The Communication Complexity of Distributed Set-Joins with Applications to Matrix Multiplication

Dirk Van Gucht[1] Ryan Williams[2] David P. Woodruff[3] Qin Zhang[1]

[1]Indiana University Bloomington
[2]Stanford University
[3]IBM Almaden

ABSTRACT

Given a set-comparison predicate $\mathcal{P}$ and given two lists of sets $\mathcal{A} = (A_1, \ldots, A_m)$ and $\mathcal{B} = (B_1, \ldots, B_m)$, with all $A_i, B_j \subseteq [n]$, the *$\mathcal{P}$-set join* $\mathcal{A} \bowtie^{\mathcal{P}} \mathcal{B}$ is defined to be the set $\{(i,j) \in [n] \times [n] \mid \mathcal{P}(A_i, B_j)\}$ ($[n]$ denotes $\{1, 2, \ldots, n\}$). When $\mathcal{P}(A_i, B_j)$ is the condition "$A_i \cap B_j \neq \emptyset$" we call this the *set-intersection-not-empty join* (a.k.a. the composition of $\mathcal{A}$ and $\mathcal{B}$); when $\mathcal{P}(A_i, B_j)$ is "$A_i \cap B_j = \emptyset$" we call it the *set-disjointness join*; when $\mathcal{P}(A_i, B_j)$ is "$A_i = B_j$" we call it the *set-equality join*; when $\mathcal{P}(A_i, B_j)$ is "$|A_i \cap B_j| \geq T$" for a given threshold T, we call it the *set-intersection threshold join*. Assuming $\mathcal{A}$ and $\mathcal{B}$ are stored at two different sites in a distributed environment, we study the (randomized) communication complexity of computing these, and related, set-joins $\mathcal{A} \bowtie^{\mathcal{P}} \mathcal{B}$, as well as the (randomized) communication complexity of computing the exact and approximate value of their size $k = |\mathcal{A} \bowtie^{\mathcal{P}} \mathcal{B}|$. Combined, our analyses shed new insights into the quantitative differences between these different set-joins. Furthermore, given the close affinity of the *natural join* and the set-intersection-not-empty join, our results also yield communication complexity results for computing the natural join in a distributed environment.

Additionally, we obtain new algorithms for computing the distributed set-intersection-not-empty join when the input and/or output is sparse. For instance, when the and output is k-sparse, we improve an $\tilde{O}(kn)$ communication algorithm of (Williams and Yu, SODA 2014). Observing that the set-intersection-not-empty join is isomorphic to *Boolean matrix multiplication* (BMM), our results imply new algorithms for fundamental graph theoretic problems related to BMM. For example, we show how to compute the transitive closure of a directed graph in $\tilde{O}(k^{3/2})$ time, when the transitive closure contains at most k edges. When $k = O(n)$, we obtain a (practical) $\tilde{O}(n^{3/2})$ time algorithm, improving a recent $\tilde{O}(n \cdot n^{\frac{\omega+1}{4}})$ time algorithm (Borassi, Crescenzi, and Habib, arXiv 2014) based on (impractical) fast matrix multiplication, where $\omega \geq 2$ is the exponent for matrix multiplication.

1. INTRODUCTION

In this paper we study the complexity of a variety of distributed set-join operations in the natural setting of two-party communication complexity.

In this introduction, we first specify the different set-joins we consider. We next turn to their distributed computation, and then provide a summary of the main results regarding their communication complexities. We obtain both positive and negative results. For the set-intersection, set-equality, and at-least-T set-joins, we obtain positive results by showing that efficiency gains in communication can be achieved through the use of sophisticated protocols. We also show that several of these protocols are optimal. On the other hand, for the set-disjointness join, and its affiliated superset and subset joins, we obtain negative results: we prove lower bounds on their communication complexity which establish that, in general, the best one can do for these set-joins is to send all the data from one site to the other site and then, at that site, perform the set-join without further communication.

In addition to our communication complexity results, we also describe new and positive results about the computation of Boolean matrix multiplication and transitive closure that follow from the algorithmic techniques developed for computing distributed set-intersection joins (i.e., composition and natural join).

1.1 Set-Joins

Among the most common tasks in querying data are finding associations/links between two data sources. Two classic cases of such tasks are computing the *composition* and the *natural join* of two relations[10].[1] More specifically, let $\mathcal{A}$ be a relation defined over attributes (I, L) and let $\mathcal{B}$ be a relation defined over attributes (L, J). Let $[z]$ denote $\{1, 2, \ldots, z\}$ for a natural number z. Assume for simplicity that $dom(I) = dom(J) = [m]$, and $dom(L) = [n]$. Then $\mathcal{A} \subseteq [m] \times [n]$ and $\mathcal{B} \subseteq [n] \times [m]$. The *composition* of $\mathcal{A}$ and $\mathcal{B}$ and the *natural join* of $\mathcal{A}$ and $\mathcal{B}$ are respectively defined

[1]The composition and the natural join were first introduced to the database community by Codd in his 1970 celebrated paper on the Relational Model [10]. Actually, compared to the natural join, the composition operator has a much longer history: its definition and properties were first studied by De Morgan, Schröder, Peirce, and Tarski in the late 19th and early 20th centuries [35]. More recently, since composition can be considered as the canonical graph traversal/navigation operation, it has been studied intensively in the context of semi-structured tree and graph databases [2, 14, 42]. Furthermore, since repeated application of the composition operation corresponds to determining graph connectivity, it is at the core of computing the *transitive closure* of a graph.

as

$$\mathcal{A} \circ \mathcal{B} := \{(i,j) \mid \exists \ell : (i,\ell) \in \mathcal{A} \ \&\ (\ell,j) \in \mathcal{B}\}, \text{ and}$$
$$\mathcal{A} \bowtie \mathcal{B} := \{(i,\ell,j) \mid (i,\ell) \in \mathcal{A} \ \&\ (\ell,j) \in \mathcal{B}\}.$$

Clearly, one can view the natural join as an *adornment* of the composition, i.e., for each composition pair $(i,j) \in \mathcal{A} \circ \mathcal{B}$, the natural join adorns (i,j) with its corresponding *witnesses* ℓ. The composition operator can also be viewed as *Boolean matrix multiplication* when $\mathcal{A}$ and $\mathcal{B}$ are construed as Boolean matrices in a natural way; see the discussion in Section 1.3.

The composition and natural join operations can be defined in an alternative way: if we let A_i be the "projection/witness" set $\{\ell \mid (i,\ell) \in \mathcal{A}\}$ and let B_j be $\{\ell \mid (\ell,j) \in \mathcal{B}\}$, then we have

$$\mathcal{A} \circ \mathcal{B} = \{(i,j) \mid A_i \cap B_j \neq \emptyset\}, \text{ and}$$
$$\mathcal{A} \bowtie \mathcal{B} = \{(i,\ell,j) \mid \ell \in A_i \cap B_j\}.$$

Because of the not-empty set-intersection conditions present in the definition of the composition and the natural join, we will call them generically *set-intersection-not-empty joins*, or more succinctly, *set-intersection joins*. We will reserve the acronym SIJ for the composition and when necessary, explicitly refer to the natural join.

Set-intersection joins are by no means the only set-joins one may consider. Imagine, for example, that $\mathcal{A}$ stores job-applicants and their skills and that $\mathcal{B}$ stores the required skills of job-openings. In this case, a useful relationship between a job-applicant and a job-opening may exist if *all* of the skills of an applicant include (contain) all the skills required for a job. We may also wish that these skills are precisely the same. Formally, we want to compute the sets $\{(i,j) \mid A_i \supseteq B_j\}$ and $\{(i,j) \mid A_i = B_j\}$, which we call the *superset join* and the *set-equality join*, respectively.

There are other set-joins of this flavor one may consider. Table 1 summarizes the set-joins of interest for this paper. Besides the set-intersection, superset, and set-equality joins, we consider the *Subset Join* $\mathcal{A} \bowtie^{\subseteq} \mathcal{B}$ which can be viewed as an inverse of the superset join, the *Set-Disjointness Join* $\mathcal{A} \bowtie^{\cap=\emptyset} \mathcal{B}$ which can be viewed as the negation of the set-intersection joins, the *Not-Superset Join* $\mathcal{A} \bowtie^{\not\supseteq} \mathcal{B}$ and the *Not-Subset Join* $\mathcal{A} \bowtie^{\not\subseteq} \mathcal{B}$ which can be viewed as negations of the superset and subset joins, respectively, and the At-Least-T join $\mathcal{A} \bowtie^{\geq T} \mathcal{B}$ which joins the A_i and B_j pairs whose intersection contain at least T elements.

Clearly, not all these set-joins paper are independent. Indeed, straightforward equivalences between the above permit us to focus only on four classes of set-joins: the set-intersection, set-disjointness, set-equality, and at-least-T join operations.[2] Observe that, since, $\mathcal{A} \bowtie^{=} \mathcal{B} = (\mathcal{A} \bowtie^{\subseteq} \mathcal{B}) \cap (\mathcal{A} \bowtie^{\supseteq} \mathcal{B})$, we could also omit the set-equality join from our study. We will see however that it is more insightful if we consider the set-equality join as a primitive set-join in its own right.

Applications, logical and algebraic properties, algorithms and data structures (and their implementations), and empirical studies of some or all of these set-joins have been described in [3, 5, 9, 13, 16–18, 23, 26, 28, 36–38, 46].

One of the main threads that runs through these papers is that, compared to the set-intersection joins, the class of the super, subset, and set-disjointness joins are considered and suspected to be computationally more expensive in time and space utilization, even though we are not aware of formal proof of this. We therefore think that our communication-complexity results are particularly meaningful since they **do** indeed formally prove, and therefore inform and substantiate, these suspected computational differences.

1.2 Distributed Set-Joins and their Communication Complexity

Distributed computations of several set-joins have been studied extensively. Mishra and Eich [29] (Section 5) and Kossman [21]) surveyed distributed set-intersection joins (principally, natural joins). More recently, and primarily because of the introduction of MapReduce, there has been renewed interest in the distributed and parallel computations of such joins (see for example [1, 6, 33].) On the other hand, there is far less work on distributed and parallel computation for the other set-joins operations we consider in this paper: we mention here the work in [4, 24, 43] wherein algorithms, data structures, and implementation are described for set-joins other than set-intersection joins.

In this paper, we study the complexity of distributed set-join operations in the natural setting of two-party communication complexity. One party, Alice, holds the collection $\mathcal{A} = \{A_1, \ldots, A_m\}$, and another party, Bob, holds $\mathcal{B} = \{B_1, \ldots, B_{m'}\}$ in another remote location. To ease the presentation we assume $m = m'$, but all our results can be easily adapted to the case where $m \neq m'$. Alice and Bob wish to exchange a *minimum* number of bits in order to determine information about a set-join operation on $\mathcal{A}$ and $\mathcal{B}$ (or compute the set-join itself, if that is a small set). We allow Alice and Bob to occasionally make random choices in their algorithms. Informally, we call the minimum number of bits needed to be exchanged to compute an operation with high probability (as a function of n and m) the *randomized communication complexity* of that operation. Further background and definitions are in Section 2.

We prove new upper and lower bounds on the communication required to compute these set-joins, building on sophisticated algorithms and tools from prior work. Our results establish formal quantitative differences between the complexities of these various set-join operations; in particular, their respective communication complexities are not the same.

We are aware of only one other theoretical result with a similar type of message: namely, Leinders and Van den Bussche [23] exhibit a way in which the set-intersection join and the superset-join can be seen as quantitatively different. Consider the problem of determining if $|\mathcal{A} \bowtie \mathcal{B}| > 0$ and the problem of determining if $|\mathcal{A} \bowtie^{\supseteq} \mathcal{B}| > 0$. Establishing that $|\mathcal{A} \bowtie \mathcal{B}| > 0$ is equivalent to establishing $|\mathcal{A} \ltimes \mathcal{B}| > 0$.[3] Since the size of $\mathcal{A} \ltimes \mathcal{B}$ is linear in the size of $\mathcal{A}$ and $\mathcal{B}$, the decision problem $|\mathcal{A} \bowtie \mathcal{B}| > 0$ can be solved in linear space, even though the size of $\mathcal{A} \bowtie \mathcal{B}$ itself may be quadratic. In sharp contrast, Leinders and Van den Bussche showed that for **any** expression E in the relational algebra with the prop-

[2] Recalling that $A_i, B_j \subseteq [n]$, let $\overline{A_i} = [n] - A_i$ and $\overline{B_j} = [n] - B_j$. Then (1) the conditions $A_i \supseteq B_j$ and $\overline{A_i} \cap B_j = \emptyset$ are equivalent; (2) the conditions $A_i \subseteq B_j$ and $A_i \cap \overline{B_j} = \emptyset$ are equivalent; (3) the conditions $A_i \not\supseteq B_j$ and $\overline{A_i} \cap B_j \neq \emptyset$ are equivalent; and (4) the conditions $A_i \not\subseteq B_j$ and $A_i \cap \overline{B_j} \neq \emptyset$ are equivalent.

[3] Recall that $\mathcal{A} \ltimes \mathcal{B}$, i.e. the semi-join of $\mathcal{A}$ and $\mathcal{B}$, is $\{(i,\ell) \mid \ell \in A_i \ \&\ A_i \cap B \neq \emptyset\}$, where $B = \pi_{[L]}(\mathcal{B})$, i.e, the set of ℓ-values in all L columns of $\mathcal{B}$.

Name	Notation	Definition
Natural Join	$\mathcal{A} \bowtie \mathcal{B}$	$\{(i, \ell, j) \mid \ell \in A_i \cap B_j\}$
Composition	$\mathcal{A} \bowtie^{\cap \neq \emptyset} \mathcal{B}$	$\{(i, j) \mid A_i \cap B_j \neq \emptyset\}$
Superset Join	$\mathcal{A} \bowtie^{\supseteq} \mathcal{B}$	$\{(i, j) \mid A_i \supseteq B_j\}$
Set-Equality Join	$\mathcal{A} \bowtie^{=} \mathcal{B}$	$\{(i, j) \mid A_i = B_j\}$
Subset Join	$\mathcal{A} \bowtie^{\subseteq} \mathcal{B}$	$\{(i, j) \mid A_i \subseteq B_j\}$
Set-Disjointness Join	$\mathcal{A} \bowtie^{\cap = \emptyset} \mathcal{B}$	$\{(i, j) \mid A_i \cap B_j = \emptyset\}$
Not-Superset Join	$\mathcal{A} \bowtie^{\not\supseteq} \mathcal{B}$	$\{(i, j) \mid A_i \not\supseteq B_j\}$
Not-Subset Join	$\mathcal{A} \bowtie^{\not\subseteq} \mathcal{B}$	$\{(i, j) \mid A_i \not\subseteq B_j\}$
At-Least-T-Join	$\mathcal{A} \bowtie^{\geq T} \mathcal{B}$	$T \in \mathbb{N}, \{(i, j) \mid \lvert A_i \cap B_j \rvert \geq T\}$

Table 1: Joins considered in this paper.

erty that $|\mathcal{A} \bowtie^{\supseteq} \mathcal{B}| > 0$ is equivalent to $|E| > 0$, such an E must have a sub-expression which, when evaluated, generates an intermediate result whose size is *quadratic* in the size of $\mathcal{A}$ and $\mathcal{B}$. In other words, in contrast with the set-intersection join, there is no linear space expression E in the relational algebra for the $|\mathcal{A} \bowtie^{\supseteq} \mathcal{B}| > 0$ decision problem for the superset join.

1.3 Main Results

To describe the main results about the communication-complexity of a set-join, we need to define its *enumerative version* and its *counting version*. Given a set-join $\mathcal{A} \bowtie^{\mathcal{P}} \mathcal{B}$, its enumerative version refers to the communication-complexity of determining (enumerating) the set of pairs $\{(i, j) | \mathcal{P}(A_i, B_j)\}$, and its counting version refers to the communication-complexity of determining the count $k = |\{(i, j) | \mathcal{P}(A_i, B_j)\}|$ of this set. For the counting version, we furthermore discern between the *exact* and the *approximate* determination of k.

Notice that a lower bound on the counting version implies a lower bound on the corresponding enumerative version, and analogously, an upper bound on the enumerative version implies and upper bound on its corresponding counting version. Actually, in many cases, the enumerative and counting versions are the same. Therefore, in the statements of our results we will not always explicitly differentiate between them.

The main contributions of our paper are

1. to the best of our knowledge, the first formal evidence of the suspected quantitative differences between the "hardness" of computing different set-join operations in terms of a computational complexity model;
2. the determination of matching lower and upper communication bound for the enumerative set-intersection join;
3. the determination of an output-sensitive upper communication bound for the enumerative natural join;
4. the determination of upper and lower communication bounds for approximating the size of set-intersection join;
5. the determination of matching lower and upper communication bound for the enumerative and counting versions of superset, subset, set-disjointness, and set-equality set-join operations; and
6. new algorithmic techniques and insights for Boolean matrix multiplication (isomorphic to set-intersection join), which lead to an improved algorithm for the fundamental problem of computing transitive closure of a directed graph in terms of running time.

Our main results for various joins are described in Table 2. In the rest of this section we illustrate our main results in more details. Below are some global notations we use in our paper.

- n to denote the size of the item universe,
- m to denote the number of tuples in a table to be joined,
- k to denote the size of the output,
- s to denote the sparsity of each tuple in the set-joins; that is, $s = \max\{|A_i|, |B_j| \mid i, j \in [m]\}$.

For a matrix M, let $M_{i,j}$ denote the entry at i-th row and j-th column; let $M_{i,*}$ denote the i-th row and let $M_{*,j}$ denote the j-th column. Let $\mathbf{nnz}(M)$ denote the number of non-zero entries in M. For simplicity, we use $\tilde{O}(f)$ to denote $f \cdot \text{poly}\log(fmn)$. That is, $\tilde{O}$ suppresses polylogarithmic factors. We use $\tilde{\Theta}(f)$ to denote at most $\tilde{O}(f)$ and at least $\Omega(f)$.

An isomorphic way to view the composition operator $\mathcal{A} \circ \mathcal{B}$ is as a *Boolean matrix multiplication* (BMM). Construe $\mathcal{A}$ as an $m \times n$ Boolean matrix P such that $P_{i,\ell} = 1$ if and only if $(i, \ell) \in \mathcal{A}$. Similarly, construe $\mathcal{B}$ as an $n \times m$ matrix Q such that $Q_{\ell,j} = 1$ if and only if $(\ell, j) \in \mathcal{B}$. Then $(P \cdot Q)_{i,j} \geq 1$ if and only if $(i, j) \in \mathcal{A} \circ \mathcal{B}$.

We start with our results on the set-intersection join (i.e., Boolean matrix multiplication). We first consider computing the set-intersection join exactly, presenting what we consider to be the main result of this paper; certainly it is the most surprising. Recent work on the communication complexity of matrix multiplication demonstrated that, when Alice (holding $\mathcal{A} = \{A_1, \ldots, A_m\}, A_i \subseteq [n]$) and Bob (holding $\mathcal{B} = \{B_1, \ldots, B_m\}, B_j \subseteq [n]$) construe $\mathcal{A}$ and $\mathcal{B}$ as $m \times n$ and $n \times m$ matrices, respectively, they can compute their product $\mathcal{A} \cdot \mathcal{B}$ over an arbitrary finite field with randomized communication $\tilde{O}(kn)$, where k is the number of nonzeroes in the matrix product [44]. Intuitively, this upper bound looks essentially optimal. It seems impossible to attain $o(kn)$ communication: shouldn't Alice and Bob need to communicate $\Omega(n)$ bits in the worst case, for every nonzero entry in the output (exchanging the relevant n-bit vectors)? Here, we show that $o(kn)$ communication is in fact possible, and the communication algorithms can be efficiently implemented as well.

	enumeration / exact counting		c-approximate counting	
	upper bound	lower bound	upper bound	lower bound
set-intersection join (composition), not-superset/not-subset join	$\tilde{\Theta}(\min(ms, \sqrt{skn}) + n)$ *		$\tilde{O}(n/\epsilon^2)$ ($c = 1+\epsilon$)	$\Omega(n/\epsilon^2)$ (1-way, $c = 1+\epsilon$) $\Omega(n/\epsilon^{\frac{2}{3}})$ (2-way, $c = 1+\epsilon$)
set-equality join	$\Theta(m \operatorname{ilog}^{\Theta(r)} m + \log\log n)$		$\Theta(m \operatorname{ilog}^{\Theta(r)} m + \log\log n)$ (any $c > 0$)	
set-disjointness join subset/superset join	$\Theta(mn)$		$\Theta(mn)$ (any $c > 0$)	
at-least-T join	$\Theta(mn)$		$O(mn)$	$\Omega(mn)$ (1-way, any $c > 0$) $\Omega(m\sqrt{nT})$ (2-way, any $c > 0$)

Table 2: Our results for joins. *This result also applies to natrual join after converting it to the form of set-intersection join by projections (see discussions in Section 1.1). $\tilde{O}(f)$ denotes $f \cdot \operatorname{poly}\log(fmn)$; $\tilde{\Theta}(f)$ denotes at most $\tilde{O}(f)$ and at least $\Omega(f)$; and $\operatorname{ilog}^r m$ means $\log \ldots \log m$ with r logs.

Result 1 (Theorem 3 in Section 3) *The randomized communication complexity of computing set-intersection join (equivalently, computing the product of two Boolean matrices $P \in \{0,1\}^{m\times n}$ and $Q \in \{0,1\}^{n\times m}$) is $\tilde{O}(\sqrt{k} \cdot n)$, when n is the universe size and k is the output join size. Furthermore, if the sparsity of $\mathcal{A}, \mathcal{B}$ is at most s, we can obtain an algorithm with $\tilde{O}(\min(ms, \sqrt{skn}) + n)$ bits of communication. Finally, this algorithm is optimal, up to polylogarithmic factors.*

The algorithm (Algorithm 1) in Theorem 3 applies both the Count-Sketch algorithm [12] and the ℓ_0-sketch algorithm [20] in novel ways. The high-level idea is to consider two kinds of column vectors that may appear in the output matrix: columns which are "sparse" and columns which are "dense". Roughly speaking, "dense" columns have at least $\sqrt{k}$ non-zeroes, and "sparse" columns are those which are not dense. First, we can determine which resulting columns will be sparse and dense using an ℓ_0-sketch, requiring little communication between the two parties. Dense columns can be handled efficiently by having Bob send $O(\sqrt{k})$ of the relevant columns to Alice (there are $O(\sqrt{k})$ columns with greater than $\sqrt{k}$ nonzeroes in the output matrix, by simple counting). Sparse columns can be computed using a Count-Sketch matrix with only $\tilde{O}(\sqrt{k})$ rows: Alice can multiply such a matrix with her own, and send the resulting $\tilde{O}(\sqrt{k}n)$ data to Bob, who can then recover the sparse columns of the matrix product.

As a "by-product", which is in fact fundamental for databases, this algorithm plus a post-processing (Algorithm 2 in Section 3) can output all witnesses of all set-intersections, that is, all ℓ's s.t. $\ell \in A_i \cap B_j$ for any $A_i \in \mathcal{A}, B_j \in \mathcal{B}$, which in fact solves the corresponding natural join problem (see the discussion on the equivalence between natural join and set-intersection join in Section 1.1).

Result 2 (Corollary 1 in Section 3) *Under the notations in Result 1, the algorithm can also be used to solve the corresponding natural join problem with $\tilde{O}(\min(ms, \sqrt{skn}) + n)$ communication.*

With more algorithmic cleverness, this algorithm can be efficiently implemented as well:

Result 3 (Theorem 13 in Section 4) *There is a randomized algorithm for multiplying $P \in \{0,1\}^{m\times n}$ and $Q \in \{0,1\}^{n\times m}$ that runs in $\tilde{O}(k + k^{1/2}(\mathbf{nnz}(P) + \mathbf{nnz}(Q)))$ time and succeeds with probability $1 - 1/n$, where k is the number of nonzero entries in the output.*[4]

Result 3 has natural applications in situations where one anticipates that the output may be sparse. For example, in Section 4 we can readily apply Result 3 to obtain:

Result 4 (Corollary 4 in Section 4) *There is a randomized algorithm for computing the transitive closure of an $n \times n$ Boolean matrix A which runs in $\tilde{O}(k^{1.5})$ time and succeeds with probability $1 - 1/n$, where k is the number of edges in the transitive closure.*

Hence any transitive closure which contains $O(\mathbf{nnz}(A))$ edges can be computed in $\tilde{O}(\mathbf{nnz}(A)^{1.5})$ time. This, in the case that $\mathbf{nnz}(A) \le n$, improves over a recent result of Borassi, Crescenzi, and Habib [7], who achieved $\tilde{O}(\mathbf{nnz}(A) \cdot n^{\frac{\omega+1}{4}})$ time, where $\omega \approx 2.373$ is the exponent of fast matrix multiplication. Moreover, their algorithm relies on fast matrix multiplication algorithms while ours does not, and so ours presumably have practical advantages. One can recognize whether a given graph is a comparability graph in the same time as computing the transitive closure, as observed in [7]. (See Section 4 for more details.) Corollary 4 therefore improves the algorithm of [7] for this task as well.

Next, we turn to approximating the *size* of the set-intersection join. Let c-SIJ be the communication problem of approximating the number of pairs (i, j) such that $A_i \cap B_j \neq \varnothing$ to within a multiplicative factor of c. We give nearly tight upper and lower bounds on the *one-way* communication complexity of this problem, where all messages are sent from Alice to Bob, and Bob computes the answer. The true answer is surprisingly low:

Result 5 (Theorem 5, 6 in Section 3) *The one-way randomized communication of $(1+\epsilon)$-SIJ is $\tilde{\Theta}(n/\epsilon^2)$.*

The upper bound applies the ℓ_0-sketch algorithm ([20]) in a simple way; the lower bound follows by a reduction from the Indexing function (see the Section 2 for definitions). We note that since our upper bound is a sketching algorithm and our lower bound is for 1-way communication, they apply to the *data stream model* of computation in which one sees the sets A_i and B_j one at a time in an arbitrary order and one would like to compute $(1+\epsilon)$-SIJ using a small amount of

[4]The algorithm can be implemented in any reasonable computational model with random access to the input, such as the RAM model.

memory. Our results imply that $\tilde{\Theta}(n/\epsilon^2)$ bits of memory is necessary and sufficient.

In the general (two-way) communication setting, we can still prove a lower bound depending on n and ϵ, although we do not have a better upper bound than that for 1-way communication.

Result 6 (Theorem 7 in Section 3) *The randomized communication complexity of $(1+\epsilon)$-SIJ is at least $\Omega(n/\epsilon^{2/3})$.*

A main question left open by this work is to close this gap for two-way approximate set-intersect join.

Finally, we consider a few other joins. In the Set-Equality Join (EQJ) problem, Alice and Bob wish to compute all pairs (i,j) such that $A_i = B_j$. We notice that the complexity of this enumerative problem is the same as the counting version. In fact, we show that a lower bound for the counting version matches an upper bound for the enumerative version. We thus use EQJ for the two versions interchangeably. In its approximation version, which we call c-EQJ, Alice and Bob wish to compute the *number* of such pairs (i,j) within a multiplicative factor of c. Recent work on the communication complexity of set intersection implies fairly tight results on the communication complexity of these two problems:

Result 7 (Theorem 8, 9 and Corollary 2 in Section 3) *The randomized communication complexity of EQJ is at least $\Omega(m \operatorname{ilog}^{\Theta(r)} m + \log\log n)$ and is at most $O(m \operatorname{ilog}^{r} m + \log\log n)$, where $\operatorname{ilog}^{r} = \log\log\cdots\log m$ (with r logs) denotes the $\log$ function iterated r times. Analogous results hold for c-EQJ for any constant $c > 0$.*

Then we turn to the Set-Disjointness Join (SDJ) problem, where Alice and Bob wish to compute all pairs (i,j) such that $A_i \cap B_j = \varnothing$. Again, we notice that the enumerative version has the same complexity as the counting version of this problem. More precisely, the lower bound for the counting version is high enough to match the trivial enumerative upper bound that Alice simply ships all her data to Bob. We thus use SDJ for the two versions interchangeably. In this case, we can show there is no substantially better communication algorithm than the one in which one party sends all their data to the other.

Result 8 (Theorem 10 in Section 3) *The randomized communication complexity of SDJ is $\Omega(mn)$. Analogous results hold for the c-approximation version c-SDJ for any constant $c > 0$.*

We also look at the At-Least-T Join (ATJ) problem, where T is a non-negative integer known in advance and Alice and Bob wish to output (or count, again they have the same complexity) those (i,j) such that $|A_i \cap B_j| \geq T$. Via a simple reduction to SDJ, this requires $\Omega(mn)$ communication. More interestingly, we consider the c-ATJ problem in which the players would like to approximate the number of such pairs up to a multiplicative factor of c. Here we can prove lower bounds:

Result 9 (Theorem 11, 12 in Section A) *For every $c > 1$, the randomized one-way communication complexity of c-ATJ is $\Omega(mn)$, and assuming $\kappa \log m \leq T \leq 99n/100$, for a constant $\kappa > 0$, the randomized two-way communication complexity of c-ATJ is $\Omega(m\sqrt{nT})$.*

2. BACKGROUND ON SKETCHES AND COMMUNICATION COMPLEXITY

In this section we introduce a few sketching algorithms from the data stream literature and some concepts in communication complexity that are needed in this paper. Please refer to the survey of Muthukrishnan [31] for more background on data steams and the book [22] for more knowledge on communication complexity. For background on approximation and randomized algorithms, please refer to the classic book [30].

Sketches. The idea being that a sketch provides a faithful but (much) smaller summary representation of a data object and that a reconstruction function has the ability to yield back a good approximation of the initial data object or some function (like the ℓ_0 norm) of this data object. Since sketches are smaller than the original data objects, they can be communicated with fewer bits. Furthermore, this communication preserves in a good way the original data object.

We will make use of a few linear sketches from the literature. Let κ be an integer. We say that a vector v is κ-sparse if v has at most κ non-zeros in its components.

Lemma 1 ([12], Count-Sketch) *For any positive integer n, $\delta \in (0,1)$, and integer $\kappa \in [n]$, there is a distribution on random matrices $S \in \mathbb{R}^{O(\kappa \cdot \log 1/\delta) \times n}$ and a reconstruction function $\texttt{Rec}(\cdot)$, such that*

1. *Given any κ-sparse vector $x \in \mathbb{R}^n$, $\texttt{Rec}(\cdot)$ can take Sx, and give x exactly with probability $1-\delta$. In addition, $\texttt{Rec}(\cdot)$ can take Sx and a parameter i, and output the i-th bit of x in $\tilde{O}(1)$ time.*
2. *The column sparsity of S is bounded by $O(\log n)$.*

Lemma 2 ([20], ℓ_0-sketch) *For any positive integer n, $\epsilon \in (0,1)$, and $\delta \in (0,1)$, there is a distribution on random matrices $M \in \mathbb{R}^{O(1/\epsilon^2 \cdot \log(1/\delta)) \times n}$ and a reconstruction function $\texttt{Rec}(\cdot)$, such that given any $x \in [N]^n$ (N is the maximum value of coordinates in x), $\texttt{Rec}(Mx)$ gives a $(1+\epsilon)$-approximation of $\mathbf{nnz}(x)$ with probability $1-\delta$. Furthermore, we can truncate the real number in each entry of M to $\log(Nn)$ bits without affecting the approximation ratio.*

Communication Complexity. In this paper we consider the classical model of two-party communication complexity. Here we briefly recall some standard notions. Let $\mathcal{X}$ and $\mathcal{Y}$ be sets of strings. One party, Alice, is given $x \in \mathcal{X}$ and another party Bob is given $y \in \mathcal{Y}$, and they want to jointly compute some function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, by exchanging messages according to a randomized algorithm Π. We allow Alice and Bob to use both private randomness (coins). Let r_A and r_B be the private coins used by Alice and Bob respectively. We use $\Pi_{xyr_Ar_B}$ to denote the transcript (i.e., the concatenation of messages) when Alice and Bob run Π on the input (x,y) using private coins r_A, r_B respectively, and $\Pi(x,y,r_A,r_B)$ denotes the output of the algorithm. We will omit x,y,r_A,r_B when clear from context. We say Π is a δ-error algorithm if for all $(x,y) \in \mathcal{X} \times \mathcal{Y}$,

$$\mathbf{Pr}_{r_A,r_B}[\Pi(x,y,r_A,r_B) \neq f(x,y)] \leq \delta.$$

Let $|\Pi|$ be the bit-length of the transcript. The *communication cost* of Π is $\max_{x,y,r_A,r_B} |\Pi_{xyr_Ar_B}|$. The δ-error randomized communication complexity of f, denoted by $R_\delta(f)$, is the minimal cost of any δ-error algorithm for f.

Let μ be a distribution over the inputs, and let $(X, Y) \sim \mu$. A deterministic algorithm Π computes f with error probability δ on μ if $\mathbf{Pr}_{(X,Y)\sim\mu}[\Pi(X, Y) \neq f(X, Y)] \leq \delta$. The *$\delta$-error μ-distributional communication complexity of* f, denoted by $D^{\mu}_{\delta}(f)$, is the minimum communication complexity of a deterministic algorithm that computes f with error probability δ on μ. Yao's Lemma [47] says that for any function f and any $\delta > 0$, $R_{\delta}(f) \geq \max_{\mu} D^{\mu}_{\delta}(f)$. Therefore to prove randomized communication lower bounds, one can choose an input distribution μ, then prove distributional communication lower bounds under μ.

For a problem $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, by a standard Chernoff bound argument, it holds that for any constant $\delta > 0$, $R_{1/3}(f) = \Omega(R_{\delta}(f)/\log(1/\delta)) = \Omega(R_{\delta}(f))$.

We use $R^{\rightarrow}_{\delta}(f)$ to denote *one-way* communication complexity where communication is either only from Alice to Bob or from Bob to Alice, and use $R^{(r)}_{\delta}(f)$ to denote *two-way* communication complexity using r rounds of communication. In a round of communication, only one player speaks to the other player, so Alice sends a single message to Bob or Bob sends a single message to Alice.

In our lower bound reductions, we allow parties to use potentially infinite amounts of public coins in algorithms (that is, Alice and Bob have free access to a shared, potentially infinite, public random string). Denote the corresponding randomised communication complexity by R^{pub}_{δ} where δ is the error parameter as before. Such coins can be "removed" (replaced with private randomness) by Newman's theorem. The increases on the communication cost and error probability are negligible in our applications.

Theorem 1 (Newman's Theorem [32]) *Let $f : \{0,1\}^t \times \{0,1\}^t \to \mathbb{N}$ be a function. For every $\delta > 0$ and every $\epsilon > 0$, $R_{\epsilon+\delta}(f) \leq R^{pub}_{\epsilon}(f) + O(\log t + \log \delta^{-1})$, where R^{pub} denotes the randomized communication complexity with public coins.*

We will need two well-known problems in communication complexity for proving our lower bounds by reductions.

Indexing. In the Indexing communication problem for $t \in \mathbb{N}$, Alice holds $x = \{0,1\}^t$, and Bob holds an index $i \in [t]$. The goal is for Bob to compute x_i. It is known that this problem effectively requires Alice to send her entire input to Bob, in the one-way communication setting:

Lemma 3 (see, e.g., [22]) $R^{\rightarrow}_{1/2-\delta}(\textit{Indexing}) \geq \Omega(t)$ *for any constant $\delta > 0$.*

Set-intersection (SI). In the SI communication problem for $t \in \mathbb{N}$, Alice has $x = (x_1, \ldots, x_t) \in \{0,1\}^t$, and Bob has $y = (y_1, \ldots, y_t) \in \{0,1\}^t$. They want to compute

$$\text{SI}(x, y) = \begin{cases} 1, & \text{if } \exists \ell \in [t] \text{ s.t. } x_\ell = y_\ell = 1, \\ 0, & \text{otherwise.} \end{cases}$$

We will use the following hard input distribution for SI:

Distribution ν (Razborov [39]). With probability $1/2$, x and y are random among all pairs of non-intersecting strings each of Hamming weight exactly $t/4$, while with probability $1/2$, x and y intersect in a unique uniformly randomly chosen i, and each x and y have Hamming weight exactly $t/4$.

Lemma 4 ([39]) $D^{\nu}_{\delta}(SI) \geq \Omega(t)$ *for a sufficiently small constant δ.*

3. COMMUNICATION COMPLEXITY OF SET JOINS

3.1 Exact Computation of Set-Intersection Join

As mentioned earlier, we can think of Alice having a matrix $P \in \{0,1\}^{m\times n}$ where $P_{i,\ell} = 1$ if A_i has item ℓ, and Bob having a matrix $Q \in \{0,1\}^{n\times m}$ where $Q_{\ell,j} = 1$ if B_j has item ℓ. Then, the set-intersection join (SIJ) is simply the Boolean matrix multiplication $R = P \cdot Q$. Let $k = \mathbf{nnz}(R) = |\{(i,j) \mid A_i \cap B_j \neq \emptyset\}|$ be the size of the output of an SIJ instance.

One-way Communication. First we show an $\Omega(mn)$ bound for SIJ when $m = O(n)$, matching the trivial algorithm where Alice sends all her data to Bob.

Theorem 2 $R^{\rightarrow}_{1/3}(SIJ) = \Omega(mn)$ *for $m = O(n)$.*

PROOF. The proof is by a reduction from the Indexing problem of size mn. We can view Alice's mn-bit input vector as an $m \times n$ Boolean matrix by partitioning the vector to chunks of size n as rows, and Bob's index $\ell \in [mn]$ as an (i,j)-pair where $i = \lfloor \ell/n \rfloor$ and $j = \ell - \lfloor \ell/n \rfloor \cdot n$.

Suppose first that $m < n/2$. We make the first m columns of Alice's matrix P be the $m \times m$ identity matrix. On the next $n/2$ columns, we place $\min\{ms, nk\}$ random $0/1$ entries in a block diagonal matrix with blocks of size $k \times s$ (recall that s is the tuple sparsity and k is the join output size). The rest $n/2 - m$ columns are just left to be all 0. Then Bob can make use of an algorithm for SIJ to do the following:

- Query $\mathbf{nnz}(P_{*,j})$ by creating an $n \times m$ matrix Q in which all but the first column is 0. In the first column, put a 1 in the j-th position.
- Query $\mathbf{nnz}(P_{*,i} + P_{*,j})$ by creating an $n \times m$ matrix Q in which all but the first column is 0. In the first column, put a 1 in the i-th position and the j-th position.

Thus Bob can compute the (i,j)-th entry of matrix P for any $i \in [m]$ and $j \in \{m+1, \ldots, n\}$ by comparing $\mathbf{nnz}(P_{*,j})$ and $\mathbf{nnz}(P_{*,i} + P_{*,j})$. By reducing from the Indexing problem we obtain an $\Omega(mn)$ lower bound. □

Remark 1 The above lower bound does not hold in the case when $m \gg n$: for instance, if $n, s = O(1)$ but $k = \Omega(m)$. (This could happen if P has $O(1)$ random non-zero columns.) In this case there is an upper bound of $2^n = O(1)$: one can just prepare an answer for each possible query set B_j. The above lower bound would be $\min\{ms, nt\} = \Omega(m)$, which cannot hold.

Two-way Communication. In the following, suppose $\mathcal{A}$ and $\mathcal{B}$ have *sparsity* s, i.e., there are at most s non-zeroes in each A_i, B_j. (Note that $s \leq n$.) In contrast to the one-way setting, the case of two-way communication becomes extremely interesting. We show that there is a two-round algorithm with communication cost $\tilde{O}(\min(ms, \sqrt{skn}) + n)$.

Theorem 3 $R^{(2)}_{2/m}(SIJ) = \tilde{O}(\min(ms, \sqrt{skn}) + n)$. *This bound also holds if we require to output for each $(i,j) \in [m]^2$, all the witnesses ℓ such that $\ell \in A_i \cap B_j$.*

Algorithm 1: Exact Algorithm for Set-Intersection Join

Input : Alice has a matrix $P \in \mathbb{N}^{m\times n}$ with rows representing $A_1, \ldots, A_m$, and Bob has a matrix $Q \in \mathbb{N}^{n\times m}$ with columns representing $B_1, \ldots, B_m$. Let $R \leftarrow PQ \in \mathbb{N}^{m\times m}$.

Output: $k \leftarrow \mathbf{nnz}(R)$

1 Set $s \leftarrow \max\{\mathbf{nnz}(A_i), \mathbf{nnz}(B_i) \mid i \in [m]\}$. Set $k_A \leftarrow 0, k_B \leftarrow 0$;
2 Alice and Bob approximate the number of non-zero entries in columns $R_{*,1}, \ldots, R_{*,m}$ up to a factor of 2 using Algorithm 3 (setting $\epsilon = 1$). Let $\tilde{k}_j$ be the 2-approximation of $\mathbf{nnz}(R_{*,j})$, and let $\tilde{k} \leftarrow \sum_{j\in[m]} \tilde{k}_j$;
3 **if** $s\tilde{k} < n$ **then**
4 Set $L = \emptyset$;
5 Let $H \leftarrow \{j \mid \tilde{k}_j > 0\}$;
6 **foreach** $j \in H$ **do**
7 Bob sends $Q_{*,j}$ to Alice;
8 Alice computes $\mathbf{nnz}(R) = \sum_{j\in H} \mathbf{nnz}(PQ_{*,j})$.
9 **else if** $s\tilde{k} \geq n$ **then**
10 Let $H \leftarrow \{j \mid \tilde{k}_j > \sqrt{s\tilde{k}/n}\}$, and $L \leftarrow [m]\backslash H$;
11 **foreach** $j \in H$ **do**
12 Bob sends $Q_{*,j}$ to Alice;
13 Alice computes $R_{*,j} = P \cdot Q_{*,j}$, and sets $k_A \leftarrow k_A + \mathbf{nnz}(R_{*,j})$;
14 Alice samples a random matrix $S \in \mathbb{R}^{O(\sqrt{s\tilde{k}/n}\cdot \log m)\times m}$ according to Lemma 1 (setting $\delta \leftarrow 1/(nm^3)$), and sends $Z \leftarrow SP$ to Bob;
15 Bob computes $k_B \leftarrow \sum_{j\in L} \mathbf{nnz}(\mathtt{Rec}(ZQ_{*,j}))$ according to Lemma 1, and sends it back to Alice;
16 Alice outputs $k \leftarrow k_A + k_B$.

We have the following immediate corollary for natural join. Let us first recall the relationship between natural join and set-intersection join. In the natural join, Alice has a relation $\mathcal{A} \subseteq [m] \times [n]$, and Bob has a relation $\mathcal{B} \subseteq [n] \times [m]$. They want to compute $\mathcal{A} \bowtie \mathcal{B}$. Now we make the following "rewrite" of the input: For each $i, j \in [m]$, let $A_i = \{\ell \mid (i, \ell) \in \mathcal{A}\}$ and $B_j = \{\ell \mid (\ell, j) \in \mathcal{B}\}$ be the projection sets of $\mathcal{A}, \mathcal{B}$ on $[m]$. It is straightforward to observe that $(\mathcal{A}' = \{A_1, \ldots, A_m\}, \mathcal{B}' = \{B_1, \ldots, B_m\})$ becomes an input to the set-intersection join. Still let $s = \max\{|A_i|, |B_j| \mid i, j \in [m]\}$ be the sparsity of sets.

Corollary 1 *Under the notations above, Alice and Bob can compute the natural join $\mathcal{A} \bowtie \mathcal{B}$ exactly in 2 rounds with probability $(1-2/m)$ using $\tilde{O}(\min(ms, \sqrt{skn}) + n)$ bits of communication.*

PROOF. (of Theorem 3) The algorithm is described in Algorithm 1, where $\tilde{k}_j, \tilde{k}$ are defined, and P, Q are the matrix representations of $\mathcal{A}, \mathcal{B}$. Now we give the analysis.

First note that Line 2 only costs $\tilde{O}(n)$ bits of communication by Theorem 5 (see Section 3.2; note that here $\epsilon = 1$).

Suppose now that $s\tilde{k} < n$, or $sk < n$ (note that $k \leq \tilde{k} \leq 2k$). The number of columns of R containing a non-zero entry is at most k. These columns can be determined with $\tilde{O}(n)$ communication using Lemma 2, since a column contains a non-zero entry if and only if $\tilde{k}_j > 0$. Given the identities of these columns j in R, Bob can send the corresponding $Q_{*,j}$, which requires only s communication since columns of Q are s-sparse. Hence, Bob can send all such columns of Q using at most $k \cdot s \cdot \log n = \tilde{O}(n)$ communication. Hence, if $sk < n$, the upper bound is $\tilde{O}(n)$, as desired.

Suppose then that $s\tilde{k} \geq n$. We can show an $O(\min(ms, \sqrt{skn}))$ upper bound. Since each $\tilde{k}_j$ ($j \in [m]$) is a 2-approximation of k_j, the number of columns in $H = \{j \mid \tilde{k}_j > \sqrt{sk/n}\}$ can be at most $2k/(\sqrt{sk/n}) = 2\sqrt{kn/s}$. For each $j \in H$, if Bob sends $Q_{*,j}$ to Alice, then Alice can compute $R_{*,j} = PQ_{*,j}$. The total communication for all such columns, given that the columns of Q are s-sparse, is at most $2\sqrt{kn/s} \cdot s = 2\sqrt{skn}$. All remaining columns of R have at most $\sqrt{sk/n}$ non-zero entries. Alice chooses a random Count-Sketch matrix S with $\kappa = \sqrt{s\tilde{k}/n}$ according to Lemma 1, computes SP, and sends this to Bob. This requires $\tilde{O}(\sqrt{sk/n} \cdot n) = \tilde{O}(\sqrt{skn})$ bits of communication.

Finally, note that if $ms < \sqrt{skn}$, Alice can alternatively just send P to Bob using $O(ms \log n)$ communication, since the rows of P are s-sparse.

The error of Algorithm 1 comes from two places. The first is the error introduced by the call of Algorithm 3 at Line 2, which is bounded by $1/m$ by Theorem 5. The second error comes from Line 14 and 15. By setting $\delta = 1/(nm^3)$ in Lemma 1 and apply a union bound on columns with indices in L, we can also bound this error by $1/m$.

For the communication rounds, Lines 2 and 14 can be done simultaneously in the first round (from Alice to Bob), and Lines 7 (or 12) and 15 can be done simultaneously in the second round (from Bob to Alice).

Algorithm 2: Post-processing to Compute All Pairs (i, j), All Witnesses ℓ s.t. $\ell \in A_i \cap B_j$

/* All notations follow Algorithm 1 */
1 **foreach** $j \in H$ **do**
2 Alice computes for all $i \in [m]$, the $A_i \cap B_j$;
3 **foreach** $j \in L$ **do**
4 **foreach** $i \in [m]$ **do**
5 **foreach** $\ell \in B_j$ **do**
6 Bob removes ℓ from B_j, getting $Q'_{*,j}$, computes $R'_{*,j} = \mathtt{Rec}(ZQ'_{*,j})$; /* Z is sent to Bob at Line 14 of Algorithm 1 */
7 **if** $R_{i,j} \neq R'_{*,j}$ **then**
8 declares $\ell \in A_i \cap B_j$;
9 Bob adds ℓ back to B_j.

Algorithm 2 shows that we can in fact compute for all pairs $(i, j) \in [m]^2$, all the witnesses ℓ such that $\ell \in A_i \cap B_j$ using a post-processing (without any further communication) after running Algorithm 1. For any $j \in H$, Alice can compute the witnesses ℓ for all (i, j) ($i \in [m]$) trivially since Bob has sent the whole column $Q_{*,j}$ (i.e., the set B_j) to her. For any $j \in L$, for any $i \in [m]$ Bob can recover all the witnesses ℓ in $A_i \cap B_j$ as follows: for each $\ell \in B_j$, Bob first removes ℓ from B_j by setting the corresponding cell $Q_{\ell,j} = 0$ (getting $Q'_{*,j}$), and compute $R'_{*,j} = \mathtt{Rec}(ZQ'_{*,j})$.

Now for each $i \in [m]$, if $R_{i,j} \neq R'_{i,j}$, then we conclude that $\ell \in A_i \cap B_j$. Note that we perform at most nm^2 such tests, thus by a union bound all tests succeed with probability $1 - 1/(nm^3) \cdot (nm^2) = 1 - 1/m$.

Note that at the end of Algorithm 2, the sets $A_i \cap B_j$ $(i, j \in [m])$ are disjointly distributed at Alice and Bob. They need to spend another $\tilde{O}(k)$ bits of communication in the worst case if Alice (or Bob) would like to obtain all the witnesses. □

Remark 2 (Not-Superset Join and Not-Subset Join) As mentioned in the introduction, the not-superset join (outputting all (i, j) such that $A_i \not\supseteq B_j$) can be thought as the complement of the set-intersection join: one can convert $\mathcal{A} = \{A_1, \ldots, A_m\}$ to $\mathcal{A}' = \{A'_1, \ldots, A'_m\}$ such that $A'_i = [n] - A_i$ for each $i \in [m]$, and then solve the set-intersection join on $\mathcal{A}'$ and $\mathcal{B}$. There is one issue: sets in $\mathcal{A}'$ may not have sparsity s (typically $\ll n$). However, we have noticed that our upper bound in Theorem 3 still holds since Algorithm 1 only requires sets in $\mathcal{B}$ to have sparsity s (or only sets in $\mathcal{A}$ by exchanging the positions of Alice and Bob). The above argument also implies to not-subset join by symmetry.

Surprising as the above algorithm is, there is a matching lower bound up to polylogarithmic factors:

Theorem 4 $R_{1/3}(SIJ) = \Omega(\min(ms, \sqrt{skn}) + n)$.

PROOF. For the $\Omega(n)$ lower bound, let $s, k \leq n$ and put an instance of n-bit SI on the diagonal of P and Q. Here we use $m \leq n$. If $m > n$ we only use the upper $n \times n$ submatrix.

It remains to show a lower bound of $\min(ms, \sqrt{skn})$. We can assume $sk > n$, as otherwise this minimum is less than the n lower bound just established. Notice that if $ms = \sqrt{skn}$, then $m = \sqrt{kn/s}$. It therefore suffices to assume that $\sqrt{kn/s} < m$ and show a $\sqrt{skn}$ lower bound. Indeed, if $m < \sqrt{kn/s}$, then we can both reduce k and n by the same multiplicative factor g, obtaining k' and n', so that $sk' > n'$ still holds and g is chosen large enough so that $m = \sqrt{k'n'/s}$. Reducing n to n' corresponds to restricting to the first n' columns of P and first n' rows of Q. As s is unchanged, and when $m = \sqrt{k'n'/s}$ we have $ms = \sqrt{sk'n'}$, by showing a $\sqrt{sk'n'}$ lower bound we obtain the desired ms lower bound.

Hence, it suffices to show a $\sqrt{skn}$ lower bound under the assumption that $\sqrt{kn/s} < m$. Set $t = \sqrt{nk/s}$. We construct P and Q as follows: In P only the first t rows are non-zero and in Q only the first t columns are non-zero. We partition coordinates $[n]$ into n/s disjoint groups each of size s. We place the first SI instance on the first row of P and first column of Q into the first group of s coordinates. We place the second independent SI instance on the second row of P and second column of Q into the second group of s coordinates. After processing n/s independent SI instances, we wrap back around to the first group of s coordinates, and second group of s coordinates, and continue. Note in total we plant t independent SI instances. Doing things in this way ensures that the inner product of the i-th row of P with the j-th column of Q is only non-zero if they both have SI instances placed on the same group of s coordinates. For each i, only an s/n fraction of the j have this property. As there are t different choices of i and j, this makes the number of non-zero entries of R equal to $t^2 \cdot s/n$, which is at most k by our choice of t.

If we can exactly determine $\mathbf{nnz}(R)$, then we can figure out the number of the t planted SI instances which evaluated to 0, since all cross-pairs (i, j) $(i \neq j)$ of SI instances will intersect with probability $1 - o(1)$ (similar to Claim 1 in Section 3.3). We will know from the proof of Theorem 10 that just determining the OR of t SI instances has complexity the same as solving a single SI instance of size $t \cdot s$. Therefore we obtain a lower bound of $\Omega(ts) = \Omega(\sqrt{skn})$. □

Algorithm 3: $(1 + \epsilon)$-Approximation Algorithm for Set-Intersection Join

Input : Alice has $P \in \mathbb{N}^{m \times n}$ with rows representing $A_1, \ldots, A_m$, and Bob has $Q \in \mathbb{N}^{n \times m}$ with columns representing $B_1, \ldots, B_m$.

Output: A $(1 + \epsilon)$-approximation of $|\{(i, j) \mid A_i \cap B_j \neq \emptyset\}|$.

1 $\delta \leftarrow 1/m^2$;

2 Alice samples a random sketching matrix $M \in \mathbb{R}^{O(1/\epsilon^2 \cdot \log(1/\delta)) \times m}$ according to Lemma 2, and sends $Z \leftarrow MP$ to Bob;

3 Bob outputs $\sum_{j \in [m]} \texttt{Rec}(ZQ_{*,j})$.

3.2 Approximate Counting for Set-Intersection Join

One-way Communication. We can use the ℓ_0-sketch (Lemma 2) to design a simple one-way algorithm for approximating SIJ to within $1 + \epsilon$, for every $\epsilon > 0$:

Theorem 5 *For all* $\epsilon > 0$, $R^{\rightarrow}_{1/m}((1 + \epsilon)\text{-}SIJ) = \tilde{O}(n/\epsilon^2)$.

PROOF. The algorithm is described in Algorithm 3. Now we give the analysis.

First, note that $\texttt{Rec}(ZQ_{*,j}) = \texttt{Rec}(MPQ_{*,j}) = \texttt{Rec}(MR_{*,j})$. By Lemma 2, $\texttt{Rec}(MR_{*,j})$ computes $\mathbf{nnz}(R_{*,j})$ up to a multiplicative error $(1 + \epsilon)$ with probability $(1 - 1/m^2)$ for each $j \in [m]$. Therefore Algorithm 3 computes $\mathbf{nnz}(R)$ up to $(1 + \epsilon)$-approximation with probability $(1 - 1/m)$ by a union bound. Finally, note that Alice only needs to send the matrix Z, which is $O(n/\epsilon^2 \cdot \log m \log(mn))$ bits by Lemma 2 (here we set $N = O(m)$). □

We can prove a matching lower bound for this approximation problem (up to polylog factors):

Theorem 6 $R^{\rightarrow}_{1/3}((1 + \epsilon)\text{-}SIJ) = \Omega(n/\epsilon^2)$.

We need the following lemma of Jayram et al. [19]. Let $\Delta(a, b)$ be the hamming distance between two bitstrings $a, b \in \{0, 1\}^{1/\epsilon^2}$.

Lemma 5 ([19]) *Let x be a random bitstring of length $\gamma = 1/\epsilon^2$, and let i be a random index in $[\gamma]$. Choose γ public random bitstrings $r^1, \ldots, r^\gamma$, each of length γ. Create γ-length bitstrings a, b as follows:*

- *For each $j \in [\gamma]$, a_j = majority$\{r^j_k \mid$ indices k for which $x_k = 1\}$.*
- *For each $j \in [\gamma]$, $b_j = r^j_i$.*

Then with success probability $1/2+\delta$ for a constant $\delta > 0$, we can determine the value of x_i from any $c\sqrt{\Delta(a,b)}$-additive approximation to $\Delta(a,b)$, provided $c > 0$ is a sufficiently small constant.

PROOF. We give a reduction from Indexing. Alice has a random bitstring y of length $(n-\gamma)\cdot\gamma$, which is partitioned to $n-\gamma$ contiguous groups $y^1,\ldots,y^{n-\gamma}$. She creates her matrix $P \in \{0,1\}^{\gamma\times n}$ as follows: she uses public coins to choose random bitstrings $r^1,\ldots,r^\gamma$, each of length γ. For the leftmost $\gamma\times\gamma$ submatrix of P, in the i-th column for each $i\in[\gamma]$, Alice uses $r^1,\ldots,r^\gamma$ and the value i to create the γ-length bitstring b according to Lemma 5 and assigns it to this column. Next, in the remaining $n-\gamma$ columns of P, in the j-th column for each $j\in\{\gamma+1,\ldots,n\}$, Alice uses $y^{j-\gamma}$ and $r^1,\ldots,r^\gamma$ to create the γ-length bitstring a according to Lemma 5 and assigns it to this column.

Bob has an index $\ell = i+(j-\gamma-1)\cdot\gamma$ ($i\in[\gamma], j\in\{\gamma+1,\ldots,n\}$), and he wants to learn the ℓ-th coordinate of y. This is a standard Indexing problem, and by Lemma 3 Alice has to send $\Omega(|y|)=\Omega(n/\epsilon^2)$ bits to get a success probability $(1/2+\delta)$ for any constant $\delta>0$. For the reduction, Bob creates his matrix $Q\in\{0,1\}^{n\times\gamma}$ in which all but the first column is 0. In the first column, he puts a 1 in the i-th position and the j-th position. Then in $P\cdot Q$ all but the first column is 0, and the first column is equal to $P_{*,i}+P_{*,j}$. Note that $(P_{*,i},P_{*,j})$ corresponds to a pair of (a,b) created from $r^1,\ldots,r^\gamma$ and $x=y^{j-\gamma}$ (and $x_i=(y^{j-\gamma})_i=y_\ell$) in Lemma 5. Also note that $(P_{*,i}+P_{*,j})_t=1$ if and only if $P_{t,i}\neq P_{t,j}$. Therefore from a $(1+c_\epsilon\epsilon)$-approximation of $\mathbf{nnz}(PQ)=\mathbf{nnz}(P_{*,i}+P_{*,j})$ for a sufficiently small constant c_ϵ, and exact values $\mathbf{nnz}(P_{*,i}),\mathbf{nnz}(P_{*,j})$ (Alice can send Bob all $\mathbf{nnz}(P_{*,k}\ (k\in[n]))$ using $O(n\log(1/\epsilon))=o(n/\epsilon^2)$ bits which is negligible in the reduction), one can compute $\Delta(P_{*,i},P_{*,j})$ up to an additive error $c_\epsilon\epsilon\cdot\mathbf{nnz}(PQ)\le c\sqrt{\Delta(P_{*,i},P_{*,j})}$ (note that both $\mathbf{nnz}(PQ)$ and $\Delta(P_{*,i},P_{*,j})$ are in the order $\Theta(1/\epsilon^2)$) for a sufficiently small constant c_ϵ, and consequently y_ℓ with probability $1/2+\delta$ for a constant δ by Lemma 5. This completes the reduction. The one-way communication complexity of $(1+\epsilon)$-SIJ follows from Lemma 3. □

Two-way Communication. While we do not have a better upper bound than given by Theorem 5, we can still prove a non-trivial lower bound using the recent work [45].

Theorem 7 *$R_\delta((1+\epsilon)$-SIJ$)=\Omega(n/\epsilon^{2/3})$ for a sufficiently small constant δ.*

PROOF. We start by choosing a hard input distribution ψ for SIJ: for each $i\in[m]$, we first choose $(A_i,B_i)\sim\mu$ (see its definition in Section 2), and then pick a special coordinate $\ell\in[n]$ uniformly at random and replace (A_i^ℓ,B_i^ℓ) with $\{0,1\}^2$ uniformly at random.

Let $\mathrm{SUM}(\mathcal{A},\mathcal{B})=\sum_{i\in[m]}\mathrm{SI}(A_i,B_i)$. In [45] the following is shown.

Lemma 6 ([45]) *Any randomized algorithm that computes SUM up to an additive error $\sqrt{m}/2$ with probability δ for a sufficiently small constant δ needs $\Omega(mn)$ bits of communication.*

We now establish a relationship between SIJ and SUM: For a fixed pair $(i,j), i\neq j$, the probability that $A_i\cap B_j=\emptyset$ is at most $(1-1/16)^{n-1}\le 2^{-\Omega(n)}$ (ignoring the special coordinate ℓ). By a union bound, with error probability at most $m^2\cdot 2^{-\Omega(n)}=o(1)$ (assuming $n=\Omega(\log m)$), we have $\mathrm{SIJ}(\mathcal{A},\mathcal{B})=\mathrm{SUM}(\mathcal{A},\mathcal{B})+m(m-1)$. By this equality and Lemma 6, we conclude that any randomized algorithm that computes SIJ up to an additive error $\sqrt{m}/2$ with probability δ' for a sufficiently small constant δ' needs $\Omega(mn)$ bits of communication. Since $\mathrm{SIJ}(A,B)\le m^2$, the theorem follows by setting $m=\epsilon^{-2/3}$. □

3.3 Other Types of Joins

Set-Equality Join. Consider the EQJ problem where Alice has sets $A_1,\ldots,A_m$, and Bob has sets $B_1,\ldots,B_m$. They want to compute all pairs (i,j) such that $A_i=B_j$ (or count the number of such pairs, which has the same complexity). This is just the problem of computing the intersection of sets $\mathcal{A}=\{A_1,\ldots,A_m\}$ and $\mathcal{B}=\{B_1,\ldots,B_m\}$, where the elements of sets $\mathcal{A}$ and $\mathcal{B}$ are themselves sets. The communication complexity of set intersection is known, with near-tight upper and lower bounds in terms of the number of rounds. We can assume that $\mathcal{A}$ and $\mathcal{B}$ are sets, rather than multisets, and the players can each locally solve the EQJ problem by also exchanging the multiplicities of each of their sets A_i and B_j, which can be done by each communicating $O(m)$ bits (since the sum of the multiplicities is m).

Theorem 8 *([8]) $R^{(r)}_{1/3}(EQJ)=O(m\log^{r}m+\log\log n)$, where $\log^{r}=\log\log\cdots\log m$ (with r logs). denotes the $\log$ function iterated r times.*

The upper bound is almost matched by the following lower bound.

Theorem 9 *([40]) $R^{(r)}_{1/3}(EQJ)=\Omega(m\log^{cr}m+\log\log n)$, where $c>0$ is a fixed constant.*

Let c-EQJ denote the problem of approximating the number of pairs (i,j) such that $A_i=B_j$ up to a multiplicative factor of c. We notice that the above lower bound for EQJ in fact holds just to check if $\mathcal{A}\cap\mathcal{B}=\emptyset$, and therefore implies:

Corollary 2 *$R^{(r)}_{1/3}(c\text{-}EQJ)=\Omega(m\log^{cr}m+\log\log n)$ for a fixed constant $c>0$.*

Set-Disjoint Join, Subset Join and Superset Join. In the Set-Disjoint-Join (SDJ) problem, where Alice and Bob wish to compute all pairs (i,j) such that $A_i\cap B_j=\varnothing$. We show that for SDJ there is no better algorithm than the trivial one in which Alice just sends all her data to Bob. The lower bound in fact holds for the case where Alice and Bob just want to count the number of such pairs (i,j), or even whether there exists a pair (i,j) such that $A_i\cap B_j=\varnothing$, and thus any constant multiplicative approximation of counting the number of such pairs has the same complexity.

Theorem 10 *$R_{1/3}(SDJ)=\Omega(mn)$. The lower bound also holds for any multiplicative approximation of SDJ.*

Due to the equivalence of SDJ and subset join (SubJ) and superset join (SupJ), (e.g., $A_i\supseteq B_j\Leftrightarrow\overline{A_i}\cap B_j=\emptyset$), we have the following corollary.

Corollary 3 *$R_{1/3}(SubJ) = R_{1/3}(SupJ) = \Omega(mn)$. The lower bound also holds for any multiplicative approximations of SubJ and SupJ.*

PROOF. Due to the space constraints, we delay the proof to Appendix B.1. □

At-Least-T Join. We will show the following two results for At-Least-T Join (ATJ) and its c-approximation counting version c-ATJ. We comment that Algorithm 1 and 2 for SIJ can also be used to solve ATJ since they can output (thus count the number of) all the witnesses of each intersecting pair (A_i, B_j) $(i, j \in [m])$.

Theorem 11 *For $T \geq \log m$, $R^{\rightarrow}_{1/3}(c\text{-}ATJ) = \Omega(mn)$.*

Theorem 12 *There is a constant $\kappa > 0$ so that for any $\kappa \log m \leq T \leq 99n/100$ it holds that $R_{1/3}(c\text{-}ATJ) = \Omega(m\sqrt{nT})$.*

Due to the space constraints, we delay this section to Appendix A.

4. IMPROVED ALGORITHM FOR OUTPUT-SENSITIVE MATRIX MULTIPLICATION, AND TRANSITIVE-CLOSURE

In this section we show that the algorithm for computing the non-zero entries of $P \cdot Q$ in Section 3 can be further improved to algorithmically compute the transitive closure for sparse graphs very efficiently, when the output of the transitive closure (i.e., the number of edges in the transitive closure) is sparse.

Output-sensitive matrix multiplication algorithms have been studied in several prior works. The standard column-row approach to matrix multiplication takes time $O(kn + n^2)$ [41] where n is the dimension of the matrices, and k is the number of nonzeroes in the output. Pagh [34] improved this to $\tilde{O}(\mathbf{nnz}(P) + \mathbf{nnz}(Q) + kn)$ time over the real numbers with a randomized algorithm. Williams and Yu [44] show that Alice and Bob (with separate matrices) can compute their product with an $\tilde{O}(kn)$ communication algorithm; their construction works over any field. Lingas [25] shows how to compute output-sensitive Boolean matrix multiplication in $\tilde{O}(n^2 \cdot k^{\omega/2-1}) \leq O(n^2 \cdot k^{0.19})$ time, which is useful when the input matrices and output matrix are dense.

Theorem 13 *There is a randomized algorithm for the Boolean matrix multiplication of $P \in \{0,1\}^{m \times n}$ and $Q \in \{0,1\}^{n \times m}$ that runs in $\tilde{O}(k + k^{1/2}(\mathbf{nnz}(P) + \mathbf{nnz}(Q)))$ time and succeeds with probability $1 - 1/n$, where $k = \mathbf{nnz}(PQ)$, that is, the number of nonzero entries in the output.*[5]

PROOF. The algorithm is similar to the two-way algorithm for SIJ in Section 3. We first estimate the number of non-zero entries k_j in each column of $R_{*,j}$ up to a factor of 2; using the approximation algorithm for SIJ (Theorem 5), this can be done in $\tilde{O}(\mathbf{nnz}(P) + \mathbf{nnz}(Q) + k)$ time. Following the notations in Algorithm 1, let $\tilde{k}_j$ be the 2-approximation of k_j. Let $H = \{j \mid \tilde{k}_j > \sqrt{k}\}$ and $L = [n] \backslash H$. That is, we divide the columns of the product $R_{*,j}$ into two types: those which are "dense" (in the set H) and those which are "sparse" (in the set L). Note that $|H| = O(\sqrt{k})$, by a counting argument.

For those $j \in H$ (those column indices with a "dense" number of non-zeroes), we compute the non-zero entries in $R_{*,j}$ by directly multiplying $Q_{*,j}$ with each $P_{i,*}$ $(i \in [n])$, which can be done in $O(\mathbf{nnz}(P) + \mathbf{nnz}(Q))$ time. Since $|H| \leq O(\sqrt{k})$, the total running time of computing $\{R_{*,j} \mid j \in H\}$ is at most $O(\sqrt{k} \cdot (\mathbf{nnz}(P) + \mathbf{nnz}(Q))$.

For those $j \in L$ (with a "sparse" number of non-zeroes), we build a Count-Sketch matrix S with $\sqrt{k} \log^2 n$ rows; in particular, we set $\kappa = \sqrt{k} \log n$ and $\delta = 1/n^{10}$ in Lemma 1. Then we compute $Z = (S \cdot P) \cdot Q$. Next, for each $j \in L$ we try to reconstruct the non-zero entries in $R_{*,j}$ from $Z_{*,j}$ using Lemma 1 (note that $Z = S \cdot R$). It is easy to see that the running time of computing the non-zero entries in Z is bounded by $\tilde{O}(\sqrt{k}(\mathbf{nnz}(P) + \mathbf{nnz}(Q))$, by first multiplying $S \cdot P$ and then multiplying Q with the result. However, the recovery procedure $\texttt{Rec}(\cdot)$ for Count-Sketch in Lemma 1 is slow, a priori: the naive way to recover each x_i in the n-bit vector x has running time at least $\Omega(n)$. In the following, we show how to augment the sketching step to speed up the recovery using a dyadic interval trick which has been used before in several places (for example, [11]).

We first recall the definition of *dyadic intervals* of $[n]$. Assume that n is a factor of 2, otherwise we can always pad dummy items. The dyadic intervals are $L = \log n + 1$ partitions of $[n]$: $\mathcal{I}_0 = (1, 2, \ldots), \mathcal{I}_1 = (\{1, 2\}, \{3, 4\}, \ldots), \mathcal{I}_2 = (\{1, 2, 3, 4\}, \{5, 6, 7, 8\}, \ldots), \ldots, \mathcal{I}_{\log n} = (\{1, 2, \ldots, n\})$. We have the following lemma regarding Count-Sketch and dyadic intervals.

Lemma 7 *Let $x \in \mathbb{N}^n$ be a vector with at most κ non-zero entries. Let $\mathcal{I} = \{\mathcal{I}_0, \ldots, \mathcal{I}_{\log n}\}$ be dyadic intervals of $[n]$. Denote $x^{\mathcal{I}_0} = (x)$, $x^{\mathcal{I}_1} = (x_1 + x_2, x_3 + x_4, \ldots)$, $x^{\mathcal{I}_2} = (x_1 + x_2 + x_3 + x_4, \ldots)$ and so on. For $0 \leq \ell \leq \log n$, let $S_\ell \in \mathbb{R}^{\kappa \log(n^{10}) \times 2^\ell}$ be a Count-Sketch matrix. There exists an algorithm that can recover x from $S_0 \cdot x^{\mathcal{I}_0}, \ldots, S_L \cdot x^{\mathcal{I}_{\log n}}$ with probability $1 - 1/n^7$, using $\tilde{O}(\kappa)$ time.*

PROOF. The algorithm is as follows:

1. Consider a binary tree whose leaves are $\{1, 2, \ldots, n\}$, with each interval node corresponding to the interval formed by the leaves of its subtree. That is, the root corresponds to $x_1^{\mathcal{I}_0}$, the left child of the root corresponds to $x_1^{\mathcal{I}_1}$, the right child of the root corresponds to $x_2^{\mathcal{I}_1}$, and so on.

2. Recover $x_1^{\mathcal{I}_0}$ from $S_0 x^{\mathcal{I}_0}$; if $x_1^{\mathcal{I}_0} > 0$ then mark the root *red*.

3. Start from the root and proceed level by level top-down. For each red node in this level, for each of its two child nodes (say the child node is the i-th node at level ℓ), mark it red if $x_i^{\mathcal{I}_\ell} > 0$, which can be recovered from $S_L x^{\mathcal{I}_\ell}$.

4. All leaves marked red correspond to non-zero entries in x.

[5]The algorithm can be implemented in any reasonable computational model with random access to the input, such as the RAM model.

Note that the above algorithm essentially probes κ root-leaf paths in the tree each of length at most $O(\log n)$; and for each path, the time spent on each node in the path is $\tilde{O}(1)$ by Lemma 1. Also by Lemma 1, we can recover $x_i^{\mathcal{I}_\ell}$'s at each probed node with probability $1 - 1/n^{10}$ (recall that we have set $\delta = 1/n^{10}$). The overall success probability follows from a union bound on all probed nodes. □

In our setting, we view each $R_{*,j}$ $(j = 0, 1, \ldots, \log n)$ as a vector of sparsity at most $\ell = \sqrt{k}$. However we cannot create $(R_{*,j})^{\mathcal{I}_0}, \ldots, (R_{*,j})^{\mathcal{I}_{\log n}}$ directly and then apply Count-Sketch matrices $S_0, \ldots, S_L$, since $R_{*,j} = P \cdot Q_{*,j}$ is the vector we want to compute. We can instead sum up the corresponding rows of P according to dyadic intervals, obtaining $P_0, P_1, \ldots, P_{\log n}$ where P_i is a $2^i \times n$ matrix, and then apply $S_0, S_1 \ldots, S_{\log n}$ on P_i's. We can apply Lemma 7 to recover non-zero entries of each $R_{*,j}$ from the sketches $S_0 \cdot P_0 \cdot Q_{*,j}$, $S_1 \cdot P_1 \cdot Q_{*,j}$, …, $S_{\log n} \cdot P_{\log n} \cdot Q_{*,j}$. In this way we can recover all non-zero entries in R with success probability $1 - 1/n^{10} \cdot \tilde{O}(n^2) \geq 1 - 1/n^7$, by a union bound.

Now let us analyze the running time for this modified recovery procedure in our setting. First, it takes $O(\mathbf{nnz}(P))$ time to implicitly (that is, only specify non-zero entries) compute each P_i $(0 \leq i \leq \log n)$, since P is sparse. Second, applying S_i on each P_i takes $\tilde{O}(\ell \cdot \mathbf{nnz}(P)) = \tilde{O}(\sqrt{k} \cdot \mathbf{nnz}(P))$ time each. Third, for each $i = 0, 1, \ldots, \log n$, it takes $\tilde{O}(\sqrt{k} \cdot \mathbf{nnz}(Q))$ time to compute $S_i \cdot P_i \cdot Q$. To sum up, it takes $\tilde{O}(\sqrt{k}(\mathbf{nnz}(P) + \mathbf{nnz}(Q))$ time to compute $\{S_0 \cdot P_0 \cdot Q, S_1 \cdot P_1 \cdot Q, \ldots, S_{\log n} \cdot P_{\log n} \cdot Q\}$, from which we can recover each column of R.

Corollary 4 *There is a randomized algorithm for computing the transitive closure of an $n \times n$ Boolean matrix M which runs in $\tilde{O}(k^{1.5})$ time and succeeds with probability $1 - 1/n$, where k is the number of edges in the transitive closure.*

PROOF. Every power M^i must contain at most k nonzeros, including $M^1 = M$ itself. Therefore, if we compute M^n by repeated squaring for $\log n$ times using the algorithm of Theorem 13, we can ensure a running time of

$$\tilde{O}\left(k + k^{1/2} \cdot \left(\max_{i=1,\ldots,n} \mathbf{nnz}(M^i)\right)\right) \leq \tilde{O}(k^{1.5}).$$

□

As noted by Borassi, Crescenzi, and Habib [7], efficient algorithms for transitive closures have further applications themselves, such as checking whether a given graph is a *comparability graph*. Given any partial order P on n elements, the *comparability graph of* P is the undirected graph with n nodes (one for each element in P) and an edge between two nodes if and only if they are comparable in P. The class of comparability graphs is simply the class of all graphs obtainable from a partial order P in this manner. Recognizing whether a given input graph can be modelled by some partial order P in this way is called the *comparability graph recognition* problem, and is very old (see discussion in [7]). To our knowledge, the previous best known algorithms ran in $O(mn)$ time (m is the number of edges of the graph) by Golumbic in 1977 [15], and $O(n^\omega)$ time where $\omega \geq 2$ is the matrix multiplication exponent (cf. [27]).

Our algorithm implies a rather quick solution to this problem for sparse graphs, avoiding fast matrix multiply:

Corollary 5 *There is a randomized algorithm for comparability graph recognition running in $\tilde{O}(m^{1.5})$ time, where m is the number of edges in the given graph.*

5. CONCLUDING REMARKS

In this section we discuss how to extend our results on set-intersection join to multiparty, as well as a few problems left open by our work and some future research directions.

We can generalize set-intersection join to the *multiparty* setting. Let $O^1, \ldots, O^t$ be t parties. We still have two collections of sets $A = (A_1, ..., A_m)$ and $B = (B_1, ..., B_m)$, but they are distributed across t players. That is, each O^i has $A^i \subseteq A$ and $B^i \subseteq B$, where $\{A^1, \ldots, A^t\}$ is a partition of A and $\{B^1, \ldots, B^t\}$ is a partition of B. The t parties want to compute SIJ$(\mathcal{A}, \mathcal{B})$.

This generalization can be solved easily using the "linearity" of Algorithm 1. Let $P \in \{0,1\}^{m \times n}$ be the matrix representation of A, and let $P^i \in \{0,1\}^{m \times n}$ $(i \in [t])$ be the matrix representing the subset $A^i \subseteq A$ (pad all '0' in rows do not correspond A^i). Similarly, let $Q \in \{0,1\}^{n \times m}$ be the matrix representation of B, and let $Q^i \in \{0,1\}^{n \times m}$ $(i \in [t])$ be the matrix representing the subset $B^i \subseteq B$. In the first step, Player O^1 sends SP^1 to player O^2 (S is a Count-Sketch same as that in Algorithm 1), and then O^2 computes $S(P^1 + P^2)$ and sends it to O^3, and so on. At the end O^t can compute SP where $P = P^1 + \cdots + P^t$. In the second step, O^t sends SP to each party $O^1, \ldots, O^{t-1}$, and the t parties use the same way to compute $SP \cdot Q$. Next, the parties can use the same way to compute MPQ where M is ℓ_0-sketch. Given SPQ and MPQ, we run Algorithm 1 to recover the columns with small number of non-zeros, and for remaining columns have the parties who possess them directly transmit these. The total communication cost is at most t times that for the 2-party case (Theorem 3).

There are many problems left open by this work. The biggest technical question left open is to close the gap between the upper bound $\tilde{O}(n/\epsilon^2)$ and the lower bound $\Omega(n/\epsilon^{2/3})$ for $(1+\epsilon)$-SIJ in the 2-way communication model. Another natural question is whether we can extend this line of research to more expressive queries. Finally, it will be interesting to investigate whether this line of work could lead to asymptotically faster algorithms for Boolean matrix multiplication that do not rely on heavy un-implementable algebra (like the last 30 years of work on the subject). That would be a major advance for both theory and practice. There are other potential applications which rely on Boolean matrix multiplication as well (triangle detection/counting, context-free grammar parsing, etc.) which are worth pursuing further.

6. REFERENCES

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

[2] R. Angles and C. Gutiérrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1), 2008.

[3] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *VLDB*, pages 918–929, 2006.

[4] A. Badia and M. Dobbs. Supporting quantified queries in distributed databases. *IJPEDS*, 29(5):421–459, 2014.

[5] A. Badia, D. VanGucht, and M. Gyssens. Querying with generalized quantifiers. In *Applications of Logic Databases*, pages 235–258, 1993.

[6] S. Blanas, J. M. Patel, V. Ercegovac, J. Rao, E. J. Shekita, and Y. Tian. A comparison of join algorithms for log processing in mapreduce. In *SIGMOD*, pages 975–986, 2010.

[7] M. Borassi, P. Crescenzi, and M. Habib. Into the square - on the complexity of quadratic-time solvable problems. *CoRR*, 2014.

[8] J. Brody, A. Chakrabarti, R. Kondapally, D. P. Woodruff, and G. Yaroslavtsev. Beyond set disjointness: the communication complexity of finding the intersection. In *PODC*, pages 106–113, 2014.

[9] J. Claußen, A. Kemper, G. Moerkotte, and K. Peithner. Optimizing queries with universal quantification in object-oriented and object-relational databases. In *VLDB*, pages 286–295, 1997.

[10] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[11] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[12] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. In *SIROCCO*, pages 280–294, 2006.

[13] M. Dadashzadeh. An improved division operator for relational algebra. *Inf. Syst.*, 14(5):431–437, 1989.

[14] G. H. L. Fletcher, M. Gyssens, D. Leinders, J. VandenBussche, D. VanGucht, S. Vansummeren, and Y. Wu. Relative expressive power of navigational querying on graphs. In *ICDT*, pages 197–207, 2011.

[15] M. C. Golumbic. The complexity of comparability graph recognition and coloring. *Computing*, 18(3):199–208, 1977.

[16] G. Graefe and R. L. Cole. Fast algorithms for universal quantification in large databases. *ACM Trans. Database Syst.*, 20(2):187–236, 1995.

[17] S. Helmer and G. Moerkotte. Evaluation of main memory join algorithms for joins with set comparison join predicates. In *VLDB*, pages 386–395, 1997.

[18] P. Hsu and D. S. P. Jr. Improving SQL with generalized quantifiers. In *ICDE*, pages 298–305, 1995.

[19] T. S. Jayram, R. Kumar, and D. Sivakumar. The one-way communication complexity of hamming distance. *Theory of Computing*, 4(1):129–135, 2008.

[20] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *PODS*, pages 41–52, 2010.

[21] D. Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32(4):422–469, 2000.

[22] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge: Cambridge Univ, 1997.

[23] D. Leinders and J. VandenBussche. On the complexity of division and set joins in the relational algebra. *J. Comput. Syst. Sci.*, 73(4):538–549, 2007.

[24] G. Li, D. Deng, J. Wang, and J. Feng. PASS-JOIN: A partition-based method for similarity joins. *PVLDB*, 5(3):253–264, 2011.

[25] A. Lingas. A fast output-sensitive algorithm for boolean matrix multiplication. In *ESA*, pages 408–419, 2009.

[26] N. Mamoulis. Efficient processing of joins on set-valued attributes. In *SIGMOD*, pages 157–168, 2003.

[27] R. M. McConnell and J. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *SODA*, pages 536–545, 1994.

[28] S. Melnik and H. Garcia-Molina. Adaptive algorithms for set containment joins. *ACM Trans. Database Syst.*, 28:56–99, 2003.

[29] P. Mishra and M. H. Eich. Join processing in relational databases. *ACM Comput. Surv.*, 24(1):63–113, 1992.

[30] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[31] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

[32] I. Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2), 1991.

[33] A. Okcan and M. Riedewald. Processing theta-joins using mapreduce. In *SIGMOD*, pages 949–960, 2011.

[34] R. Pagh. Compressed matrix multiplication. In *ITCS*, pages 442–451, 2012.

[35] V. R. Pratt. Origins of the calculus of binary relations. In *LICS*, pages 248–254, 1992.

[36] K. Ramasamy, J. M. Patel, J. F. Naughton, and R. Kaushik. Set containment joins: The good, the bad and the ugly. In *VLDB*, pages 351–362, 2000.

[37] R. Rantzau. *Query processing concepts and techniques for set containment tests*. PhD thesis, University of Stuttgart, 2004.

[38] S. Rao, A. Badia, and D. VanGucht. Providing better support for a class of decision support queries. In *SIGMOD*, pages 217–227, 1996.

[39] A. A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

[40] M. Saglam and G. Tardos. On the communication complexity of sparse set disjointness and exists-equal problems. In *FOCS*, pages 678–687, 2013.

[41] C. Schnorr and C. R. Subramanian. Almost optimal (on the average) combinatorial algorithms for boolean matrix product witnesses, computing the diameter (extended abstract). In *RANDOM*, pages 218–231, 1998.

[42] B. ten Cate and M. Marx. Navigational xpath: calculus and algebra. *SIGMOD Record*, 36(2):19–26, 2007.

[43] R. Vernica, M. J. Carey, and C. Li. Efficient parallel set-similarity joins using mapreduce. In *SIGMOD*, pages 495–506, 2010.

[44] R. Williams and H. Yu. Finding orthogonal vectors in discrete structures. In *SODA*, pages 1867–1877, 2014.

[45] D. P. Woodruff and Q. Zhang. An optimal lower bound for distinct elements in the message passing model. In *SODA*, pages 718–733, 2014.

[46] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *WWW*, pages 131–140, 2008.

[47] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *FOCS*, pages 222–227, 1977.

APPENDIX

A. AT-LEAST-T JOIN

Consider the ATJ problem where Alice has sets $A_1, \ldots, A_m$, and Bob has sets $B_1, \ldots, B_m$ all from a universe $[n]$. They want to compute the number of (i, j) pairs for which $|A_i \cap B_j| \geq T$, for a parameter $T > 0$. By padding each set with $T - 1$ common elements, Theorem 10 implies an $\Omega(mn)$ lower bound on the two-way randomized communication complexity of this problem, that is, no non-trivial algorithm exists.

The situation becomes more interesting if we want to approximate the number of (i, j) pairs for which $|A_i \cap B_j| \geq T$ up to a multiplicative factor of $c > 1$, which we call c-ATJ. To obtain a lower bound for the randomized communication of this problem for any approximation factor c, it suffices to obtain a lower bound for the problem of determining if there exists a pair (i, j) for which $|A_i \cap B_j| \geq T$. We call the latter problem $\exists$-ATJ. We give tight bounds for the one-way communication of this problem below, as well as a lower bound for the two-way communication of this problem which is tight if $T = \Omega(n)$ and m and n are polynomially related. By the aforementioned connection, we obtain the same lower bounds for c-ATJ.

Theorem 14 (restatement of Theorem 11) *For $T \geq \log m$, $R^{\rightarrow}_{1/3}(c\text{-ATJ}) = \Omega(mn)$.*

PROOF. We reduce from the Indexing problem to $\exists$-ATJ. In our reduction, Bob has a single non-empty set B_1. In each of Alice's sets A_i, in its characteristic vector on the first $r = 2T$ coordinates, she places a new *distinct* vector with $T - 1$ ones. Assuming $T \geq \log m$, it is possible to do this, since in that case there are $\binom{2T}{T-1} > m$ such vectors. This ensures that the characteristic vectors of any two sets $A_j, A_{j'}$ with $j \neq j'$ agree on at most $T - 2$ ones among the first r coordinates. On the remaining $n - r$ coordinates of the characteristic vectors of each of her m sets she places her input bits to the Indexing problem. Hence, we can embed an instance of Indexing on $(m - r)n = \Omega(mn)$ bits in Alice's vectors.

If Bob is interested in the i-th bit of the characteristic vector of A_j in the Indexing problem, in his set B_1 he puts the prefix corresponding to A_j in the first r coordinates, then he adds the single coordinate i to A_j. Then the only set this can intersect in T positions is A_j. But this happens if and only if i occurs in A_j. □

Theorem 15 (restatement of Theorem 12) *There is a constant $\kappa > 0$, so that for any $\kappa \log m \leq T \leq 99n/100$, it holds that $R_{1/3}(c\text{-ATJ}) = \Omega(m\sqrt{nT})$.*

PROOF. We reduce from SI on size $m\sqrt{nT/8}$ sets. That is, Alice is given $x \in \{0,1\}^{m\sqrt{nT/8}}$, while Bob is given $y \in \{0,1\}^{m\sqrt{nT/8}}$, and the players would like to determine if there exists an ℓ for which $x_\ell = y_\ell = 1$ (see Section 2 for a definition and discussion of SI).

We can partition x into m contiguous substrings

$$x^1, \ldots, x^m \in \{0,1\}^{\sqrt{nT/8}},$$

and similarly partition y into m contiguous substrings

$$y^1, \ldots, y^m \in \{0,1\}^{\sqrt{nT/8}}.$$

For each $i \in [m]$, we use the public randomness to define a random injection $f^i : [\sqrt{nT/8}] \to [n - 48T]$, where we assume $\sqrt{nT/8} \leq n - 48T$ (we remove this assumption below). The f^i are independent for the different i.

We define the following event $\mathcal{E}_{i,j}$ for $i \neq j \in [m]$: $|f^i(x^i) \cap f^j(y^j)| > T/2$, where $|f^i(x^i) \cap f^j(y^j)|$ denotes the number of coordinates ℓ for which $f^i(x^i)_\ell = f^j(y^j)_\ell = 1$. Since f^i is an injection, the number of coordinates $\ell \in [n - 48T]$ for which $f^i(x^i)_\ell = 1$ is at most $\sqrt{nT/8}$, and similarly the number of coordinates ℓ in $[n - 48T]$ for which $f^j(x^j)_\ell = 1$ is at most $\sqrt{nT/8}$.

We compute $\mathbf{Pr}[|f^i(x^i) \cap f^j(y^j)| > T/2]$. This probability cannot decrease if $f^i(x^i)$ and $f^j(y^j)$ are 1 on the maximum possible number of coordinates, namely, $\sqrt{nT/8}$. Then the probability $|f^i(x^i) \cap f^j(y^j)|$ exceeds $T/2$ can be upper bounded by the following process: we choose the $\sqrt{nT/8}$ ones of $f^j(y^j)$ one at a time. Each time we choose a one, there are at most $\sqrt{nT/8}$ positions it could collide with in $f^i(x^i)$, out of at least $n - 48T - \sqrt{nT/8} \geq n/2$ remaining positions, where we assume $48T + \sqrt{nT/8} \leq n/2$ (we remove this assumption below). Hence, $\mathbf{Pr}[|f^i(x^i) \cap f^j(y^j)| > T/2]$ is upper bounded by $\mathbf{Pr}[Z > T/2]$, where Z is the sum of $\sqrt{nT/8}$ independent indicator random variables each with success probability $\sqrt{nT/8}/(n/2)$. Hence, $\mathbf{E}[Z] = T/4$. By a Chernoff bound,

$$\mathbf{Pr}[Z > T/2] \leq \exp(-\Theta(T)) \leq \frac{1}{10m^2},$$

where the second inequality uses that $T \geq \kappa \log m$ for a sufficiently large constant $\kappa > 0$. By a union bound $\mathbf{Pr}[\mathcal{E}] > \frac{9}{10}$, which we condition on.

For each $i \in [m]$, we also choose a random subset S^i of $[48T]$ with $|S^i| = T - 1$. Note that for $i \neq j$,

$$\begin{aligned}
\mathbf{Pr}[|S^i \cap S^j| \geq T/2] &\leq \binom{T-1}{T/2}\left(\frac{1}{48}\right)^{T/2} \\
&\leq \left(\frac{2Te}{T}\right)^{T/2}\left(\frac{1}{48}\right)^{T/2} \\
&\leq \left(\frac{2e}{48}\right)^{T/2} \\
&< \left(\frac{1}{4}\right)^{T} \\
&\leq \frac{1}{m^2},
\end{aligned}$$

where the fist inequality follows by a union bound over all possibilities of intersecting in at least $T/2$ positions, the second inequality follows from the inequality $\binom{n}{k} \leq (ne/k)^k$, and the final inequality uses that $T \geq \log m$. It follows by a union bound that there exist sets $S^1, \ldots, S^m$ with $|S^i| = T - 1$ and $|S^i \cap S^j| < T/2$ for all $i \neq j \in [m]$. We fix a choice of $S^1, \ldots, S^m$ which has this property. Note that our choice of $S^1, \ldots, S^m$ is done independently of our choice of $f^1, \ldots, f^m$.

For each $i \in [m]$, we define $A_i \subseteq [n]$ to be the set whose

characteristic vector is $f(x^i)$ on the first $n - 48T$ coordinates, and which equals S^i on the remaining $48T$ coordinates. Similarly, define $B_i \subseteq [n]$ to be the set whose characteristic vector is $f(y^i)$ on the first $n - 48T$ coordinates, and which equals S^i on the remaining $48T$ coordinates.

For $i \neq j$, we have

$$|A_i \cap B_j| = |f^i(x^i) \cap f^j(y^j)| + |S^i \cap S^j| \leq \frac{T}{2} + \frac{T}{2} - 1 < T.$$

On the other hand, we have

$$|A_i \cap B_i| = |x^i \cap y^i| + T - 1,$$

where $|x^i \cap y^i|$ denotes the number of coordinates ℓ for which $x^i_\ell = y^j_\ell = 1$. It follows that the $\exists$-ATJ problem on inputs $A_1, \ldots, A_m$ and $B_1, \ldots, B_m$ is equal to 1 if and only if there is an ℓ for which $x_\ell = y_\ell = 1$. Hence, a protocol for $\exists$-SITJ which errs with probability $1/5$ can be used to solve SI with error probability at most $1/5 + 1/10 < 1/3$, and so $R_{1/5}(c\text{-ATJ}) \geq R_{1/5}(\exists\text{-ATJ}) \geq R_{1/3}(\text{SI}) = \Omega(m\sqrt{nT})$. Note that $R_{1/3}(c\text{-ATJ}) = \Omega(R_{1/5}(c\text{-ATJ}))$ since a protocol which errs with probability $1/3$ can be made to err with probability $1/5$ by repeating the protocol independently $O(1)$ times and outputting the majority outcome.

It remains to discuss the two assumptions in the above proof, (1) $\sqrt{nT/8} \leq n - 48T$ and (2) $48T + \sqrt{nT/8} \leq n/2$. Both assumptions are satisfied provided that $T \leq cn$ for a sufficiently small constant $c > 0$. This is always possible to assume, since given $T \leq 99n/100$ and n, we can replace (n, T) with $(n - (1-\gamma)T, \gamma T)$ for arbitrarily small constant $\gamma > 0$ and pad each of the sets A_i, B_j in the above with $(1-\gamma)T$ common elements. Since $T \leq 99n/100$, for sufficiently small γ the ratio $(\gamma T)/(n - (1-\gamma)T)$ is at most c, while n and T change by constant factors, so the same conclusion of the $\Omega(m\sqrt{nT})$ lower bound holds. $\square$

B. OMITTED PROOFS

B.1 Proof for Theorem 10

We reduce from the SI problem with a universe of size mn. That is, Alice and Bob have vectors $x, y \in \{0,1\}^{mn}$ and would like to determine if there is an $\ell \in [mn]$ for which $x_\ell = y_\ell = 1$, in which case we say x and y intersect. We use the distribution ν on (x, y) (see Section 2).

Break x into m contiguous strings $x^1, \ldots, x^m \in \{0,1\}^n$, and set A_i such that x^i is the characteristic vector of A_i. Similarly, break y into m contiguous strings $y^1, \ldots, y^m \in \{0,1\}^n$ and set B_j such that y^j is the characteristic vector of B_j.

Claim 1 *With probability $1 - m^2 \cdot e^{-\Omega(n)}$ over $(x, y) \sim \nu$, for all $i \neq j$, x^i intersects y^j.*

PROOF. For fixed i and j, by a Chernoff bound, the Hamming weights of x^i and of y^j are at least $n/5$ with probability at least $1 - e^{-\Omega(n)}$. Conditioned on the Hamming weights of x^i and y^j, the positions of the 1s in x^i and y^j are independent, since $i \neq j$. Hence, for any fixing of Hamming weights of value at least $n/5$, the probability x^i does not intersect y^j is at most the probability that a random set $B \subset [n]$ of size $n/5$ does not contain an element in $[n/5]$, which is at most $(4/5)^{n/5} = e^{-\Omega(n)}$. The claim follows by a union bound over the $\binom{m}{2}$ pairs (i, j). $\square$

If $\text{SI}(x, y) = 1$, then one additional pair (x^i, y^i) will intersect, otherwise if $\text{SI}(x, y) = 0$, then all additional pairs (x^i, y^i) will not intersect. These cases can be distinguished by an algorithm for SDJ since by Claim 1, $\text{SDJ}(\mathcal{A}, \mathcal{B}) = m - \text{SI}(x, y)$ with probability $1 - o(1)$. Lemma 4 states that $D^\nu_\delta(\text{SI}) = \Omega(mn)$, for a constant error probability $\delta > 0$, yielding the theorem.