1, e_2 \}$ and $E_\beta = \{ e_3, e_4, e_5 \}$ in $Q_\exists$. Moreover, all edges in $E_\alpha$ are vertex-disjoint, as well as edges in $E_\beta$, due to the fact that each vertex appears in at most two edges.

Consider any optimal fractional vertex covering $x$ satisfying (2) in Definition 6.4. Note that $x$ defines another partition $(E', E'')$ of $E$: $E' = \{ e \in E : \sum_{\alpha \in E} x_\alpha > 1 \}$ and $E'' = \{ e \in E : \sum_{\alpha \in E} x_\alpha = 1 \}$, for example, $E' = \{ e_2 \}$ and $E'' = \{ e_1, e_3, e_4, e_5 \}$ in $Q_\exists$. Note that the fractional edge packing and vertex covering are prime-dual. Implied by the slackness theorem, $E' \subseteq E_\alpha$. Edges in $E'$ are also vertex-disjoint. The hard instance is constructed by $x$.

More specifically, the domain of each attribute $x$ contains $N^{x_\alpha}$ distinct values. Relations in $E''$ are deterministically constructed as Cartesian products, containing $N$ tuples each, while those in $E'$ are probabilistically constructed. As $|\Gamma(e) \cap E'| \leq 1$ holds for every $e \in E$, each edge in $E' \cap E''$ derives a connected components $C(e) = \{ e' \in E'' : e' \cap e \neq \emptyset \}$. More importantly, $C(e_1) \cap C(e_2) = \emptyset$ for any pair of $e_1, e_2 \in E'$. We then apply a similar argument for $Q_\exists$ to each such component.

At last, the rationale behind the notion of “constant-small” is used to prove a upper bound on $|J(L)|$ with exponentially high probability; and more details can be found in Appendix A.7.

Remark. We only give a sufficient condition in Theorem 6, and several questions remain to be answered, for example, (1) what is a complete characterization of cyclic queries on which our framework can be applied, (2) is there any matching upper bound on the degree-two joins. It is still unclear whether $\rho^* = \tau^*$ holds for a degree-two join satisfying edge-packing-provable conditions. If this is the case, then the lower bound $\Omega(\frac{N}{p^{\alpha}})$ will be matched by the existing one-round algorithm [19].

REFERENCES

A.1 Properties of acyclic joins

An equivalent definition for $\alpha$-acyclicity is based on the GYO reduction [1]: (1) if there is a vertex $v \in V$ only appearing in edge $e$, then remove $v$ from $e$; (2) if there is a pair of edges $e, e' \in E$ such that $e \subseteq e'$, then remove $e$ from $E$. A join query $Q = (V, E)$ is $\alpha$-acyclic if the GYO reduction results in an empty hypergraph. We show some nice properties for acyclic join Lemma A.2, A.1, and A.3.

For short, we use "acyclic" to denote \( \alpha \)-acyclic below.

**Lemma A.1.** For any acyclic join $Q = (V, E)$, the residual join $Q_x = (V - x, E_x)$ is also acyclic, where $E_x = \{e - x : e \in E\}$.

**Proof.** Let $T$ be the join tree of $Q$, such that (1) there is a one-to-one correspondence between edges in $\mathcal{E}$ and nodes in $T$; (2) for any attribute $v \in V$, all nodes containing $v$ form a connected subtree. We derive another tree $T'$ by removing attributes $x$ from each node in $T$. It can be easily checked that $T'$ is a valid join tree for $Q_x$, thus $Q_x$ is acyclic. \( \Box \)

**Lemma A.2.** Acyclic join has integral optimal edge covering.

**Proof.** Let $\rho^*$ be the optimal edge covering for hypergraph $Q = (V, E)$. An edge cover of $Q$ can be obtained along with its GYO reduction. More specifically, if $Q$ is emptyset, we set $\rho^* = 0$. In general, we apply the following two procedures: (1) If attribute $v \in V$ only appears in $e$, then assign $\rho^*(e) = 1$ and remove all attributes in $e$ from $V'$; (2) if $e_1, e_2 \in E$ are distinct edges such that $e_1 \subseteq e_2$, then assign $\rho^*(e_1) = 0$ and remove $e$ from $E$.

Next we will prove its optimality. The base case is trivial. In general, we prove it for these two cases separately.

If $Q$ is reduced through (1), let $Q_x = (V - e, E_x)$ be the residual join by removing attributes in $e$ from all relations in $E$. By hypothesis, let $\rho^*(Q_x)$ be the integral optimal edge covering of $Q_x$. Note that $\rho^*(Q) = \rho^*(Q_x) + 1$, which is optimal since any edge cover require to assign $\rho^*(e) = 1$ to cover attribute $v$. Moreover, $\rho^*(Q)$ also admits integral optimal edge covering, since $\rho^*(Q_x)$ admits integral optimal edge covering.

If $Q$ is reduced through step (2), let $Q' = (V, E - \{e\})$ be the residual join by removing edge $e$. Obviously, $Q'$ is also acyclic. By hypothesis, let $\rho^*(Q')$ be the integral optimal edge covering of $Q'$. Note that $\rho^*(Q) = \rho^*(Q')$, which is optimal since we can always shift any weight assigned to $e$ to $e'$ while maintaining its optimality. Thus, $\rho^*(Q)$ is an optimal integral edge covering for $Q$. \( \Box \)

We next introduce the berge-acyclic joins. For a join query $Q = (V, E)$, consider the bipartite graph $G$, in which $V$ corresponds to vertices on one side and $E$ to vertices on the other side. There is an edge between $v \in V$ and $e \in E$ if $v \in e$. Then the hypergraph $(V, E)$ is berge-acyclic if this bipartite graph is acyclic. This notion of acyclicity preserves many natural properties in ordinary acyclic graphs. For example, there is only one path between any two vertices $u, v \in V$, and any subgraph of $(V, E)$ is still acyclic. Note that this definition of berge-acyclicity does not allow two relations to have two or more common attributes. But if these attributes always appear together in any relation, then they can be simply considered as one "combined" attribute.

**Lemma A.3.** For any reduced berge-acyclic join $Q$, $\tau^* \leq \rho^*$ where $\tau^*$, $\rho^*$ are the optimal fractional edge packing and covering number of $Q$ respectively.

**Proof.** For a berge-acyclic join $Q = (V, E)$, we can always find an internal node $e_0$ such that its children $\Gamma(e_0) = \{e_1, e_2, \cdots, e_k\}$ are all leaves. Note that $e \cap (e_1 \cup e_2 \cup \cdots \cup e_k) = \emptyset$ holds for every $e \in E - \{e_0\} - \Gamma(e_0)$.

If $e_0$ contains unique attributes, let $Q'$ be the residual query by removing all edges in $\Gamma(e_0)$. By hypothesis, $\rho^*(Q') \geq \tau^*(Q')$ since $Q'$ is also a reduced berge-acyclic join. Note that $\rho^*(Q) = \rho^*(Q') + k$ and $\tau^*(Q) \leq \tau^*(Q') + k$. Thus, $\rho^*(Q) \geq \tau^*(Q)$.

Otherwise, let $Q'$ be the residual query by removing all edges in $\{e_0\} \cup \Gamma(e_0)$. By hypothesis, $\rho^*(Q') \geq \tau^*(Q')$ since $Q'$ is also a reduced berge-acyclic join. Note that $\rho^*(Q) = \rho^*(Q') + k$ and $\tau^*(Q) \leq \tau^*(Q') + k$. Thus, $\rho^*(Q) \geq \tau^*(Q)$. \( \Box \)

A.2 Proof of Theorem 4

**Proof.** For simplicity, we introduce the function $\rho^*(E)$ to denote the optimal fractional edge covering number of the hypergraph induced by edges $E$. It suffices to show $$(\frac{|\Theta(T, R, S)|}{p})^{\frac{1}{|S|}} \leq N/p^{1/\rho^*}$$ for any subset $S \in S(E)$, i.e., $$|\Theta(T, R, S)| \leq N^{|S|} \cdot p^{1 - |S|/\rho^*(E)}.$$ We will prove it by induction on the size of $E$.

The base case when the query has only one relation is trivial. The case when there exists $e, e' \in E$ such that $e \subseteq e'$ is also easy. Any $S$ returned by $S(E)$ must fall into one of the two cases.

Case (2.1). If $S = \{e\}$, $|\Theta(T, R, S)| = |R(e)| \leq N \leq N \cdot p^{1 - |S|/\rho^*(E)}$ holds since $\rho^*(E) \geq 1$.

Case (2.2) If $S \in S(E - \{e\})$, we can bound it as $|\Theta(T, R, S)| \leq N^{|S|} \cdot p^{1 - |S|/\rho^*(E)} = N^{|S|} \cdot p^{1 - |S|/\rho^*(E)}$ where the first inequality is implied by the hypothesis on $S(E - \{e\})$ and the second one is by $\rho^*(E) = \rho^*(E - \{e\})$.

Next we consider the case when the algorithm peels off an leaf node $e'$ with its parent node $e$, where $e$ contains unique attributes. Denote $\rho^*(e)$ as the optimal edge covering function on the input hypergraph. We have $\rho^*(E) = \rho^*(E - \{e'\}) + 1$. Any $S$ returned by $S(E)$ must fall into one of the following three cases.

Case (3.1): If $S = \{e'\}$, this case is similar to (2.1).

Case (3.2): If $S \in S(E - \{e\})$, we can bound it as $|\Theta(T, R, S)| \leq N^{|S|} \cdot p^{1 - |S|/\rho^*(E)} \leq N^{|S|} \cdot p^{1 - |S|/\rho^*(E)}$ where the first inequality is implied by the hypothesis.

Case (3.3): If $S = S' + \{e\}$ for $S' \in S(E - \{e\})$, we further distinguish it into two more cases.

Case (3.3.1): If $|S'| \leq \rho^*(E - \{e_1\})$, we can bound it as $|\Theta(T, R, S)| \leq |R(e')| \cdot |\Theta(T, R, S')| \leq N \cdot N^{|S'|} = N^{|S'|} \cdot p^{1 - |S'|/\rho^*(E)}$ where the last inequality is implied by that fact that $|S| = |S'| + 1 < \rho^*(E - \{e_1\}) + 1 \leq \rho^*(E)$;
We can bound the subjoin on where the last inequality is implied by the fact that
\[ \rho (\mathcal{E} - \{e_1\}) \leq \rho (\mathcal{E}) \]
and where the rationale behinds the last inequality is that there are at most \( N^{1/3} \) values in \( A^*_1 \) and there is \( \Delta \left( \frac{L}{N^{1/3}} \right) \geq \Delta(w_0) \) for each \( a \in A^*_1 \) by the non-decreasing property of \( \Delta(.) \).

Combining the two parts for the optimal solution \( A^* \), our alternative solution loads at most 2L tuples from \( R_1(A, B, C) \) and \( R_3(A, D) \), and can report at least \( \frac{1}{2} \cdot OPT_3 \) join results.

\[ w(t) := \omega_{\{e\} \in R(e), \pi_1=\pi_2=\pi_{e} \cdot w(t_0) \} \]

In plain language, a join-aggregate query first computes the join \( Q(R) \) and the annotation of each join result, which is the \( \omega \)-aggregate of the tuples comprising the join result. Then it partitions \( Q(R) \) into groups by their projection on \( y \). Finally, for each group, it computes the \( \omega \)-aggregate of the annotations of the join results.

Many queries can be formulated as special join-aggregate queries. For example, if we take \( \mathbb{R} \) to be the domain of integers, \( \omega \) to be addition, \( \xi \) to be multiplication, and set \( w(t) = 1 \) for all \( t \), then it becomes the \( \text{COUNT} (+) \text{ GROUP} \) query; in particular, if \( y = 0 \), the query computes \( |Q(R)| \).

A join-project query \( \pi_y Q(R) \), also known as a conjunctive query, is a special join-aggregate query by discounting the annotations.

With respect to join-aggregate queries, free-connex queries [S] are an important subclass. To define a free-connex query, we introduce the notion of a width-1 GHD, which can be considered as a generalized join tree. A width-1 GHD of a hypergraph \( Q = (V, E) \) is a tree \( T \), where each node \( u \in T \) is a subset of \( V \), such that (1) for each attribute \( x \in V \), the nodes containing \( x \) are connected in \( T \); (2) for each hyperedge \( e \in E \), there exists a node \( u \in T \) such that \( e \subseteq u \); and (3) for each node \( u \in T \), there is a hyperedge \( e' \in E \) such that \( u \subseteq e' \).

Given a set of output attributes \( y, T \) is free-connex if there is a subset of connected nodes of \( T \), denoted as \( T' \) (such a \( T' \) is said to be a connex subset), such that \( y = \bigcup_{u \in T'} u \). A join-aggregate query \( Q_y \) is free-connex if it has a free-connex width-1 GHD.

A.4 Background knowledge for Join-Aggregate Query and Free-connex Query

We consider join-aggregate queries over annotated relations [10, 17] with one semiring. Let \((\mathbb{R}, \oplus, \otimes)\) be a commutative semiring. We assume that every tuple \( t \) is associated with an annotation \( w(t) \in \mathbb{R} \).

The annotation of a join result \( t \in Q(R) \) is

\[ w(t) := \omega_{\{e\} \in R(e), \pi_1=\pi_2=\pi_{e} \cdot w(t_0) \} \]

Let \( y \subseteq V \) be a set of output attributes (a.k.a. free variables) and \( \bar{y} = V - y \) the non-output attributes (a.k.a. bound variables). A join-aggregate query \( Q_y(\mathcal{R}) \) asks us to compute \( \omega_{y} Q(\mathcal{R}) = \{ (t_y, w(t_y)) : t_y \in \pi_y Q(\mathcal{R}), w(t_y) = \omega_{\{e\} \in R(e), \pi_1=\pi_2=\pi_{e} \cdot w(t_0) \} \} \).

In plain language, a join-aggregate query first computes the join \( Q(\mathcal{R}) \) and the annotation of each join result, which is the \( \omega \)-aggregate of the tuples comprising the join result. Then it partitions \( Q(\mathcal{R}) \) into groups by their projection on \( y \). Finally, for each group, it computes the \( \omega \)-aggregate of the annotations of the join results.

Many queries can be formulated as special join-aggregate queries. For example, if we take \( \mathbb{R} \) to be the domain of integers, \( \omega \) to be addition, \( \xi \) to be multiplication, and set \( w(t) = 1 \) for all \( t \), then it becomes the \( \text{COUNT} (+) \text{ GROUP} \) query; in particular, if \( y = 0 \), the query computes \( |Q(\mathcal{R})| \).

A join-project query \( \pi_y Q(\mathcal{R}) \), also known as a conjunctive query, is a special join-aggregate query by discounting the annotations.

A.5 Proof of Lemma 5.2

Proof. We first show the correctness of \( S(\mathcal{E}) \) defined above, i.e., any acyclic join query \( Q(\mathcal{V'}, \mathcal{E}) \) can be reduced to the base case by three different rules. This can be easily argued as follows. Assume \(|E| > 1\); there exists no \( e, e' \in \mathcal{E} \) such that \( e \subseteq e' \); and for each leaf
node $e'$ with its parent node $e$, $e$ doesn’t contain unique attributes. Under this circumstances, we can identify an internal node $e$ whose children are all leaves. Such a node can always be found since if deleting all leaf nodes in $T$, there must exists some internal node becoming a new leaf node. Moreover, $e$ doesn’t contain unique attributes by assumption. So, we can reduce $E$ by the last rule. □

A.6 Proof of Lemma 6.3

The proof directly follows the fact that the dual of any reduced degree-two join is an reduced binary-relation join [18, 24]. We can show more details for each property in Lemma 6.3.

For (1), assume $f$ be the mapping from $E$ to $[0, +\infty)$. Let $f(e) = \frac{1}{\tau}$ for every $e \in E$. In the reduced degree-two join $Q$, each vertex appears in exactly two hyperedges. In this way, $f$ is both a valid fractional edge packing and edge covering. Implied by the maximization of fractional edge packing and minimization of fractional edge covering, we get $\tau^* \geq \frac{1}{f} \geq \rho^*$.

For (2), let $f$ be a valid fractional edge packing for $Q$. It can be easily checked that $g = \{1 - f(e) : e \in E\}$ is a valid fractional edge covering for $Q$. Thus, we get $\rho^* = |E| - \tau^*$.

A similar result has been proved for optimal fractional vertex covering for an ordinary graph [21], that the vertex packing for an ordinary graph admits half-integral solution, and the set of vertices with value $\frac{1}{2}$ form a set of vertex-disjoint odd-length cycles. Thus, (3) and (4) follow.

A.7 Proof of Theorem 6

Before diving into the proof of Theorem 6, we point out several important properties for degree-two joins satisfying edge-packing-provable conditions.

If there is no odd-length cycle in $Q$, it admits integral optimal edge packing $\tau^*$ and covering $\rho^*$, implied by Lemma 6.3. More specifically, there exists a partition $(E_\alpha, E_\beta)$ of $E$ where $\alpha = \{e \in E : \rho^*(e) = 1, \tau^*(e) = 0\}$ and $\beta = \{e \in E : \rho^*(e) = 0, \tau^*(e) = 1\}$. Consider any vertex $v$ incident to two edges $e, e'$. There must be $e \in E_\alpha$, $e' \in E_\beta$, or $e \in E_\beta, e' \in E_\alpha$. This also implies that all edges in $E_\alpha$ are vertex-disjoint, as well as edges in $E_\beta$.

Let $x$ be an optimal fractional edge covering for $Q$. An edge $e$ is denoted as deterministic if $\sum_{v \in \epsilon e} x_v > 1$, and probabilistic otherwise. Let $E'$ be the set of probabilistic edges, i.e. $\{e \in E : \sum_{v \in \epsilon e} x_v > 1\}$. Note that vertex covering and edge packing are prime-dual problems. The following result is directly implied by the complementary slackness.

Lemma A.4. Let $E' = \{e \in E : \sum_{v \in \epsilon e} x_v > 1\}$. For any $e \in E'$, $\tau^*(e) = 0$ and $\rho^*(e) = 1$.

For each edge $e \in E$, let $Y(e)$ be the set of vertices appearing in the neighbor of $e$, i.e., $Y(e) = \cup_{e' \in \epsilon e} (e') - e$. Two nice properties on the edges in $E'$ are stated in Lemma A.5 and A.6.

Lemma A.5. Let $E' = \{e \in E : \sum_{v \in \epsilon e} x_v > 1\}$. For any edge $e \in E'$, $\sum_{v \in \epsilon e} x_v + \sum_{v \in \epsilon e} y_v = |\Gamma(e)|$.

Proof. As $\tau^*(e) = 0$ and $\rho^*(e) = 1$, we have $\tau^*(e') = 1$ and $\rho^*(e') = 0$ for every edge $e' \in \Gamma(e)$. This also implies that for any pair of edges $e_1, e_2$ in $\Gamma(e)$, $e_1 \cap e_2 = \emptyset$. Moreover, each edge $e' \in \Gamma(e)$ is deterministic implied by Lemma A.4, thus $\sum_{v \in \epsilon e} x_v = 1$.

We also observe that $e \cup \{e' \in \epsilon e' : e' \neq e\} = \cup_{e' \in \epsilon e} (e')$. Thus, the left-hand-side of the target equation can be rewritten as $\sum_{e \in E} x_e = \sum_{e' \in E} \sum_{e \in \epsilon e'} x_e = |\Gamma(e)|$ thus yielding the desired result. □

Lemma A.6. Let $E' = \{e \in E : \sum_{v \in \epsilon e} x_v > 1\}$. $\rho^* = \tau^* = |E'| - \sum_{e \in \epsilon E'} \sum_{v \in \epsilon e} x_v$.

Proof. By the duality theorem, $\tau^* = \sum_{e \in \epsilon E'} \sum_{v \in \epsilon e} x_v$. In this way, we can rewrite $\sum_{e \in \epsilon E'} \sum_{v \in \epsilon e} x_v$ as $\sum_{e \in \epsilon E'} \sum_{v \in \epsilon e} x_v = \sum_{e \in \epsilon E'} x_e - (\rho^* - |E'|)$ thus yielding the desired result. □

Now, we are able to prove Theorem 6. As it follows the same framework as Section 6.1, we will focus on addressing the difference in this non-trivial extension.

Proof. Similarly, we will show that with positive probability, an instance constructed this way will have a bounded $f(L)$, the maximum number of join results a server can produce, if it loads at most $L$ tuples from each relation. Then setting $\rho \cdot f(L) = \Omega(|Q(R)|)$ yields a lower bound on $L$.

Hard instance construction. There are $N^{\omega_\epsilon}$ distinct values in the domain of attribute $\epsilon$. Namely, a deterministic relation $R(\epsilon)$ is a Cartesian product over all attributes in $\epsilon$, with $|\epsilon| \times N^{\omega_\epsilon} = N^{\omega_\epsilon}$ tuples in total; and a probabilistic relation $R(\epsilon)$ is constructed in a probabilistic way, such that each combination $t \in \times_{\epsilon \in \epsilon} dom(t)$ has a probability of $p(e) = 1/N^{\omega_\epsilon}$ to form a tuples in $R(e)$, with $|\epsilon| \times N^{\omega_\epsilon} \cdot p(e)$ i.e. $N$ tuples in expectation. Moreover, each relation has its input size deviates from its expectation by a constant factor is at most exp$(-\Omega(N))$. Taking all relations in the edge covering, they together form $N^{\omega_\epsilon}$ results while remaining relations are deterministic Cartesian product. So this instance has its output size deviating from its expectation by a constant factor is at most exp$(-\Omega(N))$.

Step 1: Making a reasonable restriction on loading tuples.

Lemma A.7. For any deterministic edge $e$, if $|\Gamma(e) \cap E'| \leq 1$, then making the assumption that tuples loaded from $R(e)$ should be in forms of Cartesian products over all attributes, doesn’t decrease the maximum number of join results that can be produced per server by a constant factor.

Proof. Note that for any vertex $v \in V$, if the two edges incident to it are deterministic, the same argument in Lemma 6.1 can be applied to $v$, i.e., loading tuples in $R(e)$ for $v \in e$ in terms of $L_v \times L_{e - \{v\}}$ will not decrease the optimal solution by a constant factor. Moreover, if $L_v = L_{u,v} \times L_e$, then $L_{u,v} = L_{u,v} \times L_{u,v}$ for any $u \in e - \{v\}$. To prove this result, it suffices to show that $L_{u,v} = L_{u,v} \times L_{u,v}$ for every pair of vertices $u, v \in \epsilon$. We distinguish any deterministic edge $e$ into two cases.

If $\Gamma(e) \cap E' = 0$, Lemma 6.1 can be applied to all vertices to $e$, thus $L_{u,v} = L_{u,v} \times L_{u,v}$ for every pair of vertices $u, v \in \epsilon$. Otherwise, $|\Gamma(e) \cap E'| = 1$, say $\Gamma(e) \cap E' = \{e''\}$. Applying Lemma 6.1 can be applied to every attribute $a \in e - e''$; we have $L_{u,v} = L_{u,v} \times L_{u,v}$.
for every \( u, v \in e - e' \). Note that when \(|e' \cap e| = 1\), we are done. The remaining case is when \(|e' \cap e| \geq 2\), we can assume that the optimal edge covering \( e \) could shift all weights on vertices in \( e - e' \) to one specific vertex in \( e - e' \), and assign 0 for remaining vertices in \( e - e' \), without changing its optimality and property in Definition 6.4. In this way, the condition is also satisfied.

**Step 2: Prove a upper bound on \( J(L) \).**

Assume the number of distinct values from attribute \( v \) loaded by the server is \( z_v \). Observe that \( 1 \leq z_v \leq N^{x_v} \) for each vertex \( v \in V \). Moreover, \( \prod_{v \in e} z_v = L \) for each deterministic relation \( e \).

Recall that relations in \( E' \) are vertex-disjoint. After loading \( L \) tuples from all deterministic relations, there are \( L^{|E'|} \) combinations of results in total, where each of them has a probability of

\[
\prod_{e \in E'} p(e) = \prod_{e \in E'} \frac{1}{N^{\sum_{e \in E'} x_e}} = N^{|E'|-\sum_{e \in E'} \sum_{e \in E'} x_e} = N^{p^{|E'|}}
\]

to form a valid join result, implied by Lemma A.6. The expected number of join results that can be produced by one server is \( L^{p^{|E'|}} \). Next, we will show that this number of join results deviates from its expectation by a constant factor is exponentially small.

For each relation \( e \in E' \), we introduce a random variable \( Y_e \) for each combination \( t \in \times_{v \in E} L_v \), which follows the Bernoulli distribution with parameter \( p(e) \). Denote \( Y_e = \sum_{i} Y_i \). Observe that there are \( L^{|E'|} \cdot \sum_{e \in E'} x_e \) independent random variables in the space \( \times_{v \in E} L_v \). So,

\[
E[Y(e)] = L^{|E'|} \cdot \sum_{e \in E'} x_e \cdot p(e) = N \cdot \frac{L}{N}^{|E'|} \cdot \sum_{e \in E'} x_e
\]

where the last inequality is implied by Lemma A.5. By Chernoff bound, the probability that \( Y(e) \) deviates from its expectation by a constant factor is at most \( \exp(-\Omega(E[Y(e)]) \).

Let \( Y = \prod_{e \in E} Y_e \). As mentioned, these \( L^{p^{|E'|}} \) combinations are not fully independent, as long as they share any same variable \( Y_i \). In fact, the number of independent combinations is \( L^{p^{|E'|}} \cdot |E'| \), since all combinations can be put into disjoint groups by the random variables shared and each group has exactly \( L^{p^{|E'|}} \) combinations. Thus, the probability that the server produces more than \( 2^{|E'|} \cdot L^{p^{|E'|}} \) join results is at most

\[
\Pr(Y \geq 2^{|E'|} \cdot L^{p^{|E'|}}) \leq \sum_{e \in E'} \Pr(Y(e) \geq 2 \cdot E[Y(e)]) \leq \exp(-\Omega(\min_{e \in E'} E[Y(e)]))
\]

Consider an arbitrary attribute \( v \in V \), there are \( \binom{N^{x_v}}{z_v} \) choices of loading \( z_v \) distinct values from the domain of attribute \( v \). Over all possible values of \( z_v \), the number of choices over all attributes in total is

\[
\prod_{v \in E} \sum_{z_v} \exp(O(N^{x_v} - z_v)) = \exp(\tilde{\Theta}(\max_{v \in V} N^{x_v})
\]

By the union bound, the probability that any of the choices produces more than \( 2^{|E'|} \cdot L^{p^{|E'|}} \) join results is at most

\[
\exp(-\Omega(\min_{e \in E'} E[Y(e)]) + \tilde{\Theta}(\max_{v \in V} N^{x_v}))
\]

which is exponentially small if \( \min_{e \in E'} E[Y(e)] \geq c_1 \cdot \max_{v \in V} N^{x_v} \cdot \log N \) for some sufficiently large constant \( c_1 \), or

\[
N - (\frac{L}{N})^{|E'|} \geq c_1 \cdot \max_{v \in V} N^{x_v} \cdot \log N
\]

where \( \lambda(Q) = \min_{e \in E'} |Y(e)| - \sum_{e \in E'} x_e \leq |E'| \). Rearranging it,

\[
L^{\lambda(Q)} \geq c_1 \cdot \Lambda^{\lambda(Q)-1} \cdot \max_{v \in V} N^{x_v} \cdot \log N
\]

We know that \( L \geq \Omega(\frac{N}{p}) \), so this is true as long as

\[
(\frac{N}{p})^{\lambda(Q)} \geq c_2 \cdot \Lambda^{\lambda(Q)-1} \cdot \max_{v \in V} N^{x_v} \cdot \log N
\]

for some sufficiently large constant \( c_2 \), or \( N \geq c_3 \cdot p^{-\min_{v \in V} x_v} \cdot \Lambda^{\lambda(Q)} \). (log \( N \)) for some sufficiently large constant \( c_3 \). Note that \( x \) is constant-small, so \( \frac{\lambda(Q)}{\min_{v \in V} x_v} = O(|E'|) \) is still a constant.

**Step 3: Apply counting argument.**

So far, we have shown that with exponentially high probability each server produces no more than \( O(L^{p^{|E'|}} \cdot N^{p^{|E'|}}) \) join results in each round. Over \( p \) servers, the number of join results produced in total is \( O(pL^{p^{|E'|}} \cdot N^{p^{|E'|}}) \) with high probability. As there are \( N^{p^{|E'|}} \) join results, we must have \( pL^{p^{|E'|}} \cdot N^{p^{|E'|}} \geq N^{p^{|E'|}} \), thus \( L \geq N/p^{1/|E'|} \).