Parallel Communication Obliviousness: One Round and Beyond

YUFEI TAO, RU WANG, and SHIYUAN DENG, The Chinese University of Hong Kong, China

This paper studies *communication-oblivious* algorithms under the massively parallel computation (MPC) model. The communication patterns of these algorithms follow a distribution dependent only on the definition of the underlying problem, the problem size N, and the number p of machines, but not on the specific input elements. Our objective is to understand when obliviousness necessitates — or does not necessitate — heavier communication compared to the traditional MPC model that does not enforce such a requirement.

The first part of our investigation focuses on single-round algorithms. We prove that *skew-free hashing*, a fundamental problem solvable with load $\tilde{O}(N/p)$ (with high probability or w.h.p. for short) under the traditional model, demands a load of nearly $\Omega(N)$ under communication obliviousness. Intriguingly, we show that hashing can still be applied in an oblivious manner to process any natural join in one round with a load complexity matching that of the best traditional MPC algorithm. The second part of our investigation studies *compilation methods* that convert a traditional MPC algorithm $\mathcal{A}$ into a communication-oblivious counterpart. Given an $\mathcal{A}$ that operates within $\ell = \text{poly}(p)$ rounds and entails a load at most $L = \Omega(p \log p)$ w.h.p., we can produce w.h.p. a communication-oblivious version running in 2ℓ rounds with a load at most $(1 + \delta)L$, where $\delta > 0$ can be an arbitrarily small constant. Additionally, we establish hardness results indicating that the theoretical guarantees of our compilation can no longer be significantly improved.

CCS Concepts: • **Theory of computation → Massively parallel algorithms**.

Additional Key Words and Phrases: Massively Parallel Computation; Communication Obliviousness; Theory

ACM Reference Format:
Yufei Tao, Ru Wang, and Shiyuan Deng. 2024. Parallel Communication Obliviousness: One Round and Beyond. *Proc. ACM Manag. Data* 2, 5 (PODS), Article 214 (November 2024), 24 pages. https://doi.org/10.1145/3695832

Acknowledgments

This work was supported in part by GRF projects 14207820, 14203421, and 14222822 from HKRGC.

1 Introduction

In the digital era, cloud computing has emerged as a pivotal part of modern business infrastructure, reshaping the way organizations store, access, and manage data. A growing number of sectors — ranging from healthcare and finance to academia and entertainment — are harnessing the power of cloud services to enhance operational efficiency, effectuate cost savings, and fuel innovation. However, the migration of data from local systems to remote servers operated by third-party service providers has ushered in new privacy concerns. The crux of these concerns is the fact that sensitive data, once stored on these servers, is no longer fully under the user's control. Instead, it falls within the stewardship of service providers, thereby raising critical questions about data privacy.

Authors' Contact Information: Yufei Tao, taoyf@cse.cuhk.edu.hk; Ru Wang, rwang21@cse.cuhk.edu.hk; Shiyuan Deng, sydeng@cse.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong, Shatin, China.

ACM 2836-6573/2024/11-ART214
https://doi.org/10.1145/3695832

Two primary topics come to the forefront in the discourse on privacy control in cloud computing:

- **Confidential computing:** This aims to mitigate two forms of privacy leakage that may occur on an *individual* server. First, rogue administrators could exploit their positions to inspect sensitive data within the server. Second, even if encryption hinders direct data accesses, administrators could still infer valuable information from the memory access patterns that the computation generates.
- **Oblivious communication:** This aims to prevent another two forms of privacy leakage that may occur during communication across *different* servers. First, malicious network managers might scrutinize the data flowing through the gateways. Second, even if the network traffic is encrypted, these managers could still deduce sensitive information from the statistics extracted from the communication.

Robust solutions to confidential computing are already in place, including systems that employ *fully homomorphic encryption* (e.g., IBM's HE4Cloud) or hardware-based *trusted execution environments* (nowadays offered by the leading cloud-computing platforms). When integrated with *oblivious RAM* techniques [5, 11, 20, 21], these systems safeguard sensitive information throughout the entire lifecycle of data at an individual server: whether stored in the disk, transiting through the memory hierarchy, or in use by the CPU.

In this work, we will concentrate on oblivious communication, with the aim to understand how much communication is needed to enforce this requirement. To pave the foundation of our discussion, next we will formalize the concept of oblivious communication and the class of algorithms to be studied.

1.1 Communication-Oblivious MPC

Let us first clarify some math conventions used in this paper. Given positive integers x and y, let $\binom{x}{y}$ be the number of ways to choose y elements from a set of size x; specially, if $y > x$, define $\binom{x}{y} = 0$. For an integer $x \geq 1$, the notation $[x]$ denotes the set $\{1, 2, ..., x\}$. We use double-curly braces to represent multi-sets, e.g., $\{\{1, 1, 1, 2, 2, 3\}\}$ is a multi-set with 6 elements. Sets $S_1, S_2, ..., S_n$ ($n \geq 2$) form a *partition* of set S if $S_i \cap S_j = \emptyset$ for any distinct $i, j \in [n]$ and $\bigcup_{i=1}^{n} S_i = S$. Every $\log(\cdot)$ has base 2. The notation $\text{poly}(n)$ denotes the class of functions polynomial in n.

The MPC Model. Our analysis will be under the *massively parallel computation* (MPC) model, which has been extensively adopted in the database literature, as will be surveyed in Section 1.3.

In this model, p share-nothing machines are connected via a high-speed network. Initially, the input data is distributed across the p machines. An algorithm then executes in *rounds*, each consisting of two phases: in the first phase, every machine performs computation on its local data, while in the second phase, the machines exchange messages via the network. Importantly, if machine $i \in [p]$ plans to send machine $j \in [p]$ a message (in the second phase), machine i must prepare the message in the first phase. This restriction prevents machine i from, for example, deciding what to send based on the messages received *during* the second phase. The *load of a round* is the maximum number of words communicated (sent and received combined) by one machine in that round. The algorithm's performance is measured by two metrics: (i) the total number of rounds executed, and (ii) the maximum load of all rounds, which is termed as the algorithm's (overall) *load*.

Randomization is modeled by introducing a sequence of random bits, which is agreed upon by all machines. Such an agreement can be reached before receiving the input and therefore does not require communication during the algorithm's execution. In the first phase of a round, each machine can guide its local computation according to the random-bit sequence. When a random

event is said to occur "with high probability" (w.h.p.), we require that the probability must be at least $1 - 1/p^c$ where c can be an arbitrarily large constant chosen before seeing the input data.

The size N of the input data is commonly assumed to be significantly larger than p, which typically means $N \geq p^c$ where c is a constant dependent on the problem studied.

Communication-Oblivious Algorithms. We will now formalize the notion of communication obliviousness, building upon a similar formulation in [10].

We consider the existence of a countably infinite set called the *input domain*, whose concrete definition depends on the problem we aim to study. The members of the input domain are referred to as *inputs*. Every input $\mathcal{I}$ is associated with an integer *size*. While the meaning of input size depends on the underlying problem, it should always be proportional to the number of words needed to describe $\mathcal{I}$. Denote by $\mathscr{I}(N)$ the collection of all possible inputs with the same size N.

Fix an input size $N \geq p$ (where p is the number of machines in the MPC model) and an algorithm $\mathcal{A}$. We demand that every input of size N should be presented to $\mathcal{A}$ with exactly the same number of words stored on each machine initially. For this purpose, we assume a function $W(N)$ (determined by the underlying problem) mapping N to a positive integer such that $W(N) = \Theta(N/p)$. For each input $\mathcal{I} \in \mathscr{I}(N)$, we call $\Sigma = (\Sigma_1, \Sigma_2, ..., \Sigma_p)$ a *legal initial state of* $\mathcal{I}$ if

- each Σ_i ($i \in [p]$) is a sequence of $W(N)$ words;
- the p sequences $\Sigma_1, \Sigma_2, ...,$ and Σ_p collectively describe the input $\mathcal{I}$ with no ambiguity;
- placing (an encrypted version of) Σ_i on machine i for each $i \in [p]$ is permitted by the underlying problem.

Each input $\mathcal{I}$ can have multiple legal initial states, and the algorithm $\mathcal{A}$ must be designed to work with all of them. Specifically, when the algorithm is executed on a legal initial state $\Sigma = (\Sigma_1, \Sigma_2, ..., \Sigma_p)$, machine $i \in [p]$ sees Σ_i in the first phase in the first round (of $\mathcal{A}$). From there, the algorithm then performs the necessary computation and communication to solve the problem with strong performance guarantees.

Suppose that, when given a legal initial state Σ, an MPC algorithm $\mathcal{A}$ finishes in ℓ rounds where, in the r-th round ($r \in [\ell]$), machine $i \in [p]$ sends message $\mathbf{M}_r[i, j]$ to machine $j \in [p]$ (for $i = j$, $\mathbf{M}_r[i, j]$ is an empty message). Define $\mathbf{L}_r$ as the $p \times p$ matrix where $\mathbf{L}_r[i, j]$ equals the length of $\mathbf{M}_r[i, j]$, measured in the number of words ($\mathbf{L}_r[i, j]$ is always 0 for $i = j$). Every message $\mathbf{M}_r[i, j]$ is encrypted such that an adversary, who observes the communication, can only use the sequence

$$\pi = (\mathbf{L}_1, \mathbf{L}_2, ..., \mathbf{L}_\ell)$$

to infer about $\mathcal{I}$. We refer to π as the *communication pattern* of $\mathcal{A}$ when executed on Σ. For a randomized algorithm $\mathcal{A}$, the sequence π may be a random variable.

The algorithm $\mathcal{A}$ is *communication oblivious* under the input size N if its communication pattern π follows exactly the same probabilistic distribution, regardless of the choice of $\mathcal{I} \in \mathscr{I}(N)$ and the legal initial state of $\mathcal{I}$. We say that $\mathcal{A}$ is a *single-round algorithm* if ℓ is never greater than 1, regardless of the choice of $\mathcal{I}$ and Σ; otherwise, $\mathcal{A}$ is a *multi-round algorithm*.

1.2 Our Contributions

Our primary objective is to discern the inherent connections between the traditional MPC model and its communication-oblivious counterpart. The core is to understand when obliviousness would necessitate a provably higher load than the traditional, non-oblivious, MPC model. Next, we provide an overview of our findings.

Single-Round Hashing. Our first contribution is a hardness result proving that *hashing* — a building brick deployed by many MPC algorithms (in both theory and practice) — is significantly more expensive under communication obliviousness as far as single-round algorithms are concerned.

To pinpoint the source of hardness, we introduce a problem named *skew-free gathering*. Fix arbitrary integers N and p such that N is a multiple of p, and $p \geq t$ where t is a constant integer at least 2. The input collection $\mathscr{I}(N)$ has only a single input $\mathcal{I}$: this is a set of N elements, each of which — denoted as e — is associated with an integer $key(e)$, referred to as the *key* of e. The set of keys fulfills two conditions:

- There is one special key that is possessed by $t \cdot (N/p)$ elements, called the *special elements*.
- Every other key is possessed by exactly one element.

A legal initial state $(\Sigma_1, ..., \Sigma_p)$ of $\mathcal{I}$ meets the following conditions:

- $\Sigma_1, ..., \Sigma_p$ are word sequences with the same length, which is $\Theta(N/p)$;
- each Σ_i ($i \in [p]$) describes a subset $\mathcal{I}_i \subseteq \mathcal{I}$ of size N/p;
- $\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_p$ form a partition of $\mathcal{I}$;
- there are t different machine ids $j \in [p]$ such that $\mathcal{I}_j$ contains nothing but N/p special elements (i.e., the $t \cdot N/p$ special elements are placed on t distinct machines).

The goal of an algorithm is to move all the special elements to one (arbitrary) machine.

In the conventional MPC model, the problem can be solved trivially with load $O(N/p)$: simply ask the t machines where the special elements are initially stored to send their data to a common machine. In contrast, we prove that if a single-round communication-oblivious algorithm is required to succeed with probability at least 2/3, its load must be at least $\Omega(N/p^{1/t})$ with a constant probability. As t increases, the load approaches $\Omega(N)$ up to a factor sub-polynomial in p. More precisely, there does not exist any constant $\epsilon > 0$ such that one can design an algorithm promising a load of $O(N/p^{\epsilon})$ with a constant probability for all constant t. We also show that the lower bound $\Omega(N/p^{1/t})$ can be matched by a deterministic algorithm.

Let us now turn to *skew-free hashing*, where each input in $\mathscr{I}(N)$ is a set $\mathcal{I}$ of N elements, each with an integer key. It is guaranteed that every possible key is possessed by $O(N/p)$ elements of $\mathcal{I}$ (hence, skew "free"). Initially, each machine receives at most $\lceil N/p \rceil$ elements[1]. The goal is to move all the elements of the same key to a common machine. In the traditional MPC model, the problem can be settled (with hashing) in one round using load $\tilde{O}(N/p)$ w.h.p. [8, 9], where $\tilde{O}(\cdot)$ hides a polylog p factor. As skew-free hashing generalizes skew-free gathering[2], our lower bound on the latter indicates that oblivious skew-free hashing demands a load of nearly $\Omega(N)$, thus creating a huge separation from the traditional model.

Single-Round Joins. The hardness of skew-free hashing under communication obliviousness is alarming because the existing one-round join algorithms [3, 4, 7–9, 19] in the (conventional) MPC model depend on a method named[3] `Share` [4, 9], which is closely relevant to hashing. Because the community has nearly resolved the one-round communication complexity of (natural) joins [19], it would be a huge pity if the topic had to be re-opened in the name of obliviousness.

Our second contribution is to show that, fortunately, joins' single-round communication complexities are not affected by obliviousness! To pave the way for a formal discussion, let us first formalize the join problem. Fix an arbitrary integer $N \geq 1$. Let **att** be a sufficiently large finite set, where each element is called an *attribute*. Consider a non-empty set $\mathcal{X} \subseteq$ **att** of attributes. A *tuple*

[1] Rephrased in our framework, each Σ_i describes a subset $\mathcal{I}_i \subseteq \mathcal{I}$ with $|\mathcal{I}_i| \leq \lceil N/p \rceil$. Pad dummy words to Σ_i (if necessary) so that all $\Sigma_1, ..., \Sigma_p$ have the same length, which is $\Theta(N/p)$. The p subsets $\mathcal{I}_1, ..., \mathcal{I}_p$ form a partition of $\mathcal{I}$.
[2] Skew-free gathering has the extra constraint that all the elements except for the special ones need to have distinct keys.
[3] Also known as the *hyper-cube* method.

over $\mathcal{X}$ is a function $\boldsymbol{u} : \mathcal{X} \to [N]$. For any non-empty subset $\mathcal{Y} \subseteq \mathcal{X}$, we define the *projection* of $\boldsymbol{u}$ onto $\mathcal{Y}$, denoted as $\boldsymbol{u}[\mathcal{Y}]$, as the tuple $\boldsymbol{v}$ over $\mathcal{Y}$ that satisfies $\boldsymbol{v}(Y) = \boldsymbol{u}(Y)$ for every attribute $Y \in \mathcal{Y}$. A *relation* R is a set of tuples over the same set $\mathcal{Z}$ of attributes; we refer to $\mathcal{Z}$ as the *schema* of R and represent it as $schema(R)$. A *join* is a set $\mathcal{Q}$ of at least two relations such that $\sum_{R\in\mathcal{Q}} |R| = N$; the integer N will be referred as the join's *input size*. Define $schema(\mathcal{Q}) = \bigcup_{R\in\mathcal{Q}} schema(R)$. The result of $\mathcal{Q}$ is a relation over $schema(\mathcal{Q})$, formalized as:

$$\mathcal{J}oin(\mathcal{Q}) = \{\text{tuple } \boldsymbol{u} \text{ over } schema(\mathcal{Q}) \mid \forall R \in \mathcal{Q} : \boldsymbol{u}[schema(R)] \in R\}.$$

The *schema graph* of $\mathcal{Q}$ is the hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where

$$\mathcal{V} = schema(\mathcal{Q}) \text{ and } \mathcal{E} = \{\!\{schema(R) \mid R \in \mathcal{Q}\}\!\}.$$

Note that $\mathcal{E}$ is a multi-set because some relations in $\mathcal{Q}$ may have the same schema. Focusing on "data complexity", we consider only joins whose schema graphs have constant sizes.

Next, we formalize join computation under communication obliviousness. Fix a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and an integer N that is a multiple of p and satisfies $N \geq p^3$. Let us define the class of $(\mathcal{G}, N)$*-joins* as the set of joins that have schema graph $\mathcal{G}$ and input size N. The input collection $\mathscr{I}(N)$ comprises all the $(\mathcal{G}, N)$-joins. Consider now an arbitrary input $\mathcal{I}$ of $\mathscr{I}(N)$, that is, $\mathcal{I}$ is a $(\mathcal{G}, N)$-join $\mathcal{Q}$. A legal initial state $(\Sigma_1, ..., \Sigma_p)$ of $\mathcal{I}$ meets the following conditions:

- $\Sigma_1, ..., \Sigma_p$ are word sequences with the same length, which is $O(N/p + p^2) = O(N/p)$;
- each Σ_i ($i \in [p]$) describes a set $\mathcal{I}_i$ of N/p tuples in the relations of $\mathcal{Q}$, plus $O(p^2)$ so-called "heavy values" (to be clarified in Section 3);
- $\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_p$ form a partition of all the tuples in the relations of $\mathcal{Q}$.

The goal of an algorithm is to output each tuple of $\mathcal{J}oin(\mathcal{Q})$ on at least one machine.

We show that any $(\mathcal{G}, N)$-join $\mathcal{Q}$ can be settled — communication obliviously — in a single round with load $\tilde{O}(N/p^{1/\psi})$ w.h.p. where $\psi \geq 1$ is the *edge quasi-packing number* [19] of $\mathcal{G}$; see Appendix A for the definition of ψ. The load complexity asymptotically matches that of the state-of-the-art single-round algorithm [19] under the traditional MPC model (that algorithm is not communication oblivious). Both [19] and our algorithm assume that (i) $N \geq p^c$ where c is a constant dependent on $\mathcal{G}$, and (ii) the machines are aware of "heavy" values (whose meanings in our context will be elaborated in Section 3).

Like [19], our algorithm also applies hashing. However, the load $\tilde{O}(N/p^{1/\psi})$ is an intriguing contrast to our near $\Omega(N)$ lower bound on skew-free hashing. Our deployment of hashing circumvents the pitfall of skew-free gathering. Specifically, if a value is "too heavy" in the sense that it is possessed by $\Omega(N/p)$ tuples, our algorithm will always dissipate those tuples across different machines, rather than gathering them on a single machine. This feature is not shared by the algorithm of [19].

Round-Doubling Compilation and Token Passing. Our next contribution is a *round compilation* method that translates a traditional MPC algorithm $\mathcal{A}$ into the communication-oblivious model while preserving the number of rounds of $\mathcal{A}$ by a factor of 2 and its load by a factor of $1 + \delta$ for an arbitrarily small constant $\delta > 0$. Specifically, if $\mathcal{A}$ performs at most $\ell = \text{poly}(p)$ rounds and, w.h.p., incurs a load of at most L, our method yields an oblivious algorithm $\mathcal{A}'$ that computes the same information as $\mathcal{A}$ in 2ℓ rounds with a load $(1 + \delta)L$. Our compilation succeeds w.h.p. as long as $L = \Omega(p \log p)$, a condition satisfied by nearly all the existing MPC algorithms we are aware of.

We also study how much improvement can still be expected over our compilation method. First, it is clear from our earlier discussion on skew-free gathering that the blow-up factor 2 in the round number cannot be improved in general. Specifically, for skew-free gathering, it is easy to achieve load $O(N/p)$ in one round in the traditional MPC model, but under communication obliviousness, every

single-round algorithm must entail a load of nearly $\Omega(N)$ with a constant probability. Given the above, the primary remaining question is whether the condition $L = \Omega(p \log p)$ can be significantly relaxed. Our last contribution is to answer the question in the negative by establishing a hardness result on the following *token passing problem*.

Fix arbitrary integers N and p such that $p \geq 2$ and $N = p \cdot \lceil p^{1-\epsilon} \rceil$, where the constant ϵ satisfies $0 < \epsilon < 1$. Define S to be a set of N elements among which there are $N/p = \lceil p^{1-\epsilon} \rceil$ special elements called *tokens*. For the token passing problem, the input collection $\mathscr{I}(N)$ consists of triples of the form (S, x, y), where x and y are distinct integers in $[p]$; in other words, $\mathscr{I}(N)$ has $p(p-1)$ inputs. Each input $\mathcal{I} = (S, x, y) \in \mathscr{I}(N)$ has only one legal initial state $(\Sigma_1, ..., \Sigma_p)$ defined as follows:

- $\Sigma_1, ..., \Sigma_p$ are word sequences with the same length, which is $\Theta(N/p)$;
- each Σ_i ($i \in [p]$) describes the values of x and y and a subset $S_i \subseteq S$ of size N/p;
- $S_1, S_2, ..., S_p$ form a partition of S;
- S_x contains nothing but tokens (i.e., all the tokens are placed on machine x);
- the content of the other subsets S_i ($i \in [p] \setminus \{x\}$) does not matter, as long as the above conditions are satisfied.

On the above input, the goal of an algorithm is to move all the tokens to machine y.[4]

In the traditional model, the problem can be trivially solved in one round with load $L = O(p^{1-\epsilon})$: simply ask machine x to send all the tokens to machine y. On the other hand, we prove that, if a two-round communication oblivious algorithm is required to succeed with probability at least 2/3, its load must be $\Omega(p^{1-\epsilon/2})$ with a constant probability.

The polynomial gap between $\Omega(p^{1-\epsilon/2})$ and $O(p^{1-\epsilon})$ indicates that the condition $L = \Omega(p \log p)$ of our round-doubling compilation is tight up to a factor sub-polynomial in p. For example, suppose that one could relax the condition to $L = \Omega(p^{0.99})$, while still preserving our round blow-up factor 2 and load blow-up factor $1 + \delta$ w.h.p.. Then, for the token passing problem with $\epsilon = 0.01$, we would be able to compile the aforementioned trivial one-round algorithm into a communication-oblivious counterpart that w.h.p. solves the problem in two rounds with a load of $O(p^{0.99})$ w.h.p.. This, however, contradicts our negative result, which states that the load needs to be $\Omega(p^{0.995})$ with at least a constant probability.

1.3 Previous Results and Relevance to Ours

Closely related to our work is a compilation method by Chan et al. [10]. Our compilation draws inspiration from theirs but makes several improvements. First, the approach of [10] was designed for the "SODA MPC" model [17], which resembles the model in Section 1.1 but does away the notion of "load". In that model, each machine is equipped with $s = \Omega(N/p)$ words of memory, and the amount of communication is not a main concern as long as each machine receives no more than s words in each round from all other machines combined. The analysis of [10] was carried out under the condition of $s = N^{\epsilon}$ for some constant $\epsilon > 0$. When translated into our scenario, the analysis fails to capture the regime where $L \ll s$. Second, our compilation is substantially simpler, restores the clarity for the approach underneath [10] in the scenario where $L = \Omega(p \log p)$, and explicitly determines the blow-up factors $1 + \delta$ and 2 (for preserving the load and number of rounds, respectively)[5]. Finally, our discussion (through the token passing problem) on the condition that L needs to satisfy to enable round-doubling load-preserving compilation is new.

It should be further noted that the study in [10] does not address one-round MPC algorithms. Through our round compilation method, one can see that a primary distinction between the

[4]Unlike skew-free gathering where all the special elements can be moved to an arbitrary machine, here all the tokens must be sent to machine y.

[5]In [10], these factors were hidden in big-O.

conventional MPC model and its oblivious counterpart lies in the realm of one-round algorithms. Exploring fundamental problems that can characterize this difference represents an interesting direction. Our work can be seen as a step in that direction.

As mentioned, the existing single-round algorithms for join processing [3, 4, 7–9, 19] rely crucially on the `Share` method [4, 9]. Given a join $\boldsymbol{Q}$, `Share` assigns a hash function $h_X(\cdot)$ to each attribute $X \in \mathit{schema}(\boldsymbol{Q})$. For each tuple $\boldsymbol{u}$ in a relation $R \in \boldsymbol{Q}$, the machine initially storing $\boldsymbol{u}$ transmits it to a set of machines determined by the hash values in $\{h_X(\boldsymbol{u}(X)) \mid X \in \mathit{schema}(R)\}$. Beame et al. [9] presented a sharp concentration bound on the load of `Share` when certain skew-free requirements are satisfied. Their bound is useful to our analysis and will be reviewed in Section 3. Koutris et al. [19] proved that any one-round MPC algorithm (oblivious or not) solving the join problem must incur a load of $\Omega(N/p^{1/\psi})$ with a constant probability, if each machine is aware of only the tuples in its local storage initially.

There is an extensive body of literature on multi-round MPC algorithms; see [1, 2, 6, 12–16, 18, 22, 25] and the references therein. None of those algorithms was designed to achieve communication obliviousness, but all of them satisfy the condition $L = \Omega(p \log p)$ and, therefore, can be made oblivious by our compilation method.

2 Hardness of Skew-Free Gathering

This section will discuss the skew-free gathering problem defined in Section 1.2 and serves as a proof of the paper's first main result:

Theorem 2.1. *For the skew-free gathering problem parameterized by t, any communication-oblivious one-round algorithm succeeding with probability at least $2/3$ must entail a load of $\Omega(N/p^{1/t})$ with at least a constant probability.*

In Appendix B, we show that the above lower bound $\Omega(N/p^{1/t})$ can be matched deterministically.

We will focus on a subset Φ of legal initial states of the (sole) input $\mathcal{I} \in \mathscr{I}(N)$ built as follows. First, choose a set Z of distinct integers $z_1, z_2, ..., z_t$ in $[p]$; we will refer to Z as the *seed machine set*. Second, divide the $t \cdot N/p$ special elements arbitrarily into t groups of size N/p, and place one group on machine z_j for each $j \in [t]$. Third, divide the other $N \cdot (1 - t/p)$ elements of $\mathcal{I}$ arbitrarily into $p - t$ groups of size N/p, and place one group on machine i for each $i \in [p] \setminus Z$. This defines a legal initial state in Φ. As there are $\binom{p}{t}$ ways to choose Z, the size of Φ is $\binom{p}{t}$.

We require a one-round algorithm $\mathcal{A}$ to be oblivious only on Φ, namely, its communication pattern follows the same probabilistic distribution when it is executed on each initial state $\Sigma \in \Phi$. We will argue that the load of $\mathcal{A}$ must be $\Omega(N/p^{1/t})$ with at least a constant probability if $\mathcal{A}$ succeeds on every $\Sigma \in \Phi$ with probability at least $2/3$. This is sufficient for validating Theorem 2.1.

Conditional Expected Load. As $\mathcal{A}$ executes in one round, its communication pattern can be fully characterized by a single $p \times p$ matrix $\mathbf{L}$ where $\mathbf{L}[i, j]$ $(i, j \in [p])$ is the length of the message $M[i, j]$ that machine i sends to machine j. The load of $\mathcal{A}$ can now be calculated as

$$\mathit{load}(\mathbf{L}) \quad = \quad \max_{i=1}^{p} \Big(\sum_{j=1}^{p} \mathbf{L}[i, j] + \mathbf{L}[j, i] \Big). \tag{1}$$

When $\mathcal{A}$ is randomized, the $\mathbf{M}[i, j]$ of all $i, j \in [p]$ are random variables, and hence so are $\mathbf{L}$ and $\mathit{load}(\mathbf{L})$. Our objective is to prove that $\mathbf{Pr}[\mathit{load}(\mathbf{L}) = \Omega(N/p^{1/t})]$ is at least a constant.

By communication obliviousness, the distribution of $\mathbf{L}$ is the same for all the initial states in Φ. Define $\mathcal{L}$ as the set of $p \times p$ matrices $\mathbf{\Lambda}$ satisfying

- $\mathbf{Pr}[\mathbf{L} = \mathbf{\Lambda}] > 0$, namely, $\mathcal{A}$ exhibits the pattern $\mathbf{L} = \mathbf{\Lambda}$ with a non-zero probability;
- $\max_{i=1}^{p} \sum_{j=1}^{p} \mathbf{\Lambda}[i, j] \leq N/p^{1/t}$ and $\max_{i=1}^{p} \sum_{j=1}^{p} \mathbf{\Lambda}[j, i] \leq N/p^{1/t}$.

When $\mathbf{L}$ equals a matrix $\mathbf{\Lambda}$ as described above, every machine sends at most $N/p^{1/t}$ words and receives at most $N/p^{1/t}$ words. The subsequent discussion will assume

$$\mathbf{Pr}[\mathbf{L} \in \mathcal{L}] \geq 1/2. \tag{2}$$

Note that whenever $\mathbf{L} \notin \mathcal{L}$, some machine communicates at least $N/p^{1/t}$ words and hence $load(\mathbf{L}) \geq N/p^{1/t}$. Thus, $\mathbf{Pr}[\mathbf{L} \notin \mathcal{L}] \geq 1/2$ immediately gives $\mathbf{Pr}[load(\mathbf{L}) \geq N/p^{1/t}] \geq 1/2$, thereby verifying the claim in Theorem 2.1.

We will aim to validate the following inequality:

$$\mathbf{E}[load(\mathbf{L}) \mid \mathbf{L} \in \mathcal{L}] \quad \geq \quad c \cdot N/p^{1/t} \tag{3}$$

for some constant $0 < c < 1$. The inequality implies[6]

$$\mathbf{Pr}[load(\mathbf{L}) \geq (c/2) \cdot N/p^{1/t} \mid \mathbf{L} \in \mathcal{L}] \geq c/4. \tag{4}$$

Combining (2) and (4) yields $\mathbf{Pr}[load(\mathbf{L}) \geq \frac{c}{2} N/p^{1/t}] \geq \frac{c}{4} \cdot \frac{1}{2} = c/8$, thus establishing Theorem 2.1.

Heavy Senders. Suppose that the algorithm $\mathcal{A}$ exhibits a pattern $\mathbf{L} = \mathbf{\Lambda} \in \mathcal{L}$. Given a machine $i \in [p]$, we call machine $j \in [p] \setminus \{i\}$ a *heavy sender* for machine i under $\mathbf{\Lambda}$ if $\mathbf{\Lambda}[j, i] \geq N/p$, namely, machine j sends at least N/p words to machine i. Define

$$H(\mathbf{\Lambda}, i) = \text{number of heavy senders for machine } i \text{ under } \mathbf{\Lambda}. \tag{5}$$

It is easy to see that

$$H(\mathbf{\Lambda}, i) \leq p^{1-1/t} \tag{6}$$

because otherwise machine i would receive more than $\frac{N}{p} p^{1-1/t} = \frac{N}{p^{1/t}}$ words, contradicting $\mathbf{\Lambda} \in \mathcal{L}$.

Our analysis will focus on heavy senders. We will argue that at least one machine $i \in [p]$ satisfies

$$\sum_{\mathbf{\Lambda} \in \mathcal{L}} \mathbf{Pr}[\mathbf{L} = \mathbf{\Lambda}] \cdot H(\mathbf{\Lambda}, i) \quad = \quad \Omega(p^{1-1/t}). \tag{7}$$

As $load(\mathbf{\Lambda}) \geq (N/p) \cdot H(\mathbf{\Lambda}, i)$, the above leads to

$$\sum_{\mathbf{\Lambda} \in \mathcal{L}} \mathbf{Pr}[\mathbf{L} = \mathbf{\Lambda}] \cdot load(\mathbf{\Lambda}) \geq \frac{N}{p} \sum_{\mathbf{\Lambda} \in \mathcal{L}} \mathbf{Pr}[\mathbf{L} = \mathbf{\Lambda}] \cdot H(\mathbf{\Lambda}, i) = \Omega\Big(\frac{N}{p} \cdot p^{1-1/t}\Big) = \Omega(N/p^{1/t}). \tag{8}$$

This will prove (3) because

$$\mathbf{E}[load(\mathbf{L}) \mid \mathbf{L} \in \mathcal{L}] \quad = \quad \frac{1}{\mathbf{Pr}[\mathbf{L} \in \mathcal{L}]} \sum_{\mathbf{\Lambda} \in \mathcal{L}} \mathbf{Pr}[\mathbf{L} = \mathbf{\Lambda}] \cdot load(\mathbf{\Lambda})$$

which is at least (8).

Two Types of Legal Initial States. The rest of the section will focus on proving the correctness of (7). Earlier we have identified a set Φ of $\binom{p}{t}$ legal initial states, each characterized by a seed machine set $Z = \{z_1, ..., z_t\}$; for convenience, we will use Z to denote the corresponding legal initial state when no ambiguity can arise. Now, fix any legal initial state Z. The algorithm $\mathcal{A}$ succeeds on Z if and only if at least one of the following occurs:

- machines $z_1, z_2, ..., z_t$ all send their special elements to a machine $k \in [p] \setminus Z$;

[6] For every $\mathbf{L} \in \mathcal{L}$, it holds that $load(\mathbf{L}) \leq 2\frac{N}{p^{1/t}}$. If $\mathbf{Pr}[load(\mathbf{L}) \geq \frac{c}{2}\frac{N}{p^{1/t}} \mid \mathbf{L} \in \mathcal{L}] < \frac{c}{4}$, then $\mathbf{Pr}[load(\mathbf{L}) < \frac{c}{2}\frac{N}{p^{1/t}} \mid \mathbf{L} \in \mathcal{L}] > 1 - \frac{c}{4}$. Hence, $\mathbf{E}[load(\mathbf{L}) \mid \mathbf{L} \in \mathcal{L}] < \frac{2N}{p^{1/t}}\frac{c}{4} + \frac{c}{2}\frac{N}{p^{1/t}}(1 - \frac{c}{4}) < cN/p^{1/t}$, contradicting (3).

- there is some $j \in [t]$ such that machine z_j receives all the special elements from every other machine in Z.

Echoing the above, given any Z and any matrix $\mathbf{\Lambda} \in \mathcal{L}$, we define

- $X(Z, \mathbf{\Lambda})$ as an indicator variable, which equals 1 if there exists some $k \in [p] \setminus Z$ such that $\mathbf{\Lambda}[z_j, k] \geq N/p$ for every $j \in [t]$, or 0 otherwise;
- $Y(Z, \mathbf{\Lambda})$ as an indicator variable, which equals 1 if there exists some $j \in [t]$ such that $\mathbf{\Lambda}[z_k, z_j] \geq N/p$ for every $k \in [t] \setminus \{j\}$, or 0 otherwise.

When exhibiting a pattern $\mathbf{L} = \mathbf{\Lambda} \in \mathcal{L}$, the algorithm $\mathcal{A}$ succeeds on Z only if

$$X(Z, \mathbf{\Lambda}) + Y(Z, \mathbf{\Lambda}) \geq 1.$$

We declare:

$$\sum_{\mathbf{\Lambda} \in \mathcal{L}} \mathbf{Pr}[\mathbf{L} = \mathbf{\Lambda}] \cdot (X(Z, \mathbf{\Lambda}) + Y(Z, \mathbf{\Lambda})) \quad \geq \quad 1/6. \tag{9}$$

To see why, notice that if the above did not hold, then $\mathcal{A}$ would have probability less than $1/6$ succeeding on Z when $\mathbf{L} \in \mathcal{L}$. Even if $\mathcal{A}$ always succeeds on Z when $\mathbf{L} \notin \mathcal{L}$, the overall success probability on Z would still be less than $1/6 + \mathbf{Pr}[\mathbf{L} \notin \mathcal{L}]$, which is less than $2/3$ because of (2). This violates the requirement that $\mathcal{A}$ must succeed with probability at least $2/3$ on Z.

We will refer to a legal initiate state Z as

- an *X-state* if $\sum_{\mathbf{\Lambda} \in \mathcal{L}} \mathbf{Pr}[\mathbf{L} = \mathbf{\Lambda}] \cdot X(Z, \mathbf{\Lambda}) \geq 1/12$, or
- a *Y-state* if $\sum_{\mathbf{\Lambda} \in \mathcal{L}} \mathbf{Pr}[\mathbf{L} = \mathbf{\Lambda}] \cdot Y(Z, \mathbf{\Lambda}) \geq 1/12$.

Note that Z must be classified as at least one of the two types because of (9). Thus, if $\mathscr{X}$ (resp., $\mathscr{Y}$) represents the set of all X- (resp., Y-) states, we have $|\mathscr{X}| + |\mathscr{Y}| \geq |\Phi| = \binom{p}{t}$. The next lemma relates each type of states to heavy-sender machines.

Lemma 2.2. *For any $\mathbf{\Lambda} \in \mathcal{L}$, it holds that*

$$\sum_{Z \in \mathscr{X}} X(Z, \mathbf{\Lambda}) \quad \leq \quad \sum_{i \in [p]} \binom{H(\mathbf{\Lambda}, i)}{t} \tag{10}$$

$$\sum_{Z \in \mathscr{Y}} Y(Z, \mathbf{\Lambda}) \quad \leq \quad \sum_{i \in [p]} \binom{H(\mathbf{\Lambda}, i)}{t-1}. \tag{11}$$

The reader is reminded that $\binom{x}{y}$ equals 0 if $x < y$. We will prove the first inequality here and the second one in Appendix C.

Proof of (10). Our proof adopts a counting argument. Initially, create an empty set S_i for each $i \in [p]$. Process each $Z = \{z_1, ..., z_t\} \in \mathscr{X}$ as follows. If $X(Z, \mathbf{\Lambda}) = 0$, do nothing. Otherwise, under the communication pattern $\mathbf{L} = \mathbf{\Lambda}$, there is at least one machine $k \in [p] \setminus Z$ that receives N/p special elements from machine z_j for all $j \in [t]$. Thus, machines $z_1, z_2, ..., z_t$ are heavy senders for machine k under $\mathbf{\Lambda}$. We add Z to S_k. From the perspective of machine k, the set Z represents a possible way to choose t machines from its $H(\mathbf{\Lambda}, k)$ heavy senders (under $\mathbf{\Lambda}$). Since all the seed machine sets are different (hence, all the "Z" in $\mathscr{X}$ are different), all the members of S_k are distinct.

After all the sets $Z \in \mathscr{X}$ have been processed, the left hand side of (10) is bounded by $\sum_{i \in [p]} |S_i|$. On the other hand, for each machine $i \in [p]$, the size $|S_i|$ cannot exceed the total number of ways of choosing t machines from the $H(\mathbf{\Lambda}, i)$ heavy senders for machine i (under $\mathbf{\Lambda}$). Inequality (10) now follows. □

The remainder of the argument proceeds differently depending on the number of X-states.

At Least $\frac{1}{2}\binom{p}{t}$ X-States. Namely, $|\mathscr{X}| \geq \frac{1}{2}\binom{p}{t}$, with which we apply the X-state definition to obtain

$$\sum_{Z\in\mathscr{X}}\sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot X(Z,\Lambda) \geq \frac{1}{24}\binom{p}{t}. \tag{12}$$

We can now derive

$$\begin{aligned}\sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot\sum_{i\in[p]} H(\Lambda,i)^t &\geq \sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot\sum_{i\in[p]}\binom{H(\Lambda,i)}{t}\\ \text{(by (10))}\quad &\geq \sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot\sum_{Z\in\mathscr{X}} X(Z,\Lambda)\\ &= \sum_{\Lambda\in\mathcal{L}}\sum_{Z\in\mathscr{X}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot X(Z,\Lambda) \geq \frac{1}{24}\binom{p}{t}\end{aligned} \tag{13}$$

where the last step used (12). This gives:

$$\begin{aligned}\sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\sum_{i\in[p]} H(\Lambda,i) &= \sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot\sum_{i\in[p]}\frac{H(\Lambda,i)^t}{H(\Lambda,i)^{t-1}}\\ \text{(by (6))}\quad &\geq \sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot\sum_{i\in[p]}\frac{H(\Lambda,i)^t}{p^{\frac{(t-1)^2}{t}}}\\ \text{(by (13))}\quad &\geq \frac{1}{24}\binom{p}{t}/p^{\frac{(t-1)^2}{t}} = \Omega\Big(\frac{p^t}{p^{(t-1)^2/t}}\Big) = \Omega\Big(p^{2-1/t}\Big).\end{aligned}$$

Therefore, at least one $i\in[p]$ satisfies $\sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot H(\Lambda,i) = \Omega(p^{2-1/t}/p) = \Omega(p^{1-1/t})$, as claimed in (7).

At most $\frac{1}{2}\binom{p}{t}$ X-States. Because $|\mathscr{X}| + |\mathscr{Y}| \geq \binom{p}{t}$, the fact $|\mathscr{X}| \leq \binom{p}{t}/2$ indicates $|\mathscr{Y}| \geq \binom{p}{t}/2$. Following a derivation similar to the previous case, we can show that at least one $i\in[p]$ satisfies (7). With the details presented in Appendix C, we now conclude the proof of Theorem 2.1.

3 Single-Round Oblivious Joins

We now turn our attention to the join problem defined in Section 1.2. Let $\boldsymbol{Q}$ be the $(\mathcal{G}, N)$-join to be computed. Given a value $x\in[N]$, we call it

- *heavy* if at least N/p^2 tuples of a relation have value x under an attribute, or formally, $\exists R\in\boldsymbol{Q}, X\in schema(R)$ s.t. R has at least N/p^2 tuples $\boldsymbol{u}$ with $\boldsymbol{u}(X)=x$;
- or *light*, otherwise.

There can be $O(p^2)$ heavy values in total. We assume that each machine knows all the heavy values. Rephrased in our communication-obliviousness framework, in all legal initial states $(\Sigma_1,...,\Sigma_p)$, each Σ_i ($i\in[p]$) contains $O(p^2)$ words describing the heavy values (as mentioned in Section 1.2). We will establish our second main result:

> Theorem 3.1. *Consider any constant-size hypergraph $\mathcal{G}=(\mathcal{V},\mathcal{E})$. Let N be a multiple of p satisfying $N\geq p^{1+2\alpha+1/\psi}$ where $\alpha=\max_{e\in\mathcal{E}}|e|$ (called the arity of $\mathcal{G}$) and ψ is the edge quasi-packing number of $\mathcal{G}$ (see Appendix A). There is a communication oblivious algorithm that, given any $(\mathcal{G},N)$-join $\boldsymbol{Q}$, operates in a single round with load $\tilde{O}(N/p^{1/\psi})$ and computes $\mathcal{J}oin(\boldsymbol{Q})$ w.h.p..*

Our result can be compared to an algorithm of [19], which has load $\tilde{O}(N/p^{1/\psi})$ w.h.p. but is not communication oblivious. That algorithm also assumes that each machine knows all the heavy values, although their frequency threshold of a heavy value is N/p, rather than N/p^2 in our context. The rest of the section serves as a proof of Theorem 3.1. We will first review the Share method [4, 9] in Section 3.1. After that, we will present our join algorithm in Sections 3.2 and 3.3.

3.1 The Share Method

Let R^* be a relation with $schema(R^*) = \{X_1, X_2, ..., X_r\}$, and M be an integer satisfying $|R^*| \leq M \leq N$. Recall that, as defined in Section 1.2, each tuple $\boldsymbol{u} \in R^*$ is a function from $schema(R^*)$ to $[N]$. For any $\mathcal{Y} \subseteq schema(R^*)$, define:

$$\deg_{\mathcal{Y}}(R^*) \quad = \quad \max_{\text{tuple } \boldsymbol{u} \text{ over } \mathcal{Y}} \left|\{\boldsymbol{v} \in R^* \mid \boldsymbol{v}[\mathcal{Y}] = \boldsymbol{u}\}\right| \tag{14}$$

that is, the maximum number of tuples in R^* having the same projection on $\mathcal{Y}$. Suppose that we assign to each attribute X_i, $i \in [r]$, a positive integer s_i — called the *share* of X_i — satisfying the following *skew-free condition*:

$$\text{for any non-empty } \mathcal{Y} \subseteq schema(R^*), \deg_{\mathcal{Y}}(R^*) \leq \frac{M}{\prod_{i \in [r]: X_i \in \mathcal{Y}} s_i}. \tag{15}$$

Fix any constant $\alpha \geq r$. For each $i \in [r]$, choose independently a *perfectly random hash function* $h_i : [\alpha N] \rightarrow [s_i]$, that is, h_i is uniformly sampled from the $(s_i)^{\alpha N}$ possible functions from $[\alpha N]$ to $[s_i]$. Given $\boldsymbol{b} = (b_1, b_2, ..., b_r) \in [s_1] \times [s_2] \times ... \times [s_r]$, define

$$bin\, \boldsymbol{b} \quad = \quad \{\boldsymbol{u} \in R^* \mid \forall i \in [r], h_i(\boldsymbol{u}(X_i)) = b_i\}. \tag{16}$$

Note that there are in total $\prod_{i=1}^{r} s_i$ bins. The concentration bound below is due to Beame et al. [9]:

Lemma 3.2. *For any constant $c > 0$, it holds with probability at least $1 - (1/p^*)^c$ that the sizes of all the bins are bounded by $O((\log p^*/\log\log p^*)^r \cdot M/p^*)$, where $p^* = \prod_{i=1}^{r} s_i$. The constant in the big-O depends on c.*

Strictly speaking, Beame et al. [9] proved the lemma only for the case where $M = |R^*|$, which is not enough for our purposes. Fortunately, it is not difficult to extend their result to any $M \in [|R^*|, N]$, as shown in Appendix D.

3.2 A Non-Oblivious Join Algorithm

This subsection will present an algorithm, which is not communication oblivious, to compute the result of a $(\mathcal{G}, N)$-join $\boldsymbol{Q}$ with load $\tilde{O}(N/p^{1/\psi})$ w.h.p.. The reason for describing this non-oblivious version is to allow the reader to draw a direct comparison with the solution of [19]. As will be clear, a key new idea is to apply the concentration bound of Lemma 3.2 locally on each machine, rather than globally on the entire join (as was done in [19]). This in turn requires us to decrease the "heavy threshold" from N/p to N/p^2. Our algorithm ensures a new property (to be given in Lemma 3.3), which makes it easy to make the algorithm communication oblivious, as we do in the next subsection.

Given a relation $R \in \boldsymbol{Q}$ and machine id $i \in [p]$, let $R^{(i)}$ be the set of tuples of R that are initially stored on machine i. Define $R^{(i,heavy)}$ as the set of tuples $\boldsymbol{u} \in R^{(i)}$ using only heavy values (i.e., every attribute value of $\boldsymbol{u}$ is heavy). We know $|R^{(i,heavy)}| = O(p^{2\alpha})$ because $R^{(i)}$ has at most α attributes (recall that α is the arity of the hypergraph $\mathcal{G}$), and there can be $O(p^2)$ heavy values.

Before proceeding, the reader should be familiarized with the content of Appendix A, in particular, the notions of "residual graph" and "fractional vertex cover". Next, let us take an arbitrary proper

subset $\mathcal{Z} \subset \mathcal{V}$ and identify an optimal fractional vertex cover $w_{\mathcal{Z}} : \mathcal{V} \setminus \mathcal{Z} \rightarrow [0, 1]$ of the residual graph $\mathcal{G}_{\mathcal{Z}} = (\mathcal{V}_{\mathcal{Z}}, \mathcal{E}_{\mathcal{Z}})$. Define

$$\tau(\mathcal{G}_{\mathcal{Z}}) \quad = \quad \sum_{X \in \mathcal{V} \setminus \mathcal{Z}} w_{\mathcal{Z}}(X). \tag{17}$$

To each attribute $X \in \mathcal{V}$, we assign a positive integer $s_X(\mathcal{Z})$ — the share of X for $G_{\mathcal{Z}}$ — as follows:

- if $X \in \mathcal{Z}$, then $s_X(\mathcal{Z}) = 1$;
- otherwise, $s_X(\mathcal{Z}) = \left\lfloor p^{w_{\mathcal{Z}}(X)/\tau(\mathcal{G}_{\mathcal{Z}})} \right\rfloor$.

For any non-empty subset $\mathcal{Y} \subseteq \mathcal{V}$, we show in Appendix D:

$$\prod_{X \in \mathcal{Y}} s_X(\mathcal{Z}) \leq p. \tag{18}$$

Furthermore, for any relation $R \in \mathcal{Q}$ such that $schema(R) \setminus \mathcal{Z} \neq \emptyset$, we show again in Appendix D:

$$\prod_{X \in schema(R)} s_X(\mathcal{Z}) \quad = \quad \Omega\Big(p^{1/\tau(G_{\mathcal{Z}})}\Big). \tag{19}$$

Henceforth, set

$$M = N/p. \tag{20}$$

For each $i \in [p]$ and each relation $R \in \mathcal{Q}$ satisfying $schema(R) \setminus \mathcal{Z} \neq \emptyset$, we define $R^{(i,\mathcal{Z})}$ as the set of tuples $\boldsymbol{u} \in R^{(i)}$ such that

- $\boldsymbol{u}(X)$ is heavy for every attribute $X \in schema(R) \cap \mathcal{Z}$;
- $\boldsymbol{u}(X)$ is light for every attribute $X \notin schema(R) \cap \mathcal{Z}$.

For any non-empty subset $\mathcal{Y} \subseteq schema(R)$, let us observe

$$\deg_{\mathcal{Y}}(R^{(i,\mathcal{Z})}) \leq \frac{M}{\prod_{X \in \mathcal{Y}} s_X(\mathcal{Z})}. \tag{21}$$

Indeed, if $\mathcal{Y} \subseteq \mathcal{Z}$, then $\prod_{X \in \mathcal{Y}} s_X(\mathcal{Z}) = 1$ in which case (21) holds because $\deg_{\mathcal{Y}}(R^{(i,\mathcal{Z})}) \leq |R^{(i)}| \leq N/p = M$. Otherwise, identify an arbitrary attribute $X \in \mathcal{Y} \setminus \mathcal{Z}$. Then, we can derive

$$\deg_{\mathcal{Y}}(R^{(i,\mathcal{Z})}) \leq \deg_{\{X\}}(R^{(i,\mathcal{Z})}) \leq \deg_{\{X\}}(R^{(i)}) \leq N/p^2$$

where the last inequality is due to the definition of light value. By (18), $N/p^2 \leq M/\prod_{X \in \mathcal{Y}} s_X(\mathcal{Z})$, from which the correctness of (21) follows. We can therefore conclude that $R^{(i,\mathcal{Z})}$ fulfills the skew-free condition prescribed in (15).

We now elaborate on the join algorithm. Before receiving the $(\mathcal{G}, N)$-join $\mathcal{Q}$, we perform some preprocessing for each proper subset $\mathcal{Z} \subset \mathcal{V}$. First, obtain $s_X(\mathcal{Z})$ — the share of X for $G_{\mathcal{Z}}$ — for every attribute $X \in \mathcal{V}$ as explained earlier. For each $X \in \mathcal{V}$, independently pick a perfectly random hash function $h_{\mathcal{Z},X} : [\alpha N] \rightarrow [s_X(\mathcal{Z})]$ (recall that α is the arity of $\mathcal{G}$). Store a copy of all these functions on every machine. The cartesian product of $\bigtimes_{X \in \mathcal{V}} [s_X(\mathcal{Z})]$ has a size of $\prod_{X \in \mathcal{V}} s_X(\mathcal{Z})$, which is at most p (see (18)). Each element in the cartesian product can be regarded as a tuple $\boldsymbol{b}_{\mathcal{Z}}$, where $\boldsymbol{b}_{\mathcal{Z}}(X)$ is an integer in $[s_X(\mathcal{Z})]$ for each $X \in \mathcal{V}$. We assign a distinct machine to each possible $\boldsymbol{b}_{\mathcal{Z}}$ lexicographically: machine $i \in [p]$ is assigned to the $\boldsymbol{b}_{\mathcal{Z}}$ that ranks, in lexicographic order, the i-th among all the elements of $\bigtimes_{X \in \mathcal{V}} [s_X(\mathcal{Z})]$. Note that some machines (precisely, $p - \prod_{X \in \mathcal{V}} s_X(\mathcal{Z})$ machines) are not assigned to any cartesian product elements.

After the $(\mathcal{G}, N)$-join $\mathcal{Q}$ has been given, each machine — say the one with id $i \in [p]$ — carries out the following steps in parallel:

- **S1:** Broadcast $R^{(i,heavy)}$.

- **S2:** For each proper subset $\mathcal{Z} \subset \mathcal{V}$ and every relation $R \in \mathcal{Q}$ with $schema(R) \setminus \mathcal{Z} \neq \emptyset$, the machine examines every tuple $\boldsymbol{u} \in R^{(i,\mathcal{Z})}$. For each attribute $X \in schema(R)$, the machine computes the hash value $h_{\mathcal{Z},X}(\boldsymbol{u}(X))$. Then, it sends $\boldsymbol{u}$ to every machine $\boldsymbol{b}_{\mathcal{Z}} \in \bigtimes_{X\in\mathcal{V}}[s_X(\mathcal{Z})]$ satisfying the condition that $\boldsymbol{b}_{\mathcal{Z}}(X) = h_{\mathcal{Z},X}(\boldsymbol{u}(X))$ for every $X \in schema(R)$.

The above steps require one round of communication. Then, every machine joins all the tuples received. To prove correctness, consider any tuple $\boldsymbol{u} \in \mathcal{J}oin(\mathcal{Q})$. If $\boldsymbol{u}$ uses a heavy value on every attribute, it is produced by all machines due to step **S1**. Otherwise, suppose that $\boldsymbol{u}$ uses heavy values only on those attributes in some $\mathcal{Z} \subset \mathcal{V}$. Then, due to step **S2**, it must be produced at the machine corresponding to the element $\boldsymbol{b}_{\mathcal{Z}} \in \bigtimes_{X\in\mathcal{V}}[s_X(\mathcal{Z})]$ where $\boldsymbol{b}_{\mathcal{Z}}(X) = h_{\mathcal{Z},X}(\boldsymbol{u}(X))$ for all $X \in \mathcal{V}$.

We will prove that the load incurred is $O(p^{2\alpha+1}) + \tilde{O}(N/p^{1/\psi})$ w.h.p., which is $\tilde{O}(N/p^{1/\psi})$ for $N \geq p^{1+2\alpha+1/\psi}$. Clearly, step **S1** entails load $O(p^{2\alpha+1})$. Next, we will show that, for any particular $\mathcal{Z} \subset \mathcal{V}$, **S2** generates a load of $\tilde{O}(N/p^{1/\tau(\mathcal{G}_{\mathcal{Z}})})$ w.h.p., where $\tau(\mathcal{G}_{\mathcal{Z}})$ is given in (17). It will then follow that the overall load of **S2** is $\tilde{O}(N/p^{1/\psi})$ because ψ equals the maximum $\tau(\mathcal{G}_{\mathcal{Z}})$ of all $\mathcal{Z} \subset \mathcal{V}$ (see (26) in Appendix A) and $\mathcal{V}$ has a constant number of subsets.

The lemma below characterizes the behavior of step **S2**.

Lemma 3.3. *Consider an arbitrary proper subset $\mathcal{Z} \subset \mathcal{V}$ and an arbitrary relation $R \in \mathcal{Q}$ satisfying $schema(R) \setminus \mathcal{Z} \neq \emptyset$. The following statement holds w.h.p.: for all distinct machine ids $i, j \in [p]$, machine i sends $\tilde{O}(M/p^{1/\tau(\mathcal{G}_{\mathcal{Z}})})$ tuples in $R^{(i,\mathcal{Z})}$ to machine j in step* **S2**.

Proof. As explained previously in (21), relation $R^{(i,\mathcal{Z})}$ satisfies the skew-free condition (21) with respect to the shares $\{s_X(\mathcal{Z}) \mid X \in \mathcal{V}\}$. Set $p^* = \prod_{X\in schema(R)} s_X(\mathcal{Z})$. As $|R^{(i,\mathcal{Z})}| \leq M \leq N$, Lemma 3.2 asserts that the following event occurs w.h.p.:

> $\tilde{O}(M/p^*)$ tuples $\boldsymbol{u} \in R^{(i,\mathcal{Z})}$ are "hashed" to each element $\boldsymbol{b} \in \bigtimes_{X\in schema(R)}[s_X(\mathcal{Z})]$; specifically, $\boldsymbol{u}$ is hashed to $\boldsymbol{b}$ if $\boldsymbol{b}(X) = h_{\mathcal{Z},X}(\boldsymbol{u}(X))$ for all $X \in schema(R)$.

Under the above event, $\tilde{O}(M/p^*)$ tuples $\boldsymbol{u} \in R^{(i,\mathcal{Z})}$ are hashed to each element $\boldsymbol{b}_{\mathcal{Z}} \in \bigtimes_{X\in\mathcal{V}}[s_X(\mathcal{Z})]$, i.e., such a tuple $\boldsymbol{u}$ satisfies $\boldsymbol{b}_{\mathcal{Z}}(X) = h_{\mathcal{Z},X}(\boldsymbol{u}(X))$ for all $X \in schema(R)$. Hence, if the machine j stated in the lemma is assigned to an element in $\bigtimes_{X\in\mathcal{V}}[s_X(\mathcal{Z})]$, then machine i sends $\tilde{O}(M/p^*)$ tuples of $R^{(i,\mathcal{Z})}$ to machine j; otherwise, machine i sends no tuple of $R^{(i,\mathcal{Z})}$ to machine j.

The lemma now follows from $p^* = \Omega(p^{1/\tau(\mathcal{G}_{\mathcal{Z}})})$ (see (19)). □

As $\mathcal{Q}$ has $O(1)$ relations, Lemma 3.3 indicates that every machine sends and receives $\tilde{O}(p \cdot M/p^{1/\tau(\mathcal{G}_{\mathcal{Z}})}) = \tilde{O}(N/p^{1/\tau(\mathcal{G}_{\mathcal{Z}})})$ words in step **S2** w.h.p. when processing the subset $\mathcal{Z}$.

Remark. Lemma 3.3 is a feature of our algorithm that is not shared by the algorithm of [19], but is crucial for turning the algorithm into an oblivious counterpart, as discussed next.

3.3 Making the Algorithm Oblivious

Our algorithm in Section 3.2 enjoys a simple communication pattern. It can be modified into a communication-oblivious version by padding enough dummy words to make the length of each message consistent with the "worst" case.

Let us start with step **S1**, where machine $i \in [p]$ transmits the same message, which contains $R^{(i,heavy)}$ for each $R \in \mathcal{Q}$, to every other machine. This message can have a maximum length of $len_{heavy} = O(p^{2\alpha})$ words. Whenever this message is shorter than len_{heavy}, expand it with dummy words into length len_{heavy}.

In step **S2**, for each proper subset $\mathcal{Z} \subset \mathcal{V}$, machine $i \in [p]$ transmits a possibly different message to each other machine $j \in [p]$, whose length is bounded by an integer $len_{\mathcal{Z}} = \tilde{O}(M/p^{1/\tau(\mathcal{G}_{\mathcal{Z}})})$

w.h.p. (Lemma 3.3). Whenever this message is shorter than $len_{\mathcal{Z}}$ — including an empty message — expand it with dummy words into length $len_{\mathcal{Z}}$. However, it is possible for the message to be actually longer than $len_{\mathcal{Z}}$ (this happens when the high-probability event in Lemma 3.3 does not occur). In that scenario, machine i (arbitrarily) trims the message at the length of $len_{\mathcal{Z}}$ and sends the trimmed message to machine j anyway; when this happens, we say that the algorithm *errs*.

In the above modified algorithm, every machine sends precisely $len_{heavy} + \sum_{\mathcal{Z}\subset\mathcal{V}} len_{\mathcal{Z}}$ words to every other machine. It therefore exhibits a deterministic communication pattern for all $(\mathcal{G}, N)$-joins — this is true regardless of whether the algorithm errs. The load is always $p \cdot (len_{heavy} + \sum_{\mathcal{Z}\subset\mathcal{V}} len_{\mathcal{Z}})$, which is $\tilde{O}(N/p^{1/\psi})$ as analyzed in the previous subsection. The algorithm computes the join result correctly if it does not err (an adversary cannot tell whether the algorithm has erred because everything is encrypted). Due to Lemma 3.3, by choosing constants appropriately, we can reduce the probability that the algorithm errs to at most $1/p^c$ for an arbitrarily large constant c. This completes the proof of Theorem 3.1.

4 Two-Round Compilation

This section will focus on multi-round MPC algorithms in the communication-oblivious model. Our first main contribution is a compilation method that establishes the following theorem:

Theorem 4.1. *Fix an arbitrary constant δ satisfying $0 < \delta < 1$. Let $\mathcal{A}$ be an algorithm under the traditional MPC model that operates within $\ell = \text{poly}(p)$ rounds and entails a load at most L w.h.p.. If $L = \Omega(p \log p)$ where the hidden constant depends on δ, there is a communication-oblivious algorithm that performs 2ℓ rounds, requires a load at most $(1+\delta)L$, and computes the same information as $\mathcal{A}$ on every machine w.h.p..*

As explained in Section 1.2, the round blow-up factor 2 in the above theorem is the best possible, regardless of the constant δ. Our second main contribution in this section is to show that the condition $L = \Omega(p \log p)$ can no longer be relaxed significantly. This is achieved with the following theorem that establishes the hardness of the token passing problem. The reason why this hardness implies the near-tightness of the condition $L = \Omega(p \log p)$ has been explained in Section 1.2.

Theorem 4.2. *Fix any constant ϵ satisfying $0 < \epsilon < 1$. For the token-passing problem (defined in Section 1.2) parameterized by ϵ, any communication-oblivious two-round algorithm succeeding with probability at least $2/3$ must demand a load of $\Omega(p^{1-\epsilon/2})$ with at least a constant probability.*

We will present the algorithmic procedure of our compilation method in the rest of the section, but defer its analysis to Appendix E. The proof of Theorem 4.2 is provided in Appendix F.

Message Routing. To prove Theorem 4.1, we will tackle a *message routing problem* defined as follows. Suppose that, for each pair of $i, j \in [p]$, machine i needs to send a message $\mathbf{M}[i, j]$ to machine j (if $i = j$, then $\mathbf{M}[i, j]$ is an empty message). Denote by $\mathbf{L}[i, j]$ the length of $\mathbf{M}[i, j]$ in the number of words. For each $i \in [p]$, we have

$$\sum_{j\in[p]} \mathbf{L}[i, j] + \mathbf{L}[j, i] \leq L \tag{22}$$

namely, every machine sends and receives no more than L words in total. We want to design a communication-oblivious algorithm $\mathcal{A}_{route}$ with all the requirements below:

- It runs in two rounds.

- For distinct machines $i, j \in [p]$, machine i sends (precisely) L' words to machine j in each round, where $L' \leq (1+\delta)L/p$.
- W.h.p., all the messages in $\mathbf{M}$ are successfully delivered at the end of the second round.

Equipped with $\mathcal{A}_{route}$, we can prove Theorem 4.1 as follows. Recall (from Section 1.1) that each round of $\mathcal{A}$ runs in two phases where in the first phase each machine prepares the messages to be sent out in the second phase. We treat the second phase as an instance of the message routing problem and apply $\mathcal{A}_{route}$ to deliver the messages. W.h.p., at the end of $\mathcal{A}_{route}$, every machine acquires all the information that it would have obtained at the second phase of $\mathcal{A}$, and can thus perform the local computation as demanded by $\mathcal{A}$. Because $\mathcal{A}$ always terminates within $\ell = \text{poly}(p)$ rounds, our compilation succeeds on all its rounds w.h.p..

Our Algorithm. Set $\delta' = \lceil 16/\delta \rceil$. We chop each message $\mathbf{M}[i, j]$ into segments, each of which has δ' words — if the last segment is shorter than δ' words, expand it with dummy words to make its length δ'. The number of segments is $\lceil \mathbf{L}[i, j]/\delta' \rceil$. Each segment, referred to as a *packet* henceforth, is augmented with three fields: (i) *sender*, i.e., the value i (sender machine id), (ii) *recipient*, i.e., the value j (recipient machine id), and (iii) *sequence number*, i.e., the t-th packet of $\mathbf{M}[i, j]$ has sequence number t. Packets may arrive at machine j in an arbitrary order; the sender and sequence-number fields allow the packets to be put back into the original sequence in $\mathbf{M}[i, j]$. Every packet is thus $\delta' + 3$ words in length.

Denote by P_i the set of packets obtained from $\mathbf{M}[i, 1]$, $\mathbf{M}[i, 2]$, ..., $\mathbf{M}[i, p]$ combined. Let $|P_i|$ be the number of those packets. Then

$$|P_i| \leq \sum_{j=1}^{p} \left(1 + \frac{\mathbf{L}[i, j]}{\delta'}\right) \leq p + \frac{L}{\delta'}. \tag{23}$$

where the last inequality used (22). Set

$$\lambda = \left\lceil \left(1 + \frac{L}{\delta' p}\right)\left(1 + \frac{\delta}{2}\right) \right\rceil. \tag{24}$$

In the first round, each machine — say machine $i \in [p]$ — in parallel carries out the steps below:

- To each packet in P_i, assign a *relay machine* chosen from $[p]$ uniformly at random. Let $P_i(j)$ be the set of packets in P_i assigned to the same relay machine $j \in [p]$. We use $|P_i(j)|$ to represent the number of packets in $P_i(j)$.
- For each $j \in [p]$, if $|P_i(j)| < \lambda$, add enough dummy packets to $P_i(j)$ to make the number of packets therein exactly λ. If $|P_i(j)| > \lambda$, discard (arbitrarily) some packets from $P_i(j)$ to shrink $|P_i(j)|$ to λ; however, in this case, we say that the algorithm *errs*.
- For each $j \in [p]$, send $P_i(j)$ to machine j (of course, for $j = i$, "sending" $P_i(j)$ requires no communication).

For each $i \in [p]$, define

$$P_i^* = \cup_{j \in [p]} P_j(i)$$

namely, the set of all packets to be "relayed" by machine i. These packets are transmitted to machine i in the first round. Machine i sorts the packets by recipient field. For each $j \in [p]$, let $P_i^*(j)$ be the set of packets in P_i^* whose recipient fields are j. In the second round, machine i, in parallel to others, proceeds as follows:

- For each $j \in [p]$, if $|P_i^*(j)| < \lambda$, add dummy packets to $P_i^*(j)$ to increase $|P_i^*(j)|$ to λ. If $|P_i^*(j)| > \lambda$, discard (arbitrarily) some packets from $P_i^*(j)$ to shrink $|P_i^*(j)|$ to λ; in this case, we say that the algorithm *errs*.

- For each $j \in [p]$, send $P_i^*(j)$ to machine j.

This completes our 2-round algorithm for message routing.

In each round, for any distinct $i, j \in [p]$, machine i sends to machine j precisely

$$L' = \lambda \cdot (\delta' + 3) \tag{25}$$

words. It is rudimentary to verify that $L' \leq (1+\delta)L/p$ when p is greater than a constant. Thus, our 2-round algorithm has a deterministic communication pattern requiring a load of $L'(p-1) < (1+\delta)L$. If the algorithm does not err, every message $\mathbf{M}[i, j]$ is successfully delivered. In Appendix E, we prove that the algorithm errs with probability at most $1/p^c$ where c can be an arbitrarily large constant chosen before running the algorithm. This completes the proof of Theorem 4.1.

References

[1] Foto N. Afrati, Manas R. Joglekar, Christopher Ré, Semih Salihoglu, and Jeffrey D. Ullman. 2017. GYM: A Multiround Distributed Join Algorithm. In *Proceedings of International Conference on Database Theory (ICDT)*. 4:1–4:18.

[2] Foto N. Afrati, Paraschos Koutris, Dan Suciu, and Jeffrey D. Ullman. 2015. Parallel Skyline Queries. *Theory Comput. Syst.* 57, 4 (2015), 1008–1037.

[3] Foto N. Afrati, Anish Das Sarma, Semih Salihoglu, and Jeffrey D. Ullman. 2013. Upper and Lower Bounds on the Cost of a Map-Reduce Computation. *Proceedings of the VLDB Endowment (PVLDB)* 6, 4 (2013), 277–288.

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

[5] Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. 2023. OptORAMa: Optimal Oblivious RAM. *Journal of the ACM (JACM)* 70, 1 (2023), 4:1–4:70.

[6] Sepehr Assadi, Nikolai Karpov, and Qin Zhang. 2019. Distributed and Streaming Linear Programming in Low Dimensions. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 236–253.

[7] Paul Beame, Paraschos Koutris, and Dan Suciu. 2013. Communication steps for parallel query processing. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 273–284.

[8] Paul Beame, Paraschos Koutris, and Dan Suciu. 2014. Skew in parallel query processing. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 212–223.

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

[10] T.-H. Hubert Chan, Kai-Min Chung, Wei-Kai Lin, and Elaine Shi. 2020. MPC for MPC: Secure Computation on a Massively Parallel Computing Architecture. In *Innovations in Theoretical Computer Science (ITCS)*, Vol. 151. 75:1–75:52.

[11] Oded Goldreich and Rafail Ostrovsky. 1996. Software Protection and Simulation on Oblivious RAMs. *Journal of the ACM (JACM)* 43, 3 (1996), 431–473.

[12] Guanhao Hou, Xingguang Chen, Sibo Wang, and Zhewei Wei. 2021. Massively parallel algorithms for personalized pagerank. *Proceedings of the VLDB Endowment (PVLDB)* 14, 9 (2021), 1668–1680.

[13] Xiao Hu. 2021. Cover or Pack: New Upper and Lower Bounds for Massively Parallel Joins. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 181–198.

[14] Xiao Hu and Ke Yi. 2019. Instance and Output Optimal Parallel Algorithms for Acyclic Joins. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 450–463.

[15] Xiao Hu and Ke Yi. 2020. Parallel Algorithms for Sparse Matrix Multiplication and Join-Aggregate Queries. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 411–425.

[16] Xiao Hu, Ke Yi, and Yufei Tao. 2019. Output-Optimal Massively Parallel Algorithms for Similarity Joins. *ACM Transactions on Database Systems (TODS)* 44, 2 (2019), 6:1–6:36.

[17] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. 2010. A Model of Computation for MapReduce. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 938–948.

[18] Bas Ketsman, Dan Suciu, and Yufei Tao. 2022. A Near-Optimal Parallel Algorithm for Joining Binary Relations. *Log. Methods Comput. Sci.* 18, 2 (2022).

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

[20] Kasper Green Larsen and Jesper Buus Nielsen. 2018. Yes, There is an Oblivious RAM Lower Bound!. In *Proceedings of Annual International Cryptology Conference (CROPTO)*, Vol. 10992. 523–542.

[21] Sarvar Patel, Giuseppe Persiano, Mariana Raykova, and Kevin Yeo. 2018. PanORAMa: Oblivious RAM with Logarithmic Overhead. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Mikkel Thorup (Ed.). 871–882.

[22] Miao Qiao and Yufei Tao. 2021. Two-Attribute Skew Free, Isolated CP Theorem, and Massively Parallel Joins. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 166–180.
[23] Edward R. Scheinerman and Daniel H. Ullman. 1997. *Fractional Graph Theory: A Rational Approach to the Theory of Graphs*. Wiley, New York.
[24] Cheng Sheng, Yufei Tao, and Jianzhong Li. 2012. Exact and approximate algorithms for the most connected vertex problem. *ACM Transactions on Database Systems (TODS)* 37, 2 (2012), 12:1–12:39.
[25] Yufei Tao. 2018. Massively Parallel Entity Matching with Linear Classification in Low Dimensional Space. In *Proceedings of International Conference on Database Theory (ICDT)*, Vol. 98. 20:1–20:19.

Appendix

A Basic Concepts of Hypergraphs

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a hypergraph with $\mathcal{E}$ being a multi-set where each element, called an *edge*, is a non-empty subset of $\mathcal{V}$ (the set of *vertices*). A *fractional edge packing* is a function $w : \mathcal{E} \to [0, 1]$ (mapping each edge $e \in \mathcal{E}$ to a weight $w(e)$ between 0 and 1) such that, for any attribute $X \in \mathcal{V}$, we have $\sum_{e \in \mathcal{E}: X \in e} w(e) \leq 1$ (i.e., the weight sum of all the edges containing X is at most 1). The *total weight* of w is $\sum_{e \in \mathcal{E}} w(e)$, i.e., the weight sum of all the edges in $\mathcal{E}$. The *fractional edge packing number* of $\mathcal{G}$, denoted as $\tau(\mathcal{G})$, is defined as the maximum total weight of all the fractional edge packings of $\mathcal{G}$.

Dual to fractional edge packing is the notion of *fractional vertex cover*, which is a function $w : \mathcal{V} \to [0, 1]$ (mapping each vertex $X \in \mathcal{V}$ to a weight $w(X)$ between 0 and 1) such that, for any edge $e \in \mathcal{E}$, we have $\sum_{X \in \mathcal{V}: X \in e} w(X) \geq 1$ (i.e., the weight sum of all the vertices in e is at least 1). The *total weight* of w is $\sum_{X \in \mathcal{V}} w(X)$, i.e., the weight sum of all the vertices in $\mathcal{V}$. The *fractional vertex cover number* of $\mathcal{G}$ is defined as the minimum total weight of all the fractional vertex covers of $\mathcal{G}$. It can be shown [23] that the fractional vertex cover number of $\mathcal{G}$ is always identical to the fractional edge packing number $\tau(\mathcal{G})$.

Given a proper subset $\mathcal{Z} \subset \mathcal{V}$, we define $\mathcal{G}_{\mathcal{Z}}$ as the hypergraph obtained by "removing" the attributes of $\mathcal{Z}$ from $\mathcal{G}$. Specifically:

- The vertex set of $\mathcal{G}_{\mathcal{Z}}$ is $\mathcal{V} \setminus \mathcal{Z}$.
- The edges of $\mathcal{G}_{\mathcal{Z}}$ are decided as follows: for each edge $e \in \mathcal{G}$, we add the edge $e \setminus \mathcal{Z}$ to $\mathcal{G}_{\mathcal{Z}}$ if $e \setminus \mathcal{Z} \neq \emptyset$.

We will refer to $\mathcal{G}_{\mathcal{Z}}$ as a *residual graph* of $\mathcal{G}$. Specially, if $\mathcal{Z} = \emptyset$, then $\mathcal{G}_{\mathcal{Z}} = \mathcal{G}$.

For each proper subset $\mathcal{Z} \subset \mathcal{V}$, the residual graph $\mathcal{G}_{\mathcal{Z}}$ has its own fractional edge packing number $\tau(\mathcal{G}_{\mathcal{Z}})$. The maximum such number of all possible $\mathcal{Z}$ is the *edge quasi-packing number* $\psi(\mathcal{G})$ of $\mathcal{G}$, or formally:

$$\psi(\mathcal{G}) = \max_{\mathcal{Z} \subset \mathcal{V}} \tau(\mathcal{G}_{\mathcal{Z}}). \tag{26}$$

Due to the equivalence between fractional edge packing number and fractional vertex cover number, the value $\psi(\mathcal{G})$ can alternatively be defined as the maximum fractional vertex cover number of all residual graphs $G_{\mathcal{Z}}$ of $\mathcal{G}$.

B A Skew-Free Gathering Algorithm

In this section, we present a deterministic communication-oblivious algorithm to solve the skew-free gathering problem (parameterized by integer t) with load $O(N/p^{1/t})$, matching the lower bound in Theorem 2.1.

Our algorithm is in essence an adaptation of the "hyper-cube algorithm" in [4]. Define $h = \lfloor p^{1/t} \rfloor$, and order the points in the t-dimensional space $[h]^t$ lexicographically. For each $i \in [h^t]$, assign

machine i to the i-th point in $[h]^t$ under the lexicographic order. Note that some machines (precisely, $p - h^t$ machines) are not assigned to any point.

Divide domain $[p]$ into h disjoint intervals $I_1, I_2, ..., I_h$, ordered in ascending order of their starting points, such that:

- each of $I_1, ..., I_{h-1}$ covers $\lfloor p/h \rfloor$ integers;
- I_h covers $p - (h-1)\lfloor p/h \rfloor$ integers.

For each interval I_c with $c \in [h]$, we use $|I_c|$ to denote the number of integers covered by I_c. Consider any point $(c_1, c_2, ..., c_t) \in [h]^t$. Remember that the point corresponds to a machine. We associate the machine with a t-dimensional box $I_{c_1} \times I_{c_2} \times ... \times I_{c_t}$ and will refer to I_{c_j} the machine's *projection* on dimension $j \in [t]$.

Given a legal initial state of the skew-free gathering problem, each machine — say the one with id $i \in [p]$ — sends the N/p elements in its local storage to every machine whose projection on *at least* one dimension covers the value i — there are at most $t \cdot h^{t-1}$ such machines. This completes the description of our algorithm.

To see the algorithm's correctness, suppose that machines where the special elements are initially placed have ids $z_1, z_2, ..., z_t \in [p]$. For each $j \in [t]$, let c_j be the integer in $[h]$ such that the interval I_{c_j} covers z_j. Our algorithm ensures that the machine associated with the t-dimensional box $I_{c_1} \times I_{c_2} \times ... \times I_{c_t}$ receives all the special elements from the machines $z_1, ..., z_t$. The algorithm's communication obliviousness can be verified from the fact that, for each $i \in [p]$, machine i sends exactly N/p elements to a set of machines that is determined in a way independent of how the input elements are initially distributed.

It remains to analyze the algorithm's load. In terms of sending, each machine delivers N/p elements to at most $t \cdot h^{t-1} = O(h^{t-1}) = O(p^{(t-1)/t})$ machines and, thus, transmits $O((N/p) \cdot p^{(t-1)/t}) = O(N/p^{1/t})$ words. In terms of receiving, the machine corresponding to the t-dimensional box $I_{c_1} \times I_{c_2} \times ... \times I_{c_t}$ receives at most $(N/p) \cdot \sum_{j=1}^{t} |I_{c_j}|$ elements. Among all the intervals $I_1, ..., I_h$, the interval I_h is the longest and covers

$$p - (h-1)\left\lfloor \frac{p}{h} \right\rfloor < p - (h-1)\left(\frac{p}{h} - 1\right) < \frac{p}{h} + h = O(p^{1-1/t})$$

integers (recall that $t \geq 2$). Each machine, therefore, receives $O((N/p) \cdot p^{1-1/t}) = O(N/p^{1/t})$ words. This proves that our algorithm has a load of $O(N/p^{1/t})$.

C Completing the Proof of Thm. 2.1

Proof of Inequality (11). We adopt a counting argument similar to the one used to prove (10). Initially, create an empty set S_i for each $i \in [p]$. Then, process each $Z = (z_1, ..., z_t) \in \mathscr{Y}$ as follows. If $Y(Z, \mathbf{\Lambda}) = 0$, do nothing. Otherwise, under the communication pattern $\mathbf{L} = \mathbf{\Lambda}$, at least one machine z_k, for some $k \in [t]$, receives N/p special elements from each machine z_j with $j \in [t] \setminus \{k\}$. This means that all the machines in $Z \setminus \{z_k\}$ are heavy senders for machine z_k under $\mathbf{\Lambda}$. We add Z to S_k. From the perspective of machine z_k, the set $Z \setminus \{z_k\}$ represents a possible way to choose $t-1$ machines from the $H(\mathbf{\Lambda}, z_k)$ heavy senders for machine z_k (under $\mathbf{\Lambda}$).

After all the sets $Z \in \mathscr{Y}$ have been processed, the left-hand side of (11) is bounded by $\sum_{i \in [p]} |S_i|$. On the other hand, for each $i \in [p]$, the size $|S_i|$ cannot exceed the total number of ways of choosing $t-1$ machines from the $H(\mathbf{\Lambda}, i)$ heavy senders for machine i (under $\mathbf{\Lambda}$). Inequality (11) now follows.

Proof of (7) **When** $|\mathscr{Y}| \geq \frac{1}{2}\binom{p}{t}$**.** By the definition of the Y-state, the fact $|\mathscr{Y}| \geq \frac{1}{2}\binom{p}{t}$ yields

$$\sum_{Z \in \mathscr{Y}} \sum_{\mathbf{\Lambda} \in \mathcal{L}} \mathbf{Pr}[\mathbf{L} = \mathbf{\Lambda}] \cdot Y(Z, \mathbf{\Lambda}) \geq \frac{1}{24}\binom{p}{t}. \tag{27}$$

Then, we can obtain

$$\begin{aligned}
\sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot\sum_{i\in[p]} H(\Lambda,i)^{t-1} &\geq \sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot\sum_{i\in[p]}\binom{H(\Lambda,i)}{t-1}\\
\text{(by (11))} \quad &\geq \sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot\sum_{Z\in\mathscr{Y}} Y(Z,\Lambda)\\
&= \sum_{\Lambda\in\mathcal{L}}\sum_{Z\in\mathscr{Y}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot Y(Z,\Lambda)\\
\text{(by (27))} \quad &\geq \frac{1}{24}\binom{p}{t}. \qquad (28)
\end{aligned}$$

This leads to:

$$\begin{aligned}
\sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\sum_{i\in[p]} H(\Lambda,i) &= \sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot\sum_{i\in[p]}\frac{H(\Lambda,i)^{t-1}}{H(\Lambda,i)^{t-2}}\\
\text{(by (6))} \quad &\geq \sum_{\Lambda\in\mathcal{L}} \mathbf{Pr}[\mathbf{L}=\Lambda]\cdot\sum_{i\in[p]}\frac{H(\Lambda,i)^{t-1}}{p^{\frac{(t-1)\cdot(t-2)}{t}}}\\
\text{(by (28))} \quad &\geq \frac{1}{24}\binom{p}{t}/p^{\frac{(t-1)\cdot(t-2)}{t}}\\
&= \Omega\Big(\frac{p^t}{p^{\frac{(t-1)\cdot(t-2)}{t}}}\Big)=\Omega\Big(p^{3-2/t}\Big).
\end{aligned}$$

Therefore, at least one $i\in[p]$ satisfies $\sum_{\Lambda\in\mathcal{L}}\mathbf{Pr}[\mathbf{L}=\Lambda]\cdot H(\Lambda,i)=\Omega(p^{3-2/t}/p)=\Omega(p^{1-1/t})$, as claimed in (7).

D Completing the Proof of Thm. 3.1

Proof of Lemma 3.2. As mentioned, Beame et al. [9] has proved Lemma 3.2 for the case where $M=|R^*|$. We will utilize their result as a black box to prove the lemma under $|R^*|<M\leq N$.

The number of distinct values taken by the tuples in R^* is at most $|R^*|\cdot|schema(R^*)|\leq\alpha|R^*|$. Hence, we can find in the domain $[\alpha N]$ a set S of $\alpha N-\alpha|R^*|$ values that are not taken by any tuple of R^*. From $|R^*|<M\leq N$, we know

$$|S|=\alpha N-\alpha|R^*|>\alpha N-(\alpha-1)N-|R^*|=N-|R^*|\geq M-|R^*|.$$

Identify (arbitrarily) $M-|R^*|$ distinct values from S and, for each such value x, create a tuple $\boldsymbol{u}$ over $schema(R^*)$ with $\boldsymbol{u}(X)=x$ for each $X\in schema(R^*)$. Let R' be the relation obtained by inserting these $M-|R^*|$ tuples into R^*. Note that $|R'|$ has precisely M tuples.

It is easy to verify that $\deg_{\mathcal{Y}}(R^*)=\deg_{\mathcal{Y}}(R')$ for every non-empty subset $\mathcal{Y}\subseteq schema(R')$. With this, the skew-free condition (15) tells us that, for any non-empty subset $\mathcal{Y}\subseteq schema(R')$, we must have:

$$\deg_{\mathcal{Y}}(R')\leq\frac{|R'|}{\prod_{i\in[r]:X_i\in\mathcal{Y}} s_i}.$$

Recall that, for each $i\in[r]$, we have chosen a perfectly random hash function $h_i:[\alpha N]\to[s_i]$. Given $\boldsymbol{b}=(b_1,b_2,...,b_r)\in[s_1]\times[s_2]\times...\times[s_r]$, define

$$bin\,\boldsymbol{b'} \quad = \quad \{\boldsymbol{u}\in R' \mid \forall i\in[r], h_i(\boldsymbol{u}(X_i))=b_i\}. \qquad (29)$$

Clearly, bin $\boldsymbol{b'}$ contains all the tuples in the bin $\boldsymbol{b}$ defined in (16). By the result of [9], we know that every bin $\boldsymbol{b'}$ in (29) contains $O((\log p^*/\log\log p^*)^r\cdot|R'|/p^*)=O((\log p^*/\log\log p^*)^r\cdot M/p^*)$ tuples,

where $p^* = \prod_{i=1}^{r} s_i$. It thus follows that every bin $\boldsymbol{b}$ in (16) must have a size $O((\log p^*/\log\log p^*)^r \cdot M/p^*)$.

Proof of Equation (18)**.** If $\mathcal{Y} \subseteq \mathcal{Z}$, then $\prod_{X\in\mathcal{Y}} s_X(\mathcal{Z}) = 1 \leq p$. Otherwise:

$$\prod_{X\in\mathcal{Y}} s_X(\mathcal{Z}) = \prod_{X\in\mathcal{Y}\setminus\mathcal{Z}} s_X(\mathcal{Z}) \leq \prod_{X\in\mathcal{Y}\setminus\mathcal{Z}} p^{w_{\mathcal{Z}}(X)/\tau(\mathcal{G}_{\mathcal{Z}})} \leq p$$

where the last step used the definition of $\tau(\mathcal{G}_{\mathcal{Z}})$ in (17).

Proof of Equation (19)**.** Let $e_R(\mathcal{Z}) = schema(R) \setminus \mathcal{Z}$. Note that $e_R(\mathcal{Z})$ is an edge in the residual graph $\mathcal{G}_{\mathcal{Z}}$. We have

$$\begin{aligned}
\prod_{X\in schema(R)} s_X(\mathcal{Z}) &= \prod_{X\in e_R(\mathcal{Z})} s_X(\mathcal{Z}) \cdot \prod_{X\in\mathcal{Z}} s_X(\mathcal{Z}) \\
&= \prod_{X\in e_R(\mathcal{Z})} s_X(\mathcal{Z}) \\
&= \prod_{X\in e_R(\mathcal{Z})} \left\lfloor p^{w_{\mathcal{Z}}(X)/\tau(\mathcal{G}_{\mathcal{Z}})} \right\rfloor \\
(\text{as } p^{w_{\mathcal{Z}}(X)/\tau(\mathcal{G}_{\mathcal{Z}})} \geq 1) \quad &\geq \prod_{X\in e_R(\mathcal{Z})} \frac{p^{w_{\mathcal{Z}}(X)/\tau(\mathcal{G}_{\mathcal{Z}})}}{2} \\
(\text{as } |e_R(\mathcal{Z})| \leq |schema(R)| \leq \alpha) \quad &\geq \frac{1}{2^\alpha} \prod_{X\in e_R(\mathcal{Z})} p^{w_{\mathcal{Z}}(X)/\tau(\mathcal{G}_{\mathcal{Z}})} \\
&\geq p^{1/\tau(\mathcal{G}_{\mathcal{Z}})}/2^\alpha
\end{aligned}$$

where the last inequality used the fact that $w_{\mathcal{Z}}$ is a fractional vertex cover of $G_{\mathcal{Z}}$ (and, hence, $\sum_{X\in e_R(\mathcal{Z})} w_{\mathcal{Z}}(X) \geq 1$).

E Completing the Proof of Thm. 4.1

Chernoff Bounds. The following Chernoff bounds (proved in [24]) will be useful:

Lemma E.1. *Let $X_1, X_2, ..., X_f$ be independent Bernoulli random variables that are identically distributed. Define $X = \sum_{i\in[f]} X_i$. For any $\gamma \geq 2$, it holds that*

$$\mathbf{Pr}[X \geq \gamma \cdot \mathbf{E}[X]] \leq \exp(-\gamma \cdot \mathbf{E}[X]/6). \tag{30}$$

For any $0 < \gamma < 1$, it holds that

$$\mathbf{Pr}[X \geq (1+\gamma)\,\mathbf{E}[X]] \leq \exp(-\gamma^2 \cdot \mathbf{E}[X]/3). \tag{31}$$

Fix an arbitrary real value γ satisfying $0 < \gamma < 1$ and any positive real values $p \geq 1$ and β. Let us specialize the setup of random variables in the above lemma as follows:

- $\mathbf{Pr}[X_i = 1] = 1/p$ for each $i \in [f]$;
- $f \leq F$ where $F \geq \beta \cdot p \log p$.

We claim:

$$\mathbf{Pr}[X \geq (1+\gamma)F/p] \leq \exp(-\Omega(\gamma^2\beta \cdot \log p)) \tag{32}$$

where the hidden constant in the big-Ω does not depend on γ, p, or β. Next, we will prove the claim by distinguishing two cases.

Case 1: $\mathbf{E}[X] \geq F/(2p)$. We have:

$$\mathbf{Pr}[X \geq (1+\gamma)F/p]$$

$$\begin{aligned}
(\text{as } \mathbf{E}[X] = f/p \le F/p) \quad &\le \quad \mathbf{Pr}[X \ge (1+\gamma)\,\mathbf{E}[X]] \\
(\text{by (31)}) \quad &\le \quad \exp(-\gamma^2 \cdot \mathbf{E}[X]/3) \\
(\text{as } \mathbf{E}[X] \ge F/(2p)) \quad &\le \quad \exp(-\gamma^2 \cdot F/(6p)) \\
(\text{as } F \ge \beta \cdot p \log p) \quad &\le \quad \exp(-(\gamma^2\beta/6) \cdot \log p)
\end{aligned}$$

as claimed.

Case 2: $\mathbf{E}[X] < F/(2p)$. We can derive:

$$\begin{aligned}
\mathbf{Pr}\,[X \ge (1+\gamma)F/p] \quad &= \quad \mathbf{Pr}\left[X \ge \frac{(1+\gamma)}{\mathbf{E}[X]/(F/p)} \cdot \mathbf{E}[X]\right] \\
\left(\text{by (30), noticing } \frac{(1+\gamma)}{\mathbf{E}[X]/(F/p)} \ge 2\right) \quad &\le \quad \exp(-(1+\gamma) \cdot F/(6p)) \\
(\text{as } F \ge \beta \cdot p \log p \text{ and } \gamma > 0) \quad &\le \quad \exp(-(\beta/6) \cdot \log p) \\
(\text{as } \gamma < 1) \quad &\le \quad \exp(-\Omega(\gamma^2\beta \cdot \log p))
\end{aligned}$$

as claimed.

Completing the Analysis of Our Message-Routing Algorithm. As discussed in Section 4, it remains to prove that our algorithm errs with probability at most $1/p^c$ where c can be an arbitrarily large constant chosen before running the algorithm.

For the algorithm to err in the first round, $|P_i(j)|$ must exceed λ (see its value in (24)) for some $i, j \in [p]$. Set $f = |P_i|$ and $F = p + L/\delta'$; we thus have $f \le F$ by (23). For each $k \in [f]$, introduce a Bernoulli random variable X_k that equals 1 if the k-th packet of P_i is relayed to machine j, and 0 otherwise. Thus, $\mathbf{Pr}[X_k = 1] = 1/p$, and $|P_i(j)| = \sum_{k\in[f]} X_k$.

Suppose that $L \ge \beta \cdot p \log p$ where β is a constant to be decided later. Note that this means $F > L/\delta' \ge (\beta/\delta')p \log p$. We can derive:

$$\begin{aligned}
\mathbf{Pr}[|P_i(j)| \ge \lambda] \quad &\le \quad \mathbf{Pr}\left[|P_i(j)| \ge \left(1 + \frac{L}{\delta' p}\right)(1+\delta/2)\right] \qquad (\text{applying (24)}) \\
&= \quad \mathbf{Pr}\,[|P_i(j)| \ge (1+\delta/2) \cdot F/p] \\
(\text{by (32)}) \quad &\le \quad \exp(-\Omega(\delta^2 \cdot (\beta/\delta') \cdot \log p)) \\
&= \quad \exp(-\Omega(\delta^3 \cdot \beta \cdot \log p)) \qquad (33)
\end{aligned}$$

which can be made less than $1/p^{c_0}$ for an arbitrarily large constant c_0 by setting β sufficiently large (recall that δ is a constant).

How likely the algorithm errs in the second round is somewhat less obvious. For each $j \in [p]$, denote by P_j^+ the set of packets — produced from $\mathbf{M}[1, j]$, $\mathbf{M}[2, j]$, ..., $\mathbf{M}[p, j]$ combined — destined for machine j (i.e., those packets have j as the recipient). Because of (22), $|P_j^+|$, the number of packets in P_j^+, is at most $p + L/\delta'$, following a derivation similar to that in (23). The algorithm errs in the second round only if $|P_i^*(j)| > \lambda$ for some $i, j \in [p]$. Observe that $|P_i^*(j)|$ is precisely the number of packets in P_j^+ that have i as their relay machines. Now, set $f = |P_j^+|$ and $F = p + L/\delta'$; we thus have $f \le F$ as just explained. For each $k \in [f]$, introduce a Bernoulli random variable X_k that equals 1 if the k-th packet of P_j^+ is relayed to machine i, and 0 otherwise. Thus, $\mathbf{Pr}[X_k = 1] = 1/p$, and $|P_i^*(j)| = \sum_{k\in[f]} X_k$. Following a derivation similar to the one leading to (33), we obtain

$$\mathbf{Pr}[|P_i^*(j)| \ge \lambda] \quad \le \quad \exp(-\Omega(\delta^3 \cdot \beta \cdot \log p)) \qquad (34)$$

We can now conclude from (33) and (34) that, when $L \ge \beta \cdot p \log p$ for a sufficiently large β, our algorithm errs with a probability at most $1/p^c$.

F Proof of Theorem 4.2

Recall that the input collection $\mathscr{I}(N)$ of token passing has $p(p-1)$ inputs (S, x, y), where S is a set of N elements (same for all inputs), and x and y are distinct integers in $[p]$ (the inputs differ in the pair (x, y)). Further recall that every input (S, x, y) has only one legal initial state, where every machine is informed of the values x and y, and all the $N/p = \lceil p^{1-\epsilon} \rceil$ tokens are placed on machine x. The goal of an algorithm is to move all tokens from machine x to machine y.

Let $\mathcal{A}$ be a two-round communication oblivious algorithm that succeeds with probability at least $2/3$ on each every input in $\mathscr{I}(N)$. We will prove that $\mathcal{A}$ must entail a load of $\Omega(p^{1-\epsilon/2})$ with probability at least $1/2$.

As $\mathcal{A}$ performs two rounds, its communication pattern can be represented as two $p \times p$ matrices $\mathbf{L}_1$ and $\mathbf{L}_2$. Specifically, for each $r = 1$ and 2, the value $\mathbf{L}_r[i, j]$ (where $i, j \in [p]$) is the number of words that machine i sends to machine j in round r. If $\mathcal{A}$ is randomized, then $\mathbf{L}_1$ and $\mathbf{L}_2$ are random variables. For each $r = 1$ and 2, we define a $p \times p$ matrix $\mathbf{L}_r^*$ from $\mathbf{L}_r$ as follows:

- $\mathbf{L}_r^*[i, j] = \mathbf{L}_r[i, j]$ for any $i, j \in [p]$ satisfying $i \neq j$.
- $\mathbf{L}_r^*[i, i] = \lceil p^{1-\epsilon} \rceil$ for each $i \in [p]$.

We remind the reader that $\mathbf{L}_r[i, i] = 0$ for all $i \in [p]$, by definition of communication pattern. The matrix $\mathbf{L}_r^*$ thus defined is a random variable decided by $\mathbf{L}_r$.

By communication obliviousness, the distribution of $(\mathbf{L}_1, \mathbf{L}_2)$ is the same for all the inputs in $\mathscr{I}(N)$. Let $\mathcal{L}$ be the set of ordered pairs $(\mathbf{\Lambda}_1, \mathbf{\Lambda}_2)$ satisfying

- each of $\mathbf{\Lambda}_1$ and $\mathbf{\Lambda}_2$ is a $p \times p$ matrix;
- $\mathbf{Pr}[(\mathbf{L}_1^*, \mathbf{L}_2^*) = (\mathbf{\Lambda}_1, \mathbf{\Lambda}_2)] > 0$, namely, $\mathcal{A}$ exhibits with a non-zero probability a pattern $(\mathbf{L}_1, \mathbf{L}_2)$ such that the corresponding $(\mathbf{L}_1^*, \mathbf{L}_2^*)$ equals $(\mathbf{\Lambda}_1, \mathbf{\Lambda}_2)$;
- the following inequality holds for all $k \in [p]$:

$$\Big(\sum_{i \in [p]} \mathbf{\Lambda}_1[i, k] \Big) \cdot \Big(\sum_{j \in [p]} \mathbf{\Lambda}_2[k, j] \Big) \leq p^{2-\epsilon}/24. \tag{35}$$

The core of our argument is to prove

$$\mathbf{Pr}[(\mathbf{L}_1^*, \mathbf{L}_2^*) \notin \mathcal{L}] \geq 1/2. \tag{36}$$

Note that whenever $(\mathbf{L}_1^*, \mathbf{L}_2^*) \notin \mathcal{L}$, it follows from the definition in (35) that there is at least one $k \in [p]$ such that

$$\Big(\sum_{i \in [p]} \mathbf{L}_1^*[i, k] \Big) \cdot \Big(\sum_{j \in [p]} \mathbf{L}_2^*[k, j] \Big) > p^{2-\epsilon}/24$$

under which one of the following inequalities must be true:

- $\sum_{i \in [p]} \mathbf{L}_1^*[i, k] > p^{1-\epsilon/2}/\sqrt{24}$, or
- $\sum_{j \in [p]} \mathbf{L}_2^*[k, j] > p^{1-\epsilon/2}/\sqrt{24}$.

In the former case, we have

$$\sum_{i \in [p]} \mathbf{L}_1[i, k] \quad = \quad \Big(\sum_{i \in [p]} \mathbf{L}_1^*[i, k] \Big) - \mathbf{L}_1^*[k, k] = \Big(\sum_{i \in [p]} \mathbf{L}_1^*[i, k] \Big) - \lceil p^{1-\epsilon} \rceil = \Omega\Big(p^{1-\epsilon/2}\Big).$$

This means that the load of $\mathcal{A}$ is $\Omega(p^{1-\epsilon/2})$ as $\sum_{i \in [p]} \mathbf{L}_1[i, k]$ is the number of words received by machine k in the first round. In the latter case, following a similar derivation one can obtain $\sum_{j \in [p]} \mathbf{L}_2[k, j] = \Omega(p^{1-\epsilon/2})$. This also implies that $\mathcal{A}$ has load $\Omega(p^{1-\epsilon/2})$ because $\sum_{j \in [p]} \mathbf{L}_2[k, j]$ is the number of words sent by machine k in the second round. Therefore, with probability at least $1/2$, algorithm $\mathcal{A}$ demands a load of $\Omega(p^{1-\epsilon/2})$, which completes the proof of Theorem 4.2.

Proof of Inequality (36)**.** Assume for contradiction purposes that (36) does not hold, which means $\mathbf{Pr}[(\mathbf{L}_1^*, \mathbf{L}_2^*) \in \mathcal{L}] > 1/2$. For every input $(S, x, y) \in \mathscr{I}(N)$, we know

$$\mathbf{Pr}[\mathcal{A} \text{ succeeds on } (S, x, y) \mid (\mathbf{L}_1^*, \mathbf{L}_2^*) \in \mathcal{L}] \geq 1/6. \tag{37}$$

Otherwise, $\mathbf{Pr}[\mathcal{A} \text{ succeeds on } (S, x, y)] < 1/6 + \mathbf{Pr}[(\mathbf{L}_1^*, \mathbf{L}_2^*) \notin \mathcal{L}] < 2/3$, contradicting the fact that $\mathcal{A}$ succeeds on any input with probability at least 2/3.

Lemma F.1. *If $\mathcal{A}$ succeeds on $(S, x, y) \in \mathscr{I}(N)$ under the condition $(\mathbf{L}_1^*, \mathbf{L}_2^*) = (\mathbf{\Lambda}_1, \mathbf{\Lambda}_2)$, then*

$$\sum_{k \in [p]} \min\left\{\mathbf{\Lambda}_1[x,k], \mathbf{\Lambda}_2[k,y]\right\} \geq p^{1-\epsilon}. \tag{38}$$

Proof. Let us rewrite the left hand side of (38) as

$$\begin{aligned}
&\sum_{k \in [p]} \min\left\{\mathbf{\Lambda}_1[x,k], \mathbf{\Lambda}_2[k,y]\right\} \\
&= \Big(\sum_{k \in [p] \setminus \{x,y\}} \min\left\{\mathbf{\Lambda}_1[x,k], \mathbf{\Lambda}_2[k,y]\right\}\Big) + \min\left\{\mathbf{\Lambda}_1[x,y], \mathbf{\Lambda}_2[y,y]\right\} + \min\left\{\mathbf{\Lambda}_2[x,x], \mathbf{\Lambda}_2[x,y]\right\} \\
&= \Big(\sum_{k \in [p] \setminus \{x,y\}} \min\left\{\mathbf{\Lambda}_1[x,k], \mathbf{\Lambda}_2[k,y]\right\}\Big) + \min\left\{\mathbf{\Lambda}_1[x,y], \lceil p^{1-\epsilon} \rceil\right\} + \min\left\{\lceil p^{1-\epsilon} \rceil, \mathbf{\Lambda}_2[x,y]\right\}. \quad (39)
\end{aligned}$$

If a machine $k \in [p] \setminus \{x, y\}$ receives a token from machine x in the first round and sends the token to machine y in the second round, we say that the token is *relayed* by machine k. Under the condition $(\mathbf{L}_1^*, \mathbf{L}_2^*) = (\mathbf{\Lambda}_1, \mathbf{\Lambda}_2)$, machine k can relay no more than $\min\{\mathbf{\Lambda}_1[x,k], \mathbf{\Lambda}_2[k,y]\}$ tokens. On the other hand, if a token is not relayed by any machines, it needs to be sent from machine x to machine y directly (in either the first or the second round). The number of tokens that can be sent this way is at most

$$\min\left\{\mathbf{\Lambda}_1[x,y], \lceil p^{1-\epsilon} \rceil\right\} + \min\left\{\lceil p^{1-\epsilon} \rceil, \mathbf{\Lambda}_2[x,y]\right\}.$$

Thus, (39) gives an upper bound on the total number of tokens that $\mathcal{A}$ can move to machine y at the end of the second round. The upper bound must be at least $\lceil p^{1-\epsilon} \rceil$ for $\mathcal{A}$ to succeed on (x, y). □

By putting together (37) and (38), we obtain the following for every pair $(x, y) \in [p] \times [p]$ with $x \neq y$:

$$\mathbf{E}\left[\sum_{k \in [p]} \min\left\{\mathbf{L}_1^*[x,k], \mathbf{L}_2^*[k,y]\right\} \,\middle|\, (\mathbf{L}_1^*, \mathbf{L}_2^*) \in \mathcal{L}\right] \geq \frac{p^{1-\epsilon}}{6} \tag{40}$$

For any $(\mathbf{\Lambda}_1, \mathbf{\Lambda}_2) \in \mathcal{L}$ and any $k \in [p]$, we have:

$$\begin{aligned}
\Big(\sum_{x \in [p]} \mathbf{\Lambda}_1[x,k]\Big) \cdot \Big(\sum_{y \in [p]} \mathbf{\Lambda}_2[k,y]\Big) &= \sum_{x \in [p]} \sum_{y \in [p]} \mathbf{\Lambda}_1[x,k] \cdot \mathbf{\Lambda}_2[k,y] \\
&\geq \sum_{x \in [p]} \sum_{y \in [p]} \min\left\{\mathbf{\Lambda}_1[x,k], \mathbf{\Lambda}_2[k,y]\right\} \\
&\geq \sum_{\substack{x,\, y \,\in\, [p] \\ \text{s.t. } x \neq y}} \min\left\{\mathbf{\Lambda}_1[x,k], \mathbf{\Lambda}_2[k,y]\right\} \qquad (41)
\end{aligned}$$

where the first inequality used the fact that $ab \geq \min\{a, b\}$ holds for any non-negative integers a and b.

We can now derive:

$$\begin{aligned}
& \sum_{k\in[p]} \mathbf{E}\left[\Big(\sum_{x\in[p]} \mathbf{L}_1^*[i,k]\Big)\cdot\Big(\sum_{y\in[p]} \mathbf{L}_2^*[k,j]\Big)\;\Big|\;(\mathbf{L}_1^*,\mathbf{L}_2^*)\in\mathcal{L}\right] \\
\geq\;& \mathbf{E}\Big[\sum_{k\in[p]}\sum_{\substack{x,\,y\,\in\,[p]\\ \text{s.t. } x\neq y}} \min\left\{\mathbf{L}_1^*[x,k],\mathbf{L}_2^*[k,y]\right\}\;\Big|\;(\mathbf{L}_1^*,\mathbf{L}_2^*)\in\mathcal{L}\Big] \\
& \text{(by linearity of expectation and (41))} \\
=\;& \mathbf{E}\Big[\sum_{\substack{x,\,y\,\in\,[p]\\ \text{s.t. } x\neq y}}\sum_{k\in[p]} \min\left\{\mathbf{L}_1^*[x,k],\mathbf{L}_2^*[k,y]\right\}\;\Big|\;(\mathbf{L}_1^*,\mathbf{L}_2^*)\in\mathcal{L}\Big] \\
=\;& \sum_{\substack{x,\,y\,\in\,[p]\\ \text{s.t. } x\neq y}} \mathbf{E}\Big[\sum_{k\in[p]} \min\left\{\mathbf{L}_1^*[x,k],\mathbf{L}_2^*[k,y]\right\}\;\Big|\;(\mathbf{L}_1^*,\mathbf{L}_2^*)\in\mathcal{L}\Big] \\
& \text{(linearity of expectation)} \\
\geq\;& \frac{p^{1-\epsilon}\cdot p(p-1)}{6}. \qquad \text{(by (40))}
\end{aligned}$$

Therefore, there is at least one $k^*\in[p]$ such that

$$\mathbf{E}\left[\Big(\sum_{x\in[p]} \mathbf{L}_1^*[i,k^*]\Big)\cdot\Big(\sum_{y\in[p]} \mathbf{L}_2^*[k^*,j]\Big)\;\Big|\;(\mathbf{L}_1^*,\mathbf{L}_2^*)\in\mathcal{L}\right] \geq \frac{p^{1-\epsilon}(p-1)}{6} \geq p^{2-\epsilon}/12$$

where the last inequality used $p\geq 2$.

However, by the definition of $\mathcal{L}$ (in particular, the inequality (35)), when $(\mathbf{L}_1^*,\mathbf{L}_2^*)\in\mathcal{L}$, the product $\sum_{x\in[p]}\mathbf{L}_1^*[i,k]\cdot\sum_{y\in[p]}\mathbf{L}_2^*[k,j]$ can be at most $p^{2-\epsilon}/24$ for all $k\in[p]$. This contradicts the existence of k^*. Therefore, our assumption $\mathbf{Pr}[(\mathbf{L}_1^*,\mathbf{L}_2^*)\in\mathcal{L}]>1/2$ must be a false one.

Received May 2024; accepted August 2024