Neither Cover nor Pack: Distributed Worst-Case Optimality of Degree-2 Joins

Heba Aamer
Vrije Universiteit Brussel, Belgium

Xiao Hu
University of Waterloo, Canada

Bas Ketsman
Vrije Universiteit Brussel, Belgium

Abstract

We study the worst-case communication complexity of the join query evaluation problem over large-scale data in distributed shared-nothing systems under the MPC model. We focus on multi-round MPC algorithms that run in constant number of rounds. The problem is well-understood for a few classes of queries, mainly the class of acyclic queries and the class of graph-like queries. For queries not belonging to either class, the complexity picture is much less clear. We study the class of degree-two queries and fragments thereof. In this paper, we tighten the gap between the upper and lower bounds for the studied classes and establish worst-case optimality for some fragments of the considered classes. We also debunk a well-believed conjecture about which query-related quantity, in the worst-case, optimally captures the communication complexity of the studied problem.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Logic and databases; Theory of computation → Abstract machines

Keywords and phrases degree-two joins, worst-case optimality, distributed algorithms

Digital Object Identifier 10.4230/LIPIcs.ICDT.2026.8

Funding This work is partially funded by FWO-grant G062721N.
Heba Aamer: The work of Heba Aamer is fully funded by FWO-grant 1210525N.

1 Introduction

In this work, we study the communication complexity of evaluating a (join) query over large-scale data in distributed shared-nothing systems. Our model of computation is the Massively-Parallel Communication (MPC) model, which has become the standard model of computation in this context, and which puts communication complexity, rather than time complexity, as representative cost measure. We particularly focus on worst-case communication complexity in the constant-round MPC model. This worst-case complexity is well-understood for a few classes of queries, mainly including the class of acyclic queries and the class of graph-like queries, which are queries over binary (and possibly unary and nullary) relations. For queries not belonging to these classes, the picture is much less clear.

To have a better understanding of the communication complexity of the join evaluation problem, we initiate in this work the study on the class of *degree-two queries* as it is a broad class of cyclic queries that allows for relations with unrestricted arities, yet its queries have structural properties. A degree-two query is a query in which every attribute belongs to precisely two relations.[1] Among the class of degree-two queries, we give the subclass of

[1] From this definition, we see that the dual (in terms of hypergraphs) of every degree-two query is a graph-like query. As graph-like queries enjoy some graph-theoretic properties, naturally degree-two queries have similar properties. Those properties for graph-like queries were *sufficient* to obtain a *unified* understanding for their communication complexity. On the other hand, to obtain any non-trivial bounds for degree-two queries, more restrictions were needed as degree-two queries turn out to be more diverse.

semi-grids special attention as its queries enjoy more structural properties. A semi-grid is a degree-two query without odd cycles. Semi-grids can be recognized visually as the nodes of their hypergraph can be placed on a two-dimensional grid such that all relations are vertical or horizontal lines containing precisely the nodes they cross, and with every attribute belonging to precisely one horizontally and one vertically-oriented relation.

In this paper, we investigate what is known about the communication complexity of degree-two queries in terms of upper and lower bounds, we tighten the gap between those bounds for some fragments of degree-two queries, and we show worst-case optimality for a few of those fragments.

In the rest of the introduction, we summarize the literature of the studied problem, mention some concrete research questions that we try to answer, and summarize our contributions.

MPC model

The MPC model is well-established in the database theory community for studying distributed algorithms. The model takes a number p as parameter, representing the number of available servers. Servers have no direct access to each other's memory, but they are connected through a network and can communicate with each other via private (one-to-one) message-passing.

Algorithms in the MPC model are executed in rounds, with each round consisting of a *local computation* phase followed by a *global communication* phase. In the local computation phase, every server performs a computation over the data it has locally. During this phase, no messages are sent or received. In the communication phase, every server can send messages to other servers in the network. During this phase, no computation is performed by the servers. We refer to the number of messages (i.e., tuples) received by a server during the communication phase as the server's *load* in that round.

In general, the cost of an MPC algorithm is measured in terms of *maximum load* and number of rounds. By maximum load, we mean the maximum load of any server during any of the algorithm's rounds. Nonetheless, in this work we only consider constant-round algorithms, for which the maximum load naturally defines the asymptotic cost of the entire algorithm. Since our analysis is for constant-round algorithms, sometimes we do not explicitly mention it in our results.

MPC worst-case optimality of the join evaluation problem

A join algorithm in the MPC model computes a join query q over a database instance $\mathcal{D}$ that is distributed over the p servers *correctly*, when every tuple in the output of q over $\mathcal{D}$ is computable (i.e., producible) by at least one of the servers in some round. In this paper, we focus on *worst-case optimality*. That is, we are interested in algorithms that optimize (i.e., minimize) the maximum load of the algorithm w.r.t. given cardinality constraints, rather than for the specific database instance over which the query is computed. The instance $\mathcal{D}$ is initially evenly (and arbitrarily) partitioned over the p servers, hence requiring a *linear* load of $|\mathcal{D}|/p$ to read the input (with $|\mathcal{D}|$ being the number of tuples in $\mathcal{D}$). This linear load is essentially the best load one can hope for, but it is unrealistically low for most queries.

Several asymptotic lower and upper bounds for the join query evaluation problem in the MPC model are known. All the bounds relevant to this line of research, and hence to this paper, are of the form $|\mathcal{D}|/p^{1/\ell}$, with ℓ a number depending on the structure of the target query. The larger the value of ℓ, the higher the load is. In the context of lower bounds, we will often say the *load is based on* ℓ to mean that it is bounded from below by $|\mathcal{D}|/p^{1/\ell}$. Similarly, in the context of upper bounds, we say that the *cost of an algorithm is based on* ℓ

to mean that the load is bounded from above by $|\mathcal{D}|/p^{1/\ell}$. Finally, it is noteworthy that the bounds are in terms of $|\mathcal{D}|$ (and not the number of bits) because we consider tuple-based algorithms, inline with other works.

Known bounds and algorithms

Early work mostly focuses on single-round `MPC` algorithms [2, 3, 4, 10] and in this variant of the model (opposed to the multi-round variant) the worst-case load is now well-understood. An algorithm of particular importance in this context is `HyperCube` [2, 3], also called the *shares* algorithm. The `HyperCube` algorithm runs with a load of $|\mathcal{D}|/p^{1/\tau}$ where τ is the *fractional vertex covering number* (aka, fractional edge packing number) of the query (see Section 2). This load is worst-case optimal in the sense that its load matches a $|\mathcal{D}|/p^{1/\tau}$ lower bound [4] up to a polylog factor and with high probability when the input database is *skew-free* and with all relations having the same size. Skew-freeness means that the degrees of attribute values (and by extension, partial tuples over attributes) do not exceed certain threshold values that are defined by a parameterization of the run for `HyperCube`.

Without constraints on the database, the worst-case optimal load of single-round `MPC` algorithms is $|\mathcal{D}|/p^{1/\psi}$ and an algorithm exists that can achieve it w.h.p. and up to a polylog factor [10]. The number ψ is the *edge quasi-packing number* and is equal to the maximum fractional vertex covering number of all residuals of the target query. A residual of a query q is the query obtained after removing some attributes of q (from all relations that have these attributes). Since every query is a residual of itself, it is immediate that $\tau \leq \psi$ holds.

When more than one round is allowed, the load can be significantly improved, below ψ, for some queries. A key advantage of the multi-round MPC model is that (residual) queries can be simplified by computing semi-joins (i.e., removing redundant relations) in one round with linear load, before proceeding with the actual join computation.

For the multi-round variant, a few results are known, and several questions are open. It is known that, for every query, the worst-case optimal load is at least $|\mathcal{D}|/p^{1/\rho}$, where the number ρ is the *fractional edge covering number* (aka, fractional vertex packing number) of the query (see Section 2). This lower bound result was obtained in [10], and for some join queries this bound is tight. This is the case for Loomis-Whitney (LW) queries (which is a very specific family of queries with $\tau = \rho$) [10], all graph-like queries (i.e., join queries involving relations with at most two attributes) [10, 8, 12, 11, 9, 1] as well as for all join queries that are acyclic [5, 13, 1]. We remark that $\rho \leq \psi$ holds for all queries, and hence, the one-round algorithm is already optimal in the multi-round setting for all queries with $\rho = \psi$.

For a while, it has been thought that $|\mathcal{D}|/p^{1/\rho}$ could be the worst-case optimal load in general, but this idea was recently debunked. In particular, it is shown in [5] that $|\mathcal{D}|/p^{1/\tau}$ is a tighter lower bound for a very specific fragment of semi-grids. This new lower bound directly shows that there are other queries for which the one-round algorithm based on ψ is effectively worst-case optimal in the multi-round case. This is precisely the case for all degree-two queries with $\tau = \psi$ and with lower bound on the load based on τ such as boat queries (see Section 2 for the precise definition).

It is worth noting that the initial worst-case optimal multi-round algorithms proposed in the literature for LW, graph-like, or acyclic queries cannot straightforwardly be generalized to arbitrary join queries, and hence, the one-round worst-case load based on ψ remains a reasonable generic upper bound in the multi-round setting as well.

Known generic algorithms

In efforts to have a multi-round algorithm that works for arbitrary join queries, two algorithms of different nature have been proposed in the literature [11, 1]. We will refer to these as the ϕ-algorithm [11] (see below what ϕ is) and the PAC algorithm [1],[2] respectively. It is worth noting that the load of the PAC algorithm depends on a parameter whose optimal value is non-trivial to compute.

The load of the ϕ-algorithm has a fixed upper bound of $|\mathcal{D}|/p^{1/\alpha\phi/2}$ with α being the maximum arity of the relations in the query and ϕ being a new query-related quantity called *generalized vertex packing number*. It was shown that $\phi \geq \rho$ holds for every query as well as $\alpha\phi \geq k$ where k is the total number of attributes in the query.[3] The general upper bound can be shown to be worst-case optimal for graph-like queries and boat queries. However, this bound does not recover the worst-case optimality for acyclic queries nor for LW queries. Moreover, there is no clear relationship between ψ and $\alpha\phi/2$. In particular, some queries have $\psi < \alpha\phi/2$, which means that for these queries the worst-case-optimal one-round algorithm guarantees a strictly better load than the multi-round ϕ-algorithm.

On the other hand, the PAC algorithm takes a parameter r called the spectrum which is a sequence of discrete values allowing for a more fine-grained definition of skew. Parameterized with a specific spectrum r, the algorithm runs with load γ_{r} that is defined to be the best possible load of the most difficult subproblem. The more-fine grained the choice of r, the tighter the load of the algorithm can be. It is shown that for any choice of spectrum r, $\gamma_{\mathsf{r}} \leq \psi$ holds and hence it runs with a load that is at most that of the one-round algorithm. This straightforwardly recovers the worst-case optimality for boat queries. It was also shown that with a choice of $\mathsf{r} = 0, 1/2, 1$, the relationship $\gamma_{\mathsf{r}} = \rho$ holds for graph-like queries and for acyclic queries, which is worst-case optimal for both classes. The LW queries, on the other hand, are the only class of queries whose worst-case optimality was not recovered by the PAC algorithm, which is alluded to a non-tight load analysis of the algorithm.

Questions, results, and contributions

Based on the current literature, in this work we investigate the following questions:

1) It is known that $\tau \geq \rho$ holds for all degree-two queries. Moreover, in case the database is already skew-free, the aforementioned optimal one-round load bound is based on τ. Accordingly, the generic multi-round lower bound that is based on ρ seems unreasonable. We thus investigate if the τ-lower bound can be generalized to other degree-two queries.

2) There is a believed conjecture in [5] that the worst-case load for any query is characterized by $\max\{\tau, \rho\}$. Can we verify if this conjecture truly hold for all degree-two queries?

3) We have a clear understanding of the load for the class of boat queries, which is a very restricted fragment of degree-two queries. Can we show any non-trivial upper bounds for subclasses of degree-two queries that *structurally* generalize the class of boat queries? And can any of the existing algorithms be used to achieve this?

To answer the third question, we consider two generalizations of the class of boat queries: *semi-binary* queries and *grid* queries. The two classes are, in a sense, complementary to each other, where, intuitively, the first limits the *connectivity* of the relations, while the other fully

[2] For completeness, there is a map-reduce join algorithm that shares some similarities with the original PAC algorithm [7]. The algorithm allows the number of servers to grow as needed to achieve a specific load guarantee which suits the map-reduce model but not the MPC model. For this reason, we no longer mention that algorithm.

[3] The relationship of $\alpha\phi \geq k$ follows from Lemma 3.1 and the discussion of Lemma 4.3 in [11].

Table 1 Summary of existing known worst-case complexity results on the communication load in the MPC model along with our new results. The notation $\tilde{\mathcal{O}}(\cdot)$ hides a polylog factor that depends on p. Recall that τ is fractional vertex covering number, ρ is fractional edge covering number, ψ is the edge quasi-packing number, k is the number of attributes in the query, α is the maximum arity, and ϕ is the generalized vertex packing number. Moreover, recall that $\alpha\phi \geq k$ holds for any query.

Generic Bounds	Class of Queries	Specific Multi-Round Bounds [Sources]
	acyclic	$\tilde{\mathcal{O}}\left(\lvert\mathcal{D}\rvert/p^{\frac{1}{\rho}}\right)$ [5, 1]
$\Omega\left(\lvert\mathcal{D}\rvert/p^{\frac{1}{\rho}}\right)$ [10]	LW	$\tilde{\mathcal{O}}\left(\lvert\mathcal{D}\rvert/p^{\frac{1}{\rho}}\right)$ [10]
$\tilde{\mathcal{O}}\left(\lvert\mathcal{D}\rvert/p^{\frac{1}{\psi}}\right)$ [10, 1]	graph-like	$\tilde{\mathcal{O}}\left(\lvert\mathcal{D}\rvert/p^{\frac{1}{\rho}}\right)$ [8, 12, 11, 9, 1]
$\tilde{\mathcal{O}}\left(\lvert\mathcal{D}\rvert/p^{\frac{1}{\alpha\phi/2}}\right)$ [11] $\tilde{\mathcal{O}}\left(\lvert\mathcal{D}\rvert/p^{\frac{1}{\gamma_r}}\right)$ [1]	degree-two $(\min\{\psi, k/2\} \geq \tau \geq \rho)$	$\Omega\left(\lvert\mathcal{D}\rvert/p^{\frac{1}{\tau}}\right)$ [5] extended by Theorem 9 $\tilde{\mathcal{O}}\left(\lvert\mathcal{D}\rvert/p^{\frac{1}{\min\{\psi,k/2\}}}\right)$ Theorem 22
	grids $(k/2 \geq \psi = \tau + \rho - 2)$	$\Omega\left(\lvert\mathcal{D}\rvert/p^{\frac{1}{\max\{\tau,\tau+\rho-4\}}}\right)$ Theorem 7 $\tilde{\mathcal{O}}\left(\lvert\mathcal{D}\rvert/p^{\frac{1}{\max\{\tau,\tau+\rho-3\}}}\right)$ Theorem 24

exploits this such that the diameter of any query is exactly two.[4] To better understand the rest of this section, we summarize the relationship between the aforementioned classes in the next depiction. See Section 2 for formal definitions.

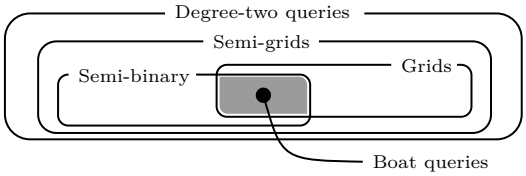


Table 1 gives a summary of our worst-case load results in addition to the already known bounds. These complexity results and our other contributions are summarized as follows.

Lower bounds: debunk and go beyond.

1. We show that the load for computing grid queries is at least $|\mathcal{D}|/p^{1/\max\{\tau,\tau+\rho-4\}}$. For the class of grid queries, this load is both higher than the loads based on τ and ρ, debunking the existing conjecture in [5] that the worst-case optimal load in the constant-round MPC model is based on $\max\{\tau, \rho\}$ for all join queries.
2. We identify a simple class of semi-binary queries as the hardness core of all semi-grid queries. Precisely, we show that if the load for this class of queries is at least the load based on τ, then the same holds for the entire class of semi-grid queries. The latter enables a new class of queries whose lower bound depends on τ and extends the existing understanding of [5].
3. *Additional Technical Contribution:* The previous lower-bound results are obtained through a new generic approach that can reduce a query to its sub-queries. More precisely, we present in Theorem 6 a generic framework that allows to deduce that some queries are at least as hard to solve as *some* of their residual queries.

[4] The diameter of a query is the maximum length of a shortest path connecting any two of its attributes.

New upper bounds.

4. We give an algorithm that computes every degree-two query with load $|\mathcal{D}|/p^{1/\min\{\psi,k/2\}}$ (recall that k is the number of attributes in the query). This improves on the upper bound known by the ϕ-algorithm since $\alpha\phi \geq k$ holds for all queries.
5. We give different algorithms showing that every grid query is computable with a load that is at most $|\mathcal{D}|/p^{1/\max\{\tau,\tau+\rho-3\}}$, which is strictly better than the one-round algorithm and the ϕ-algorithm since $\alpha\phi/2 \geq \psi = \tau + \rho - 2$ for grid queries.
6. *Additional Technical Contribution:* All our algorithms are instantiations of the `PAC` algorithm [1]. However, in some cases, the load guarantee of the `PAC` algorithm was not tight enough. To obtain tighter upper-bound guarantees, we provide a new analysis of the load guarantee of the `PAC` algorithm that is tighter compared to the original analysis. Thereto, we define a new query related quantity, the *edge-weighted fractional vertex cover*, which better captures the load of a query when evaluated on a skew-free database with relations having different cardinalities.

Understanding the gap based on query structures.

7. Our generic upper bound for degree-two queries is already optimal for all semi-binary queries with $\tau = k/2$ and without *odd-like* cycles (see Corollary 23). This result is interesting because it is the first class of queries with $\rho \leq \tau$ to show optimality for, for which the one-round algorithm is not. Remark that all other solved classes of queries by multi-round algorithms have $\tau \leq \rho$ in common.
8. Additionally, our algorithms to compute grid queries are strictly better than the one-round algorithm and the ϕ-algorithm, since $\alpha\phi/2 \geq \psi$ for grid queries. This upper bound is also worst-case optimal for all grid queries with $\rho \leq 3$ and unbounded τ (see Corollary 25).

Paper outline

The paper is further organized as follows.[5] In Section 2, we give the necessary preliminaries and formally define the class of degree-two queries and the considered subclasses with their properties. Section 3 presents our new lower-bound results. In Section 4, we briefly review the `PAC` algorithm where we give a tighter load analysis for the algorithm. Afterwards, in Section 5, we use the results of the algorithm section to show our new upper-bound results. Finally, we conclude in Section 6.

2 Preliminaries

For a positive integer n, we write $[n]$ to denote the set $\{1, \ldots, n\}$ of integers, and $[n]_0$ to denote the set $\{0, 1, \ldots, n\}$. We write $[0, 1]$ as an abbreviation for the set $\{i \in \mathbb{Q}_{\geq 0} \mid 0 \leq i \leq 1\}$ of rationals between, and including, 0 and 1. For a family $\mathcal{S} = \{S_1, \ldots, S_m\}$ of sets, we use $\mathsf{ins}(\mathcal{S})$ to denote the set of all nonempty subsets of the sets of $\mathcal{S}$; that is, $\mathsf{ins}(\mathcal{S}) := \left(\bigcup_{i\in[m]} 2^{S_i}\right) \setminus \emptyset$. We also use $\mathsf{elem}(\mathcal{S})$ to denote the set of all elements appearing in the sets of $\mathcal{S}$; that is, $\mathsf{elem}(\mathcal{S}) := \bigcup_{i\in[m]} S_i$.

Henceforth, we assume countably infinite disjoint domains **rels**, **atts**, and **dom** of relation names, attributes, and data values, respectively. For convenience of notation, and specifically to avoid nondeterminism in some of our results, we assume a total order $\leq_{\mathbf{rels}}$ over **rels**.

[5] We remark that due to space limitations, most of the proofs will appear in a full version of the paper.

Queries

A (join) query q, that is full and self-join free, is a pair $(\mathsf{rels}_q, \mathsf{sch}_q)$ with rels_q being a nonempty finite set of relation names from **rels** and sch_q being a total function that maps every relation name from rels_q to a nonempty finite set of attributes from **atts**. We say that a set A of attributes is a *full set* (w.r.t. q) if there is some relation R with $\mathsf{sch}_q(R) = A$ and there does not exist a relation R' with $\mathsf{sch}_q(R') \supsetneq A$.

Henceforth, we will use $\mathsf{atts}(q)$ to refer to precisely the set of attributes appearing in the image of the function sch_q. We will also use $\mathsf{ins}(q)$ to mean $\mathsf{ins}(S)$ where S is the image of the function sch_q; that is, $\mathsf{ins}(q)$ is the set of sets of attributes of q appearing together in at least one relation of q. For consistency, we will use $\mathsf{rels}(q)$ as an equivalent notation to denote the set rels_q. Moreover, for a query q and a set $A \in \mathsf{ins}(q)$ of attributes, we slightly abuse notation and use $\mathsf{rels}_q(A)$ to denote the set of relations in q that contains all the attributes of A; that is, $\mathsf{rels}_q(A) := \{R \in \mathsf{rels}(q) \mid A \subseteq \mathsf{sch}_q(R)\}$. When $A = \{\mathbf{a}\}$ is a set of a single attribute, we will opt to write $\mathsf{rels}_q(\mathbf{a})$ in place of $\mathsf{rels}_q(\{\mathbf{a}\})$.

Instances

A tuple $\boldsymbol{t}$ over a finite set of attributes $A \subseteq \mathbf{atts}$ is a total function from A to **dom**. When we want to emphasize the fact that a tuple $\boldsymbol{t}$ is defined over a set A, we will say that $\boldsymbol{t}$ is an A-tuple. For an A-tuple $\boldsymbol{t}$ and a set $B \subseteq A$, we write $\boldsymbol{t}[B]$ to denote the B-tuple with $\boldsymbol{t}[B](\mathbf{a}) = \boldsymbol{t}(\mathbf{a})$ for every $\mathbf{a} \in B$ in which case we say that $\boldsymbol{t}$ is an *extension* of $\boldsymbol{t}[B]$. By convention, we consider every tuple an extension of itself and of the empty tuple. We say that an A-tuple $\boldsymbol{t}$ and a B-tuple $\boldsymbol{u}$ are *consistent* if $\boldsymbol{t}[A \cap B] = \boldsymbol{u}[A \cap B]$.

A finite set of tuples over the same set A of attributes is called a *relation instance* over A. A (database) *instance* $\mathcal{D} := (\llbracket \cdot \rrbracket_{\mathcal{D}})$ for a query q is a mapping $\llbracket \cdot \rrbracket_{\mathcal{D}}$ associating every relation name $R \in \mathsf{rels}(q)$ with a relation instance $\llbracket R \rrbracket_{\mathcal{D}}$ over $\mathsf{sch}_q(R)$, where we write $|\mathcal{D}| := \sum_{R \in \mathsf{rels}(q)} |\llbracket R \rrbracket_{\mathcal{D}}|$ to denote the total number of tuples in $\mathcal{D}$. An instance $\mathcal{F}$ is said to be a *fragment* of $\mathcal{D}$ if $\llbracket R \rrbracket_{\mathcal{F}} \subseteq \llbracket R \rrbracket_{\mathcal{D}}$ for every relation R that $\mathcal{F}$ is defined for.

Sometimes we will be interested in relating the cardinalities of relation instances in a fragment $\mathcal{F}$ to the size of $\mathcal{D}$. To capture this information, we associate a *size reduction* mapping Δ for a query q that is defined as $\Delta : \mathsf{rels}(q) \to [0, 1]$. Henceforth, we use the term *default size reduction*, denoted Δ_0, to refer to the mapping that assigns 0 to every relation of q. We define (where the value for v will be set accordingly when needed):

▶ **Definition 1.** *Let q be a query and $\mathcal{F}$ an instance that is a fragment of an instance $\mathcal{D}$. We say that $\mathcal{F}$ is* compatible with *a size reduction Δ for q and a positive value v if $|\llbracket R \rrbracket_{\mathcal{F}}| \leq |\mathcal{D}|/v^{\Delta(R)}$ for every relation $R \in \mathsf{rels}(q)$.*

Given an instance $\mathcal{D}$ for a query q and a tuple $\boldsymbol{t}$ over $A \subseteq \mathsf{atts}(q)$, we use $\mathcal{D}_{[\boldsymbol{t}]}$ to refer to the instance with $\llbracket R \rrbracket_{\mathcal{D}_{[\boldsymbol{t}]}} = \{\boldsymbol{u} \in \llbracket R \rrbracket_{\mathcal{D}} \mid \boldsymbol{u} \text{ is consistent with } \boldsymbol{t}\}$ for every $R \in \mathsf{rels}(q)$; that is, $\mathcal{D}_{[\boldsymbol{t}]}$ contains *all and only* the tuples of $\mathcal{D}$ that are consistent with $\boldsymbol{t}$ in every relation instance. We say that $\boldsymbol{t}$ is consistent with $\mathcal{D}$ if, for every $R \in \mathsf{rels}(q)$, $\llbracket R \rrbracket_{\mathcal{D}_{[\boldsymbol{t}]}}$ is nonempty. We denote the set of all A-tuples consistent with $\mathcal{D}$ as $\mathsf{joins}^{\mathcal{D}}(A) := \{\boldsymbol{t} \mid \boldsymbol{t} \text{ is an } A\text{-tuple consistent with } \mathcal{D}\}$. The output of a query q over an instance $\mathcal{D}$, denoted $\llbracket q \rrbracket_{\mathcal{D}}$, is defined as $\llbracket q \rrbracket_{\mathcal{D}} := \mathsf{joins}^{\mathcal{D}}(\mathsf{atts}(q))$.

Query transformations

Given a query q, a set $\mathcal{R}$ of relations, and a set A of attributes, we use $\mathsf{sub}_q(\mathcal{R}, A)$ to denote the *subquery* q' of q that is defined with $\mathsf{rels}(q') := \mathsf{rels}(q) \setminus (\mathcal{R} \cup \mathcal{R}')$ and with $\mathsf{sch}_{q'}(R) := \mathsf{sch}_q(R) \setminus A$ for $R \in \mathsf{rels}(q')$ where $\mathcal{R}'$ is the set $\{R \in \mathsf{rels}(q) \mid \mathsf{sch}_q(R) \subseteq A\}$

of relations. That is, q' is obtained from q by removing the relations of $\mathcal{R}$ along with the relations whose attributes are exclusively from A, and by removing all the attributes of A from the remaining relations. Note that when the set $\mathcal{R}$ is empty then q' is the *residual query* of q (w.r.t. A); in which case, we will sometimes write q_A in place of $\mathsf{sub}_q(\emptyset, A)$.

For two distinct relations $R_1, R_2 \in \mathsf{rels}(q)$, we say that R_1 is *reducible* into R_2 when $\mathsf{sch}_q(R_1) \subsetneq \mathsf{sch}_q(R_2)$. We say that q is *reduced* if it has no reducible relations. We also say that a query q is not *clean* if there are two distinct relations $R_1, R_2 \in \mathsf{rels}(q)$ with $\mathsf{sch}_q(R_1) = \mathsf{sch}_q(R_2)$; otherwise, the query q is said to be clean. A query q that is not reduced or not clean can be turned into a clean and reduced query by removing duplicated and reducible relations. Since multiple outcomes are possible depending on the order in which relations are removed, we denote by $\lfloor q \rfloor$ the *unique* clean and reduced query derived from q by removing duplicated and reducible relations, one-by-one, following the order of $\leq_{\mathbf{rels}}$.

Weight mappings and edge-weighted fractional vertex covers

It is common to view queries in terms of their hypergraph representation. This allows us to use graph-theoretic notions in our analysis. For this, the term *vertex* usually corresponds to *attribute* and *edge* corresponds to *relation (atom)*. In this work, we define a new query-related quantity called *edge-weighted fractional vertex cover* that captures the load of computing a query over a fragment taking into consideration a size reduction. For this, we consider weight mappings for queries q which are functions $f : \mathsf{atts}(q) \to [0, 1]$ associating nonnegative rational weight $f(\mathbf{a}) \in [0, 1]$ to every attribute $\mathbf{a} \in \mathsf{atts}(q)$. For simplicity of notation, we often write $f(A)$ with a set of attribute $A \subseteq \mathsf{atts}(q)$ to denote the sum $\sum_{\mathbf{a}\in A} f(\mathbf{a})$ of the individual weights. The *weight* of f is then defined to be the value of $f(\mathsf{atts}(q))$.

A relation R of q is said to be *covered* by a weight mapping f *w.r.t. a given size reduction* Δ if $f(\mathsf{sch}_q(R)) \geq 1 - \Delta(R)$. We then call f an *edge-weighted fractional vertex cover for* q *w.r.t.* Δ if f covers every relation of q w.r.t. Δ. Consistent with the literature, we call f a *fractional vertex cover for* q if f is an edge-weighted fractional vertex cover for q w.r.t. Δ_0. Observe that the minimum weight taken over all fractional vertex covers for q defines the query-related quantity $\tau(q)$ which is the *fractional vertex covering number* of q.

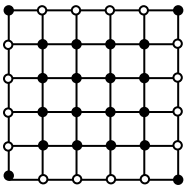
Two other query-related quantities are important: the *edge quasi-packing number* ($\psi(q)$) and the *fractional vertex packing number* ($\rho(q)$). The value of $\psi(q)$ is defined to be the maximum fractional vertex covering number taken over all the residual queries of q; that is, $\psi(q) := \max\{\tau(q') \mid q' \text{ is the query } q_A \text{ for a set } A \text{ of attributes}\}$. On the other hand, $\rho(q)$ is defined to be the maximum weight of a weight mapping $f : \mathsf{atts}(q) \to [0, 1]$ such that the sum of weights assigned to the attributes of every relation is ≤ 1.[6]

Classes of degree-two queries and their properties

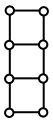
We formally define, in this section, the classes of queries we consider. We give in Figure 1 depiction of three queries that illustrate different query classes we consider.

A query is *linear* if every two relations have at most one attribute in common. A degree-two query is a query q whose all attributes belong to exactly two relations, that is, $|\mathsf{rels}_q(\mathbf{a})| = 2$ for every attribute $\mathbf{a}$. Given a query q, a sequence of $n + 1$ relations $R_1, R_2, \ldots, R_n, R_1$ from $\mathsf{rels}(q)$, with the relations $R_1, \ldots, R_n$ being pairwise distinct, is

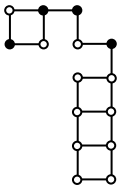
[6] Fractional vertex covers and fractional edge packings are duals with τ being the optimal for both. Similarly, fractional vertex packings and fractional vertex coverings are duals with ρ being their optimal values.



(a) Hypergraph of the query q_a, which is 6×6 grid query with $\tau = \rho = 6$.



(b) Hypergraph of the query q_b, which is a boat query with $\tau = 4$ and $\rho = 2$.



(c) Hypergraph of the query q_c, which is a semi-binary query with $\tau = 7$ and $\rho = 5$.

Figure 1 Examples of different degree-two queries that we will use throughout the paper. To avoid clutter, in all depictions, we represent hyperedges by straight lines such that all the nodes connected by a straight line form a hyperedge (i.e., a relation). The color of the nodes will be meaningful later.

called a *cycle* if there is an attribute in common between each two consecutive relations. The length of the cycle in that case is the number n. A cycle of length $n \geq 3$ is said to be *odd-like* if n is even and $n/2$ is an odd integer.

In what follows, we will refer to the class of linear reduced degree-two queries without odd-length cycles as *semi-grids*. Semi-grids enjoy the following nice property :

▶ **Proposition 2.** *Every semi-grid query has a fractional vertex cover, fractional edge cover, fractional vertex packing, and fractional edge packing that are optimal with integral weights.*

In this work, we are interested in the class of semi-grids in general and particularly in two fragments thereof: semi-binary queries and grid queries. A query q is said to be *semi-binary* if it is a semi-grid and every attribute belongs to at least one binary relation of q.

As for grid queries, for convenience, we will only define them in some canonical form. An $\mathtt{r} \times \mathtt{c}$ grid query, with integers $\mathtt{r}, \mathtt{c} \geq 2$, is composed of two disjoint sets of relations $\mathcal{R}_\mathtt{r} = \{R_{(i,*)} \mid i \in [\mathtt{r}]\}$ and $\mathcal{R}_\mathtt{c} = \{R_{(*,j)} \mid j \in [\mathtt{c}]\}$. The former are called *row* relations while the latter are called *column* relations. Each row relation $R_{(i,*)}$ has the set of $\{\mathbf{a}_{(i,j)} \mid j \in [\mathtt{c}]\}$ as its attributes. Similarly, each column relation $R_{(*,j)}$ has $\{\mathbf{a}_{(i,j)} \mid i \in [\mathtt{r}]\}$. Thus, the set of attributes of q is $\{\mathbf{a}_{(i,j)} \mid i \in [\mathtt{r}], j \in [\mathtt{c}]\}$. Notice that every attribute $\mathbf{a}_{(i,j)}$ belongs to exactly two relations: the row relation $R_{(i,*)}$ and the column relation $R_{(*,j)}$. The query q_a in Figure 1a is an example of a grid query.

For grid queries, we know the following (see Appendix A.1 for a proof):

▶ **Proposition 3.** *Let q be an arbitrary $\mathtt{r} \times \mathtt{c}$ grid query with $\mathtt{r}, \mathtt{c} \geq 2$. Then, we have $\tau(q) = \max\{\mathtt{r}, \mathtt{c}\}$, $\rho(q) = \min\{\mathtt{r}, \mathtt{c}\}$, and $\psi(q) = \mathtt{r} + \mathtt{c} - 2$.*

Observe that the class of boat queries is precisely the class of semi-grids that are grids and semi-binary queries at the same time. Every boat query is an $\mathtt{r} \times \mathtt{c}$ grid query with $\min\{\mathtt{r}, \mathtt{c}\} = 2$. From that observation and by Proposition 3, we directly obtain:

▶ **Lemma 4.** *Every boat query q has $\tau(q) = \psi(q)$.*

3 Lower bounds

All existing lower-bounds on the worst-case load for computing queries in the constant-round MPC model are of the form $m/p^{1/\ell}$, with ℓ a number based on the structure of the considered query q, with m the size of the relations in the database instance, and with p the considered number of servers. Depending on the different results [5], some restrictions may apply to

m and p. The bounds that we introduce in this section also have this form. Therefore, we will write $\mathsf{LB}[\ell]$ for positive number ℓ to denote the class of queries for which $\Omega\left(m/p^{1/\ell}\right)$ is considered a correct worst-case load. To be precise, for a query q, we write $q \in \mathsf{LB}[\ell]$ to mean that for every m and every p evaluating q requires a load $\Omega\left(m/p^{1/\ell}\right)$ on some instance $\mathcal{D}$ with relations of size m using p servers. From the literature, we have $\tau(q)$ and $\rho(q)$ as common values for ℓ; in which case, we are simply going to write $q \in \mathsf{LB}[\tau]$ or $q \in \mathsf{LB}[\rho]$ to mean the classes $\mathsf{LB}[\tau(q)]$ and $\mathsf{LB}[\rho(q)]$ respectively.

The following results are known to hold [10, 5]:

- for every query q, $q \in \mathsf{LB}[\rho]$. Notice that all queries of Figure 1 belong to this category;
- for every query q that belongs to a strict subset of semi-grid queries (including boat queries), $q \in \mathsf{LB}[\tau]$. Note that, from Figure 1, only q_a and q_b belong to this category where for q_b the τ-based lower bound is higher (i.e., tighter) compared to the ρ-based lower bound and it is indeed optimal.

Similar in spirit to how the one-round lower bound for the worst-case load considers the worst-case load of all residual queries, our new lower bounds, for the classes of degree-two queries, take the lower bounds of *some* sub-queries into consideration.

Subquery-based lower-bound framework

We motivate the framework with the following example.

▶ **Example 5.** For illustration of what we mean by subquery-based lower bounds, consider q_a and q_c from Figures 1a and 1c, respectively. The best-known lower bound for q_a is based on either τ or ρ as they are equal, hence, we have no knowledge that $q_a \in \mathsf{LB}[\ell]$ for $\ell > 6$. As for q_c, its best-known lower bound is based on ρ although for this query $\tau = 7 > \rho = 5$. For each of the two queries, take q'_a and q'_c to be the subqueries of q_a and q_c obtained by removing all the black-colored attributes from Figures 1a and 1c respectively. We will be able to verify that $q'_a \in \mathsf{LB}[\ell_a]$ for $\ell_a > 6$ and $q'_c \in \mathsf{LB}[\ell_c]$ for $\ell_c > 5$.

In this section, we argue that best-known lower bounds for q_a and q_c are not tight enough. To establish this, we show that the worst-case load for q'_a and q'_c cannot be higher than the worst-case load for the original queries.

We formulate this idea as follows:

▶ **Theorem 6.** *Let q be a query and q' be a reduced subquery of q with the same set of relations. Then, for* MPC *join algorithms running in constant number of rounds, the worst-case load for q is at least the worst-case load for q'; that is, if $q' \in \mathsf{LB}[\ell]$ for some ℓ, then $q \in \mathsf{LB}[\ell]$.*

Next, we apply the framework for the class of grid queries establishing a tighter lower bound for this class and we also apply it to the class of semi-grids. In the latter case, we rely on a new result on the worst-case load required to compute a Cartesian-product query.

Grids lower bound

For grid queries, it is known that $\tau \geq \rho$, and the τ-based lower-bound framework of [5] applies. In this section, we improve on that lower bound for grid queries. Precisely:

▶ **Theorem 7** (Grids Lower Bound)**.** *For every $\mathtt{r} \times \mathtt{c}$ grid query q and integer m, there is a family of instances $\mathcal{D}$ with relation instances of size m such that the load required to compute $[\![q]\!]_{\mathcal{D}}$ over p servers is $\Omega\left(m/p^{1/\max\{\tau,\tau+\rho-4\}}\right)$ w.h.p.*

Without loss of generality, we assume that $\mathtt{r} \geq \mathtt{c}$, and hence, by Proposition 3, we know that $\tau = \mathtt{r}$ and $\rho = \mathtt{c}$ in this case. For grids with $\mathtt{c} \leq 4$, the maximum between $\{\mathtt{r}, \mathtt{r} + \mathtt{c} - 4\}$ is simply $\mathtt{r} = \tau$ which is the best-known bound. For this reason, we continue the argument for grids with $\mathtt{c} \geq 5$, which are the queries for which a tighter lower bound can be established.

The next proposition along with Theorem 6 completes the argument for Theorem 7. The idea of the construction is similar to obtaining the subquery q'_a from the query q_a as per Example 5.

▶ **Proposition 8.** *For every $\mathtt{r} \times \mathtt{c}$ grid query q with $\mathtt{r} \geq \mathtt{c} \geq 5$, there is a reduced and clean subquery q' of q with $\mathsf{rels}(q) = \mathsf{rels}(q')$ and $q' \in \mathsf{LB}[\mathtt{r} + \mathtt{c} - 4]$.*

Semi-grids lower bound

We now turn to the (full) class of semi-grids. It is known that every semi-grid query has $\tau \geq \rho$. Nonetheless, for some semi-grids, the best-known lower bound on the worst-case load is based on ρ (query q_c of Figure 1c is an example.) In the rest of this section, we show that the lower bound based on τ can be extended to all semi-grid queries *if* all semi-binary queries with some specific property have τ as a valid lower bound on the worst-case load.

▶ **Theorem 9** (Semi-Grids Conditional Lower Bound)**.** *If $q \in \mathsf{LB}[\tau]$ holds for every semi-binary query q with $\tau(q) = |\mathsf{atts}(q)|/2$. Then, for every semi-grid query q' and integer m, there is a family of instances $\mathcal{D}$ of relation instances of size m such that the load required to compute $[\![q']\!]_{\mathcal{D}}$ over p servers is $\Omega\left(m/p^{1/\tau}\right)$ w.h.p.*

The above theorem directly pinpoints to the simple fragment of degree-two queries that is most challenging within the class of semi-grids. In order to have a better understanding for this fragment, we give a rather simple (sufficient) condition for semi-binary queries q with $\tau(q) = |\mathsf{atts}(q)|/2$ for which the τ-based lower-bound argument applies. The condition is based on the absence of odd-like cycles from queries.

▶ **Theorem 10.** *For every semi-binary q with $\tau = k/2$ and without odd-like cycles, $q \in \mathsf{LB}[\tau]$.*

To establish Theorem 9, we relate the worst-case load of computing a Cartesian product between two queries to the worst-case load known for those queries in the following way.

▶ **Proposition 11.** *Let q_1 and q_2 be two distinct queries with disjoint sets of attributes such that $q_1 \in \mathsf{LB}[\tau]$ and $q_2 \in \mathsf{LB}[\rho]$. Suppose that q is the Cartesian product query $q_1 \times q_2$. Then, $q \in \mathsf{LB}[\tau(q_1) + \rho(q_2)]$.*

By Proposition 11, Theorem 9 easily follows from the next proposition and Theorem 6.

▶ **Proposition 12.** *Let q be a semi-grid. Then, there is a (possibly empty) set A of attributes for which q_A is reduced and with $\mathsf{rels}(q) = \mathsf{rels}(q_A)$ such that one of the following holds:*

- *$\tau(q) = \rho(q_A)$;*
- *$\tau(q) = \tau(q_A)$ and q_A is a semi-binary query with $\tau(q_A) = |\mathsf{atts}(q_A)|/2$; or*
- *q_A is the Cartesian product of two queries q_1 and q_2 such that $\tau(q) = \tau(q_1) + \rho(q_2)$ where q_1 is a semi-binary query with $\tau(q_1) = |\mathsf{atts}(q_1)|/2$.*

From Example 5, note that q'_c is a subquery of the third form as in Proposition 12 with q_1 being a 4×2 grid query (i.e., boat query) of $q_1 \in \mathsf{LB}[\tau]$ and q_2 being a query of 3 singleton duplicated relations of $q_2 \in \mathsf{LB}[\rho]$ and $\tau(q) = 7 = \tau(q_1) + \rho(q_2) = 4 + 3$. Thus, by Proposition 11, we can deduce q_c of Figure 1c has the property $q_c \in \mathsf{LB}[\tau]$. Thus,

▶ **Corollary 13.** *There exist semi-grid queries q with $q \in \mathsf{LB}[\tau]$ for which the τ-based lower-bound framework of [5] does not apply.*

4 The PAC algorithm: new analysis

We mentioned, in the introduction, the load guarantee, γ, that the PAC algorithm provides, is dependant on a parameter r referred to as the *spectrum*. In its general form, the spectrum r is a finite increasing sequence $r_0 = 0, r_1, \ldots, r_k = 1$ of rationals from $[0, 1]$ for $k \geq 1$.

For a particular choice of r, we know the following (where $\gamma_{\mathsf{r}}(q)$ is the guarantee we can *algorithmically* obtain for a query q given r and it will be defined later):

▶ **Theorem 14** (Theorem 8 of [1]). *Let q be a query and $\mathcal{D}$ an instance. Then, the PAC algorithm computes $[\![q]\!]_{\mathcal{D}}$ in three rounds using p servers with load $\tilde{\mathcal{O}}\left(|\mathcal{D}|/p^{1/\gamma_{\mathsf{r}}}\right)$ w.h.p.*

We remark that, although we give in this section a slightly different, yet more accessible, presentation for the PAC algorithm, all the relevant upper-bound results carry over to our presentation. One such result is the following (where $\mathsf{r} = 0, 1$ is a valid spectrum):

▶ **Theorem 15** (Theorem 18 of [1]). *Let q be an arbitrary query. Then, for any choice of spectrum r, the relationship of $\gamma_{\mathsf{r}} \leq \psi$ holds.*

In what follows, we give a high-level overview of the PAC algorithm. We then point out to one limitation in the original analysis and show how to overcome this limitation resulting in a tighter analysis. Due to space limitations, all missing details will appear in the full version.

Spectrum indicator, ranges, and fragments

In this work, we restrict ourselves to a particular form of spectrums that we define using what we call *spectrum indicator*. A spectrum indicator s is a positive integer. The spectrum defined by such a spectrum indicator s is the sequence $0/\mathsf{s}, 1/\mathsf{s}, \ldots, (\mathsf{s}-1)/\mathsf{s}, \mathsf{s}/\mathsf{s}$ of rationals (that is, $r_i = i/\mathsf{s}$ for $i \in [\mathsf{s}]_0$). Henceforth, when we write γ_{s} we naturally mean γ_{r} with r being the spectrum defined by the indicator s.

Similar to all existing algorithms, the PAC algorithm relies on the technique of heavy-light decomposition in which (partial) tuples are labeled as heavy or light based on their degrees. Given an instance $\mathcal{D}$, the spectrum indicator s divides the range of degrees from $|\mathcal{D}|$ to 1 into $\mathsf{s}+1$ discrete sequence of ranges: $\mathsf{rng}(0), \ldots, \mathsf{rng}(\mathsf{s})$ such that the degrees in $\mathsf{rng}(i)$ are lighter (i.e., lower) than degrees in $\mathsf{rng}(j)$ for all $j < i$, in which case the degrees of $\mathsf{rng}(\mathsf{s})$ are considered *light* while all the rest are considered different levels of *heavy*. Having these ranges, $\mathcal{D}$ is divided into complementary *fragments* $\mathcal{F}$ such that for every set A of attributes from $\mathsf{ins}(q)$, all the A-tuples in $\mathcal{F}$ have similar degrees in $\mathcal{D}$ (i.e., within the same range).

Configurations provide all the degree information relevant to a particular fragment. A *configuration* $\mathcal{C}$ of a query q and spectrum indicator s is a total mapping $\mathcal{C} : \mathsf{ins}(q) \to [\mathsf{s}]_0$. Note that the value assigned to a set A of attributes by a configuration $\mathcal{C}$, say $\mathcal{C}(A) = i$, means that the degrees of all A-tuples in the corresponding fragment are in the ith range (that is, $\mathsf{rng}(i)$). In what follows, we will use $\mathsf{fragment}(\mathcal{C})$ to refer to the unique fragment $\mathcal{F}$ of $\mathcal{D}$ corresponding (i.e., defined by) the configuration $\mathcal{C}$.

We remark that not all mappings from $\mathsf{ins}(q)$ to $[\mathsf{s}]_0$ are equally meaningful. For this, we define the notion of valid configuration.

▶ **Definition 16.** *A configuration $\mathcal{C}$ of q and s is* valid *if $\mathcal{C}(B) \leq \mathcal{C}(A)$ for every pair of sets $A, B \in \mathsf{ins}(q)$ with $B \subseteq A$, and, if $\mathcal{C}(A) = \mathsf{s}$ for every full set $A \in \mathsf{ins}(q)$. The set of all valid configurations of q and s is denoted by $\mathsf{conf}_{\mathsf{s}}(q)$.*

The first condition states that the degrees of tuples over a set of attributes cannot be lighter than any of its extensions. The second condition is a consequence of having instances defined as sets. Thus, it is straightforward to show that, for every invalid configuration $\mathcal{C}$, $[\![q]\!]_{\mathcal{F}} = \emptyset$ where $\mathcal{F} := \mathsf{fragment}(\mathcal{C})$. Therefore, from now on, we only consider valid configurations.

Solution procedure and its load analysis

Note that the value for s is assumed to be fixed beforehand. The PAC algorithm then runs in parallel on all the fragments and hence the load of the algorithm is determined by the most difficult fragment. That is, $\gamma_{\mathsf{s}} := \max_{\mathcal{F}} \mathsf{cost}(\mathcal{F})$. For a particular fragment $\mathcal{F}$, this cost measure intuitively means that there is a way with which the algorithm runs over the fragment $\mathcal{F}$ with load $\tilde{\mathcal{O}}\left(|\mathcal{D}|/p^{1/\gamma_{\mathsf{s}}}\right)$ using at most $p^{\mathsf{cost}(\mathcal{F})/\gamma_{\mathsf{s}}} \leq p$ servers.

Naturally, the query q can be computed over a fragment $\mathcal{F}$ in various ways (i.e., parameterization), each is referred to as a *solution*. In that case, the load of computing q over $\mathcal{F}$ is simply determined by the most efficient solution where $\mathsf{cost}(\mathcal{F}) := \min_{\mathcal{S}} \mathsf{cost}(\mathcal{S})$ with $\mathcal{S}$ here ranging over *compatible* solutions only. Before we explain what solutions are, and what compatible means, we define some additional concepts. Henceforth, for a fragment $\mathcal{F}$ of $\mathcal{D}$ corresponding to a configuration $\mathcal{C}$, we define

- $\mathsf{light}(\mathcal{F})$ to denote the set of all sets $A \in \mathsf{ins}(q)$ such that A is light in $\mathcal{F}$ (i.e., $\mathcal{C}(A) = \mathsf{s}$);
- $\mathsf{heavy}(\mathcal{F})$ to refer to the set of $\mathsf{ins}(q) \setminus \mathsf{light}(\mathcal{F})$; i.e., all the heavy sets of attributes; and
- $\mathsf{heavy\text{-}rel}(\mathcal{F}) := \{R \in \mathsf{rels}(q) \mid \forall \mathbf{a} \in \mathsf{sch}_q(R) : \{\mathbf{a}\} \in \mathsf{heavy}(\mathcal{F})\}$.

To define what $\mathcal{S}$ is, it is more intuitive to introduce its components from the computation of q over a fragment $\mathcal{F}$. Thereto, to compute $[\![q]\!]_{\mathcal{F}}$, the following phases will be followed:

Broadcast phase: broadcast all relations in $\mathsf{heavy\text{-}rel}(\mathcal{F})$ to all servers. Now, the query q can be simplified to the query $q' := \mathsf{sub}_q(\mathcal{R}_{\mathcal{F}}, \emptyset)$ with $\mathcal{R}_{\mathcal{F}}$ being the relations of $\mathsf{heavy\text{-}rel}(\mathcal{F})$.

Decomposing phase: choose a subset $\mathcal{H}$ from $\mathsf{heavy}(\mathcal{F})$ such that the sets of $\mathcal{H}$ are pairwise disjoint and choose a subset $\mathcal{H}' \subseteq \mathcal{H}$. For each set $A \in \mathcal{H}'$, choose a set $B_A \in \mathsf{light}(\mathcal{F})$ of attributes with $A \subsetneq B_A$. Henceforth, we use the terms of *patches* and *anchors* to mean the sets $\mathcal{P} := \mathcal{H} \setminus \mathcal{H}'$ and $\mathcal{A} := \{(B_A, A) \mid A \in \mathcal{H}'\}$, respectively. Let $\mathcal{R}_{\mathcal{A}}$ be the set $\bigcup_{A \in \mathcal{H}'} \mathsf{rels}_{q'}(B_A)$ of relations from $\mathsf{rels}(q)$ and $\mathcal{R}$ be the union $\mathcal{R}_{\mathcal{F}} \cup \mathcal{R}_{\mathcal{A}}$.
The query q' is decomposed into a set $\mathcal{Q}$ of subqueries: for each anchor $(B_A, A) \in \mathcal{A}$, the relations of $\mathsf{rels}_{q'}(B_A)$ make one subquery; all the remaining relations in q' make one last subquery, which is $\mathsf{sub}_{q'}(\mathcal{R}_{\mathcal{A}}, \emptyset) = \mathsf{sub}_q(\mathcal{R}, \emptyset)$ in this case.

Partitioning phase: for each tuple $\boldsymbol{h}$ in $\mathcal{F}$ over $\mathsf{elem}(\mathcal{H})$, we assign a group of $p_{\boldsymbol{h}}$ servers to compute the residual query of q', obtained by removing the attributes from $\mathsf{elem}(\mathcal{H})$, over the sub-fragment $\mathcal{F}_{[\boldsymbol{h}]}$. Let $q'' := q'_{\mathsf{elem}(\mathcal{H})} = \mathsf{sub}_q(\mathcal{R}_{\mathcal{F}}, \mathsf{elem}(\mathcal{H}))$.

Semi-join phase: over each group of $p_{\boldsymbol{h}}$ servers, we apply semi-joins to simplify the query q'' from all the redundancies to obtain the query $q''' := \lfloor q'' \rfloor = \lfloor \mathsf{sub}_q(\mathcal{R}_{\mathcal{F}}, \mathsf{elem}(\mathcal{H})) \rfloor$.

Final join phase: over each group of $p_{\boldsymbol{h}}$ servers, we compute the join of all relations of q''' over the instance $\left\lfloor \mathcal{F}_{[\boldsymbol{h}]} \right\rfloor$. In this phase, the remaining relations in q''' relevant for each subquery of $\mathcal{Q}$ are handled separately. In which case, the relations of the subquery $\mathsf{sub}_{q'''}(\mathcal{R}_{\mathcal{A}}, \emptyset)$ are joined using HyperCube algorithm parameterized by a weight assignment f. Henceforth, we use q_f to refer to the subquery $\mathsf{sub}_{q'''}(\mathcal{R}_{\mathcal{A}}, \emptyset)$.

With this procedure in mind, $\mathcal{S}$ is defined as $\mathcal{S} := (\mathcal{P}, \mathcal{A}, f)$ [1]. For each such solution $\mathcal{S}$, the cost is defined as

$$\mathsf{cost}(\mathcal{S}) := \underbrace{\left(\left(\sum_{A \in \mathcal{P}} \mathcal{C}(A)\right) + |\mathcal{P}|\right)}_{\mathsf{cost}(\mathcal{P})} / \mathsf{s} + \underbrace{|\mathcal{A}|}_{\mathsf{cost}(\mathcal{A})} + \underbrace{f(\mathsf{atts}(q_f))}_{\mathsf{cost}(f)}.$$

We remark that the load induced by some choices of $\mathcal{S}$ cannot be bounded, as there are conditions that f needs to satisfy to have a *compatible* solution. According to [1], a solution $\mathcal{S}$ to a fragment $\mathcal{F}$ corresponding to a configuration $\mathcal{C} \in \mathsf{conf}_{\mathsf{s}}(q)$ is said to be *compatible* if f is a frac. vertex cover for q_f of weight ≥ 1 with $f(A) \leq \mathcal{C}(A)/\mathsf{s}$ for every heavy set A.[7] The following has been shown to compatible solutions (from which Theorem 14 directly follows):

▶ **Theorem 17** (Theorem 14 of [1]). *Let $\mathcal{S}$ be a compatible solution for a fragment $\mathcal{F}$ of $\mathcal{D}$. Then $[\![q]\!]_{\mathcal{F}}$ is computable using p servers with load $\tilde{\mathcal{O}}\left(|\mathcal{D}|/p^{1/\mathsf{cost}(\mathcal{S})}\right)$ w.h.p.*

Limitation

The requirement that f has to be a fractional vertex cover for q_f is not necessary and it can be made tighter in some cases. Thereto, for a given configuration $\mathcal{C} \in \mathsf{conf}_{\mathsf{s}}(q)$ and an A-tuple $\boldsymbol{t}$, we define the *size-reduction induced by* $\boldsymbol{t}$, denoted $\Delta_{\mathcal{C},\boldsymbol{t}}$, to be the size reduction of q with, for every $R \in \mathsf{rels}(q)$, $\Delta_{\mathcal{C},\boldsymbol{t}}(R) := \mathcal{C}(A_R)/\mathsf{s}$ such that A_R is the set $A \cap \mathsf{sch}_q(R)$.

We now observe and claim that after the partitioning phase in the aforementioned procedure, there exists some value λ such that the (sub)fragment $\mathcal{F}_{[\boldsymbol{h}]}$ is compatible with the size reduction $\Delta_{\mathcal{C},\boldsymbol{h}}$ and λ. We formulate and prove this claim in the full version. Nonetheless, this observation allows us to obtain a tighter analysis in some cases. Thereto, we say that a solution $\mathcal{S} := (\mathcal{P}, \mathcal{A}, f)$ to $\mathcal{F}$ corresponding to a configuration $\mathcal{C}$ is said to be *reduced* if f is an edge-weighted frac. vertex cover for q_f and the size reduction $\Delta_{\mathcal{C},\boldsymbol{h}}$, of weight ≥ 1 and, with $f(A) + \Delta_{\mathcal{C},\boldsymbol{h}}(R) \leq \mathcal{C}(A)/\mathsf{s}$ for every heavy set A and every $R \in \mathsf{rels}_{q_f}(A)$. We show:

▶ **Theorem 18.** *Let $\mathcal{S}$ be a reduced solution for a fragment $\mathcal{F}$ of $\mathcal{D}$. Then $[\![q]\!]_{\mathcal{F}}$ is computable using p servers with load $\tilde{\mathcal{O}}\left(|\mathcal{D}|/p^{1/\mathsf{cost}(\mathcal{S})}\right)$ w.h.p.*

Finally remark that all compatible solutions are reduced but not all reduced solutions are compatible. Since Theorem 18 is analogous to Theorem 17, we obtain that $\mathsf{cost}(\mathcal{F})$ in some cases is strictly less compared to considering compatible solutions only. For instance, this will be essential in establishing the (optimal) upper bound of some grid queries in the following section.

5 New upper bounds

In this section, we show all the upper bounds we claimed in the introduction. To show that the worst-case load of computing a query q over any instance $\mathcal{D}$ using p servers is asymptotically upper bounded by $|\mathcal{D}|/p^{1/\ell}$, we choose a value for the spectrum indicator s and then show that for every fragment $\mathcal{F}$ of $\mathcal{D}$, $\mathsf{cost}(\mathcal{F}) \leq \ell$. That is, we argue that every fragment has at least one compatible or reduced solution $\mathcal{S}$ with $\mathsf{cost}(\mathcal{S}) \leq \ell$.

Degree-two algorithm

For all degree-two queries q, we set $\ell = k/2$ where $k = |\mathsf{atts}(q)|$ and we choose $\mathsf{s} = 2$. We next show

▶ **Proposition 19.** *Every fragment $\mathcal{F}$ corresponding to a configuration $\mathcal{C}$ has $\mathsf{cost}(\mathcal{F}) \leq k/2$.*

[7] The condition of $f(A) \leq \mathcal{C}(A)/\mathsf{s}$ for all heavy sets A guarantees that applying `HyperCube` with the shares of f to the attributes make the relations skew-free.

Thereto, let $\mathcal{C}$ be an arbitrary configuration from $\mathsf{conf_s}(q)$ and let $\mathcal{F}$ be the corresponding fragment to $\mathcal{C}$. The image of $\mathcal{C}$ in this case is $\subseteq \{0, 1, 2\}$ where 0 corresponds to *heavy*, 1 corresponds to *semi-heavy*, and 2 corresponds to *light*.

Notice that we only consider relations of q with at least one attribute $\mathbf{a}$ with $\mathcal{C}(\{\mathbf{a}\}) = \mathsf{s} = 2$. All other relations are going to be broadcast in the first phase of the algorithm.

An attribute $\mathbf{a}$ is called *isolated* if it is light and all its adjacent attributes are individually heavy. We use $\mathcal{I}$ to refer to set of all isolated attributes. Now, we differentiate between:

- **case 1**: every non-broadcastable relation contains an isolated attribute.
- **case 2**: there are non-broadcastable relations without isolated attributes.

For **case 1**, we choose a solution $\mathcal{S}$ for $\mathcal{F}$ with empty sets for $\mathcal{P}$ and $\mathcal{A}$ and with a weight mapping f such that $f(\mathbf{a}) = 1$ for every light attribute (i.e., all the $\mathcal{I}$ attributes) and $f(\mathbf{a}) = 0$ for every other attribute. It is easy to see that $\mathcal{S}$ is a compatible solution.

Observe that, in this case, $\mathsf{cost}(\mathcal{S}) = \mathsf{cost}(f) = |\mathcal{I}|$. To complete the argument for this case, we need to show that $|\mathcal{I}| \leq k/2$, which follows from the next lemma and the fact that no two attributes of $\mathcal{I}$ can be adjacent by definition.

▶ **Lemma 20.** *Every vertex packing for q is of size $\leq k/2$.*

As for **case 2**, take $\mathcal{R}$ to be the set of all pairs of adjacent attributes $(\mathbf{a}, \mathbf{b})$ such that $\mathbf{a} \in \mathcal{I}$. We call this set $\mathcal{R}$ as anchor candidates. Now, choose a subset $\mathcal{R}'$ of $\mathcal{R}$ of pairwise disjoint pairs such that it covers all the $\mathcal{I}$ attributes. That is, every isolated attribute $\mathbf{a}$ is contained in exactly one pair of $\mathcal{R}'$. Observe that such set $\mathcal{R}'$ must exist by the following lemma :

▶ **Lemma 21.** *Viewing the set $\mathcal{R}$ as an undirected bipartite graph $\mathcal{G}$ between $\mathcal{I}$ and non-$\mathcal{I}$ attributes: there is an edge packing for $\mathcal{G}$ of size equal to $|\mathcal{I}|$. Hence, all $\mathcal{I}$ attributes must be covered by that edge packing.*

We construct the set of anchors from $\mathcal{R}'$ as follows. Take $\mathcal{A} := \{(\{\mathbf{a}, \mathbf{b}\}, \{\mathbf{b}\}) \mid (\mathbf{a}, \mathbf{b}) \in \mathcal{R}'\}$. Obviously, $\{\mathbf{a}, \mathbf{b}\}$ is light and $\mathbf{b}$ is not, as desired. Moreover, notice that $\mathsf{cost}(\mathcal{A}) = |\mathcal{R}'| = |\mathcal{I}|$ and the set of attributes in $\mathcal{A}$ is of size $2 * |\mathcal{I}|$.

Now, take $\mathcal{P}$ to be the set $\{\{\mathbf{a}\} \mid \mathbf{a}$ is heavy and does not appear in $\mathcal{A}\}$. Observe that, since each set of attributes in $\mathcal{P}$ is heavy, we have $\mathsf{cost}(\mathcal{P}) = |\mathcal{P}|/2$ and the set of attributes in $\mathcal{P}$ is of size $|\mathcal{P}|$.

Finally, take f to be the weight mapping with $f(\mathbf{a}) = 1/2$ for every remaining attribute that is either semi-heavy or light but non-isolated. It is easy to see that $\mathsf{cost}(f)$ is equal to half the attributes it assigns weight to. Hence, overall, each attribute contributes to the solution with a cost of at most $1/2$. Thus, $\mathsf{cost}(\mathcal{P}) + \mathsf{cost}(\mathcal{A}) + \mathsf{cost}(f) \leq k/2$ follows from the disjointness of the different components. We argue in the full version that $\mathcal{S}$ is indeed a compatible solution. The previous discussion and Theorem 15 directly entails:

▶ **Theorem 22** (Degree-two Upper Bound). *For every linear degree-two query q and instance $\mathcal{D}$, $[\![q]\!]_{\mathcal{D}}$ is computable over p servers with load $\tilde{\mathcal{O}}\left(|\mathcal{D}|/p^{1/\min\{\psi, k/2\}}\right)$ w.h.p.*

As a direct consequence of the previous theorem and Theorem 10, we obtain:

▶ **Corollary 23.** *For every semi-binary query q with $\tau = k/2$ and without odd-like cycles and instance $\mathcal{D}$, $[\![q]\!]_{\mathcal{D}}$ is computable over p servers with load $\tilde{\mathcal{O}}\left(|\mathcal{D}|/p^{1/\tau}\right)$ w.h.p, which is optimal.*

Grid algorithms

In this part, we show the following main result:

▶ **Theorem 24** (Grids Upper Bound). *For every* $\mathtt{r} \times \mathtt{c}$ *grid query* q *and every instance* $\mathcal{D}$, $[\![q]\!]_{\mathcal{D}}$ *is computable over* p *servers with load* $\tilde{\mathcal{O}}\left(|\mathcal{D}|/p^{1/\max\{\tau,\tau+\rho-3\}}\right)$ *w.h.p.*

Without loss of generality, we assume that $\mathtt{r} \geq \mathtt{c}$. Hence, by Proposition 3, the upper bound in the statement can be equivalently expressed as $\tilde{\mathcal{O}}\left(|\mathcal{D}|/p^{1/\max\{\mathtt{r},\mathtt{r}+\mathtt{c}-3\}}\right)$ instead. To establish Theorem 24, we set $\ell = \max\{\mathtt{r}, \mathtt{r}+\mathtt{c}-3\}$. We continue our analysis by distinguishing two cases based on grid dimensions.

First, consider the case of grids with $\mathtt{c} = 2$. This is precisely the case when q is a boat query. By Lemma 4 and Theorem 15, we obtain the desired result by setting $\mathtt{s} = 1$ since $\ell = \max\{\mathtt{r}, \mathtt{r}+\mathtt{c}-3\} = \mathtt{r} = \tau = \psi$.

Now, consider grids with $\mathtt{c} \geq 3$. We observe that in this case, it would be sufficient to only consider fragments for which a solution with a fractional vertex cover alone cannot trivially solve the query with the desired load. For this, we define in Appendix A.2 the notion of a *difficult fragment* of a grid query. In the full version, we give the specific arguments for $\mathtt{r} = 3$ and $\mathtt{r} \geq 4$ separately to finalize the proof of Theorem 24, where we use different values for the spectrum indicator, and moreover, we utilize completely different solution strategies to solve the difficult fragments. As for $\mathtt{r} = 3$, we use a spectrum defined by the indicator $\mathtt{s} = 6$, and we utilize multiple solution strategies in which case none uses anchors. On the other hand, for $\mathtt{r} \geq 4$, we find that a spectrum indicator of $\mathtt{s} = 1$ is sufficient to obtain the upper bound we claim, and all solution strategies, we use, involve anchors.

As a result of Theorem 7 and Theorem 24, we directly obtain:

▶ **Corollary 25.** *For every* $\mathtt{r} \times \mathtt{c}$ *grid query* q *with* $\min\{\mathtt{r}, \mathtt{c}\} \leq 3$ *and every instance* $\mathcal{D}$, $[\![q]\!]_{\mathcal{D}}$ *is computable over* p *servers with load* $\tilde{\mathcal{O}}\left(|\mathcal{D}|/p^{1/\tau}\right)$ *w.h.p, which is optimal.*

6 Conclusion

In this work, we have studied worst-case communication complexity of the join evaluation problem for the class of degree-two queries and some of its fragments in the MPC model. We have established several new lower bounds and upper bounds for the problem for which some bounds are shown to be worst-case optimal. Although, all our upper bound results have been established through different instantiations of the PAC algorithm, it is remarkable that several techniques, that depended on the structure of the query, were needed. This shows that degree-two queries are very diverse and understanding their load complexity is crucial to eventually obtain distributed join algorithms with optimality guarantees for all queries.

Moreover, by showing a lower bound on the load for grid queries that is larger than both, the load based on the fractional vertex covering number τ, and on fractional vertex packing number ρ, we showed that neither τ nor ρ characterizes the worst-case optimal load. This refutes a well-believed conjecture [6] that the worst-case load in the constant-round MPC model is entirely captured by either quantities.

Our work poses several new questions and revives existing open problems. Without a doubt, the most important problem still is: *what is it then the most accurate query-related quantity which captures the worst-case optimal communication complexity of the join evaluation problem?*

References

1 Heba Aamer and Bas Ketsman. PAC: Computing Join Queries with Semi-Covers. In Sudeepa Roy and Ahmet Kara, editors, *28th International Conference on Database Theory (ICDT 2025)*, volume 328 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICDT.2025.6.

2 Foto N. Afrati and Jeffrey D. Ullman. Optimizing multiway joins in a map-reduce environment. *IEEE Trans. Knowl. Data Eng.*, 23(9):1282–1298, 2011. doi:10.1109/TKDE.2011.47.

3 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. In Richard Hull and Wenfei Fan, editors, *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA, 2013*, pages 273–284. ACM, 2013. doi:10.1145/2463664.2465224.

4 Paul Beame, Paraschos Koutris, and Dan Suciu. Skew in parallel query processing. In Richard Hull and Martin Grohe, editors, *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 212–223. ACM, 2014. doi:10.1145/2594538.2594558.

5 Xiao Hu. Cover or pack: New upper and lower bounds for massively parallel joins. In Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo, editors, *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, 2021*, pages 181–198. ACM, 2021. doi:10.1145/3452021.3458319.

6 Xiao Hu and Ke Yi. Instance and output optimal parallel algorithms for acyclic joins. In *PODS*, pages 450–463, 2019. doi:10.1145/3294052.3319698.

7 Manas Joglekar and Christopher Ré. It's all a matter of degree. *Theor. Comp. Sys.*, 62(4):810–853, May 2018. doi:10.1007/s00224-017-9811-8.

8 Bas Ketsman and Dan Suciu. A worst-case optimal multi-round algorithm for parallel computation of conjunctive queries. In Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts, editors, *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 417–428. ACM, 2017. doi:10.1145/3034786.3034788.

9 Bas Ketsman, Dan Suciu, and Yufei Tao. A near-optimal parallel algorithm for joining binary relations. *Log. Methods Comput. Sci.*, 18(2), 2022. doi:10.46298/lmcs-18(2:6)2022.

10 Paraschos Koutris, Paul Beame, and Dan Suciu. Worst-case optimal algorithms for parallel query processing. In Wim Martens and Thomas Zeume, editors, *19th International Conference on Database Theory, ICDT 2016*, volume 48 of *LIPIcs*, pages 8:1–8:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICDT.2016.8.

11 Miao Qiao and Yufei Tao. Two-attribute skew free, isolated cp theorem, and massively parallel joins. In *PODS*, pages 166–180, 2021. doi:10.1145/3452021.3458321.

12 Yufei Tao. A simple parallel algorithm for natural joins on binary relations. In Carsten Lutz and Jean Christoph Jung, editors, *23rd International Conference on Database Theory, ICDT 2020*, volume 155 of *LIPIcs*, pages 25:1–25:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICDT.2020.25.

13 Yufei Tao. Parallel acyclic joins with canonical edge covers. In Dan Olteanu and Nils Vortmeier, editors, *25th International Conference on Database Theory, ICDT 2022*, volume 220 of *LIPIcs*, pages 9:1–9:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICDT.2022.9.

A Some proofs

A.1 Proof of Proposition 3

Before we prove Proposition 3, we first give two helpful lemmas that give upper bounds on the fractional vertex covers of the different residual queries of a grid query. These lemmas will be helpful to obtain the value of ψ for grids.

Firstly:

▶ **Lemma 26.** *Let q be an $\mathtt{r} \times \mathtt{c}$ grid query with $\mathtt{r}, \mathtt{c} \geq 2$. Let A be a set of attributes from $\mathsf{atts}(q)$. Suppose there exist integers i and j with $i \in [\mathtt{r}]$ and $j \in [\mathtt{c}]$ such that every attribute $\mathbf{a}$ of the residual query q_A either belongs to the ith row relation (i.e., $\mathbf{a}$ is of the form $\mathbf{a}_{(i,w)}$) or belongs to the jth column relation (i.e., $\mathbf{a}$ is of the form $\mathbf{a}_{(w,j)}$). Now, let $B_{(i,*)}$ and $B_{(*,j)}$ be the set of attributes of q_A belonging to the ith row relation and the jth column relation, respectively. Now, if either $B_{(i,*)} \setminus \{\mathbf{a}_{(i,j)}\}$ or $B_{(*,j)} \setminus \{\mathbf{a}_{(i,j)}\}$ is empty, then take B to be the set of attributes defined as $B := B_{(i,*)} \cup B_{(*,j)}$. Otherwise, take B to be the set with $B := \big(B_{(i,*)} \cup B_{(*,j)}\big) \setminus \{\mathbf{a}_{(i,j)}\}$. Then, $\tau(q_A) = |B| \leq \mathtt{r} + \mathtt{c} - 2$.*

Proof. Without loss of generality, assume that both i and j are equal to 1; hence, the row relation and the column relation of the statement are $R_{(1,*)}$ and $R_{(*,1)}$. Accordingly, we know that $A \supseteq \{\mathbf{a}_{(u,v)} \mid 2 \leq u \leq \mathtt{r}, 2 \leq v \leq \mathtt{c}\}$. Moreover, notice that the set of attributes $B_{(*,1)} \subseteq \{\mathbf{a}_{(u,1)} \mid 1 \leq u \leq \mathtt{r}\}$ and that $B_{(1,*)} \subseteq \{\mathbf{a}_{(1,v)} \mid 1 \leq v \leq \mathtt{c}\}$.

Since both sets form the entire set of attributes in q_A, it is trivial to argue that

$$\tau(q_A) \leq |B_{(1,*)} \cup B_{(*,1)}| \tag{1}$$

Now, observe that every attribute $\mathbf{a}_{(u,1)}$ (with $u \geq 2$) in the first set ($B_{(*,1)}$) is the only remaining attribute in the row relation $R_{(u,*)}$; similarly, every attribute $\mathbf{a}_{(1,v)}$ (with $v \geq 2$) in the second set ($B_{(1,*)}$) is the only remaining attribute in the column relation $R_{(*,v)}$. Accordingly, any valid fractional vertex cover f for q_A in this case must assign a weight of 1 to each such attribute. Thus,

$$\tau(q_A) \geq |B \setminus \{\mathbf{a}_{(1,1)}\}| \tag{2}$$

If $\mathbf{a}_{(1,1)} \in A$ and hence it is not an attribute in q_A, then $B \setminus \{\mathbf{a}_{(1,1)}\} = B = B_{(1,*)} \cup B_{(*,1)}$. From Equations (1) and (2), we direction obtain that $\tau(q_A) = |B|$. Notice that in this case, the maximal possible set for B occurs when $B_{(*,1)} = \{\mathbf{a}_{(u,1)} \mid 2 \leq u \leq \mathtt{r}\}$ and $B_{(1,*)} = \{\mathbf{a}_{(1,v)} \mid 2 \leq v \leq \mathtt{c}\}$ which are sets of size $\mathtt{r} - 1$ and $\mathtt{c} - 1$ respectively. Thus, $|B| \leq (\mathtt{r} - 1) + (\mathtt{c} - 1) = \mathtt{r} + \mathtt{c} - 2$ as desired.

Now, suppose that $\mathbf{a}_{(1,1)} \notin A$, and hence, $\mathbf{a}_{(1,1)} \in B_{(1,*)} \cap B_{(*,1)}$. When both $B_{(1,*)}$ and $B_{(*,1)}$ have other attributes, then the aforementioned weight assignment f is already a valid fractional vertex cover for q_A since both $R_{(1,*)}$ and $R_{(*,1)}$ are covered by at least one other attribute from $B_{(1,*)}$ and $B_{(*,1)}$ respectively. Notice that in this case $B = B \setminus \{\mathbf{a}_{(1,1)}\}$. Hence, $\tau(q_A) = |B|$. The argument that $|B| \leq \mathtt{r} + \mathtt{c} - 2$ is similar to that of the previous case.

On the other hand, when either $B_{(1,*)}$ or $B_{(*,1)}$ does not contain any other attribute, the previous weight assignment f cannot be a valid fractional vertex cover as there is either $R_{(1,*)}$ or $R_{(*,1)}$ that is not covered and the only remaining attribute in either relation is the attribute $\mathbf{a}_{(1,1)}$. Thus, f must also assign a weight of 1 to $\mathbf{a}_{(1,1)}$, which is minimal in this case. Accordingly, we obtain that $\tau(q_A) = |B_{(*,1)} \cup B_{(1,*)}|$. Notice that by construction $B = B_{(1,*)} \cup B_{(*,1)} \neq B \setminus \{\mathbf{a}_{(1,1)}\}$. Moreover, the maximal size of B is either $\mathtt{r}$ (when $B_{(*,1)} = \{\mathbf{a}_{(1,1)}\}$) or $\mathtt{c}$ (when $B_{(1,*)} = \{\mathbf{a}_{(1,1)}\}$). In both cases, $|B| \leq \max\{\mathtt{r}, \mathtt{c}\} \leq \mathtt{r} + \mathtt{c} - 2$ as $\mathtt{r}, \mathtt{c} \geq 2$ by definition. ◀

Secondly:

▶ **Lemma 27.** *Let q be an $\mathtt{r} \times \mathtt{c}$ grid query with $\mathtt{r}, \mathtt{c} \geq 2$. Let A be a set of attributes from $\mathsf{atts}(q)$ such that the residual query q_A does not satisfy the conditions of Lemma 26. Then, $\tau(q_A) \leq \max\{\mathtt{r}, \mathtt{c}, \mathtt{r} + \mathtt{c} - 3\}$.*

Proof. We prove the lemma by case distinction.

First, we consider the case that the residual query q_A contains a set B of three attributes with $\{\mathbf{a}_{(i_1,j_1)}, \mathbf{a}_{(i_2,j_2)}, \mathbf{a}_{(i_3,j_3)}\}$ such that $i_1 \neq i_2 \neq i_3$ and $j_1 \neq j_2 \neq j_3$. Take f to be the weight mapping with $f(\mathbf{a}) = 1$ for $\mathbf{a} \in B$. So far f covers 3 different row relations and 3 different column relations. Let q' be the query q_A after removing the 6 covered relations. Notice that $|\mathsf{rels}(q')| \leq \mathtt{r} + \mathtt{c} - 6$ in this case. To cover the rest of the relations of q_A we then need a weight mapping with total weight of at most $\mathtt{r} + \mathtt{c} - 6$. Thus, the total weight mapping for the entire q_A is $\leq \mathtt{r} + \mathtt{c} - 6 + 3 = \mathtt{r} + \mathtt{c} - 3$ as desired.

Now, we continue with the assumption that q_A contains no set B of three attributes with $\{\mathbf{a}_{(i_1,j_1)}, \mathbf{a}_{(i_2,j_2)}, \mathbf{a}_{(i_3,j_3)}\}$ such that $i_1 \neq i_2 \neq i_3$ and $j_1 \neq j_2 \neq j_3$. Furthermore recall that q_A does not satisfy the conditions of Lemma 26. We obtain that the attributes of q_A must be in one of the following two forms:

- the attributes span exactly two row relations and each of the two rows has at least two attributes; or
- analogously, the attributes span exactly two column relations and each of the two columns has at least two attributes.

Without loss of generality, we only give the argument for the first form. Take $\mathcal{I}_1$ to be the set of indices of column relations having an attribute at the first row and similarly take $\mathcal{I}_2$ to be the set of indices of column relations having an attribute at the second row. As mentioned above, each of the two sets $\mathcal{I}_1$ and $\mathcal{I}_2$ is of size ≥ 2. Let $\mathbf{a}_{(i_1,j_1)}$ and $\mathbf{a}_{(i_2,j_2)}$ be two attributes from q_A such that $i_1 \neq i_2$ and $j_1 \neq j_2$. Observe that such two attributes must exist by the aforementioned conditions. Take f to be the weight mapping with $f(\mathbf{a}) = 1$ for every $\mathbf{a} \in \{\mathbf{a}_{(i_1,j)} \mid j \in \mathcal{I}_1, j \neq j_2\} \cup \{\mathbf{a}_{(i_2,j)} \mid j \in \mathcal{I}_2 \setminus \mathcal{I}_1\}$. Notice that this assignment covers all the column relations of q_A. Moreover, since the attributes $\{\mathbf{a}_{(i_1,j_1)}, \mathbf{a}_{(i_2,j_2)}\}$ are assigned 1 by f. Then, this assignment also covers each of the two row relations remaining in q_A. This shows that $\tau(q_A) \leq |\mathcal{I}_1 \cup \mathcal{I}_2| \leq \mathtt{c}$ as desired.

The analogous argument for the second form will result in $\tau(q_A) \leq \mathtt{r}$ as desired. ◀

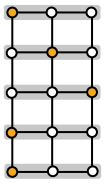
Now, we are in position to prove Proposition 3.

▶ **Proposition 3.** *Let q be an arbitrary $\mathtt{r} \times \mathtt{c}$ grid query with $\mathtt{r}, \mathtt{c} \geq 2$. Then, we have $\tau(q) = \max\{\mathtt{r}, \mathtt{c}\}$, $\rho(q) = \min\{\mathtt{r}, \mathtt{c}\}$, and $\psi(q) = \mathtt{r} + \mathtt{c} - 2$.*

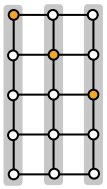
Proof. Without loss of generality, assume that $\mathtt{r} \geq \mathtt{c}$. Each of the constructions of this proof is illustrated in Figure 2.

First, we show that $\tau = \mathtt{r}$. For this, take f to be the weight mapping with $f(\mathbf{a}) = 1$ for $\mathbf{a} \in \{\mathbf{a}_{(i,i)} \mid 1 \leq i \leq \mathtt{c}\} \cup \{\mathbf{a}_{(i,1)} \mid \mathtt{c} < i \leq \mathtt{r}\}$ and all other attributes have $f(\cdot) = 0$. Note that f assigns a weight of 1 to exactly one attribute per row relation and at least one attribute per column relation. Hence, f is a valid vertex cover for q whose total weight is equal to $\mathtt{r}$. Now, observe that the set $\mathcal{R}_{\mathtt{r}}$ of row relations form an edge packing with size $= \mathtt{r}$ as well. This shows optimality for the vertex cover obtained by f and hence $\tau = \mathtt{r}$ as desired.

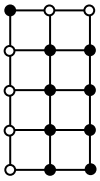
Second, we show that $\rho = \mathtt{c}$. For this, take f to be the weight mapping with $f(\mathbf{a}) = 1$ for $\mathbf{a} \in \{\mathbf{a}_{(i,i)} \mid 1 \leq i \leq \mathtt{c}\}$ and all other attributes has $f(\cdot) = 0$. Note that f is a valid vertex packing in this case with weight equal to $\mathtt{c}$ in which case it assigns 1 to exactly one attribute per column relation and at most one attribute per row relation. To establish the optimality (i.e., maximality) for the vertex packing obtained by f, we remark that the set $\mathcal{R}_{\mathtt{c}}$ of column relations form a valid edge cover whose size is equal to $\mathtt{c}$ as well. Hence, f is optimal and $\rho = \mathtt{c}$ as desired.



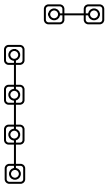
(a) The orange-colored attributes form an optimal vertex cover, while the grey-highlighted relations form an optimal edge packing.



(b) The orange-colored attributes form an optimal vertex packing, while the grey-highlighted relations form an optimal edge cover.



(c) The black-colored attributes form a possible set A of attributes, for which $\tau(q_A)$ is equal to $\psi(q)$.



(d) The residual query q_A. Note that each square in the depiction represents a singleton relation.

Figure 2 Illustrations for τ, ρ, and ψ on the 5×3 grid query.

Finally, we show the last property of $\psi = \mathtt{r}+\mathtt{c}-2$. We begin by showing that $\psi \geq \mathtt{r}+\mathtt{c}-2$. For this, take A to be the set $\{\mathbf{a}_{(1,1)}\} \cup \{\mathbf{a}_{(i,j)} \mid 2 \leq i \leq \mathtt{r}, 2 \leq j \leq \mathtt{c}\}$ of attributes. The residual query q_A consists of $\mathtt{r}+\mathtt{c}-2$ attributes in this case; those are formed by the two sets of $\{\mathbf{a}_{(i,1)} \mid 2 \leq i \leq \mathtt{r}\}$ and $\{\mathbf{a}_{(1,j)} \mid 2 \leq j \leq \mathtt{c}\}$. Notice that our choice for A (and hence, the query q_A) satisfies the conditions of Lemma 26 with $|B| = \mathtt{r}+\mathtt{c}-2$. Thus, we obtain that $\tau(q_A) = \mathtt{r}+\mathtt{c}-2$ as desired.

The other direction ($\psi \leq \mathtt{r}+\mathtt{c}-2$) follows straightforwardly from Lemma 26 and Lemma 27 and the fact that $\max\{\mathtt{r}, \mathtt{c}, \mathtt{r}+\mathtt{c}-3\} \leq \mathtt{r}+\mathtt{c}-2$ (recall $\mathtt{r}, \mathtt{c} \geq 2$). ◀

A.2 Difficult grid fragments

In this section, we define the notion of a *difficult fragment* of a grid query (regardless of the value indicator chosen for $\mathtt{s}$).

▶ **Definition 28.** *Let $\mathcal{C}$ be a configuration from $\mathsf{conf}_{\mathtt{s}}(q)$ and $\mathcal{F} := \mathsf{fragment}(\mathcal{C})$. The fragment $\mathcal{F}$ is said to be* difficult *if there is a row relation $R_{(i,*)} \in \mathcal{R}_{\mathtt{r}}$ and a column relation $R_{(*,j)} \in \mathcal{R}_{\mathtt{c}}$ such that:*

- *every $R_{(i,*)}$ attribute $\mathbf{a}_{(i,w)}$ with $w \neq j$ is light;*
- *every $R_{(*,j)}$ attribute $\mathbf{a}_{(w,j)}$ with $w \neq i$ is light; and*
- *every attribute $\mathbf{a}_{(u,v)}$ with $u \neq i$ and $v \neq j$ is heavy.*

An example of such fragment can be found in Figure 2d.

Observe that indeed such fragments $\mathcal{F}$ are difficult to solve in the following sense:

▶ **Lemma 29.** *Suppose that H is the set $\mathsf{elem}(\mathsf{heavy}(\mathcal{F}))$ of attributes. Then, $\tau(q_H) > \mathtt{r}+\mathtt{c}-3$ if and only if $\mathcal{F}$ is a difficult fragment.*

The proof of the lemma follows from the discussion about the value of ψ for grids of Proposition 3, which can be found in Appendix A.1.