Vector-Matrix-Vector Queries for Solving Linear Algebra, Statistics, and Graph Problems

Cyrus Rashtchian
Department of Computer Science & Engineering, UC San Diego, CA, USA
crashtchian@eng.ucsd.edu

David P. Woodruff
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
dwoodruf@cs.cmu.edu

Hanlin Zhu
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
zhuhl17@mails.tsinghua.edu.cn

Abstract

We consider the general problem of learning about a matrix through vector-matrix-vector queries. These queries provide the value of $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}$ over a fixed field $\mathbb{F}$ for a specified pair of vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}^n$. To motivate these queries, we observe that they generalize many previously studied models, such as independent set queries, cut queries, and standard graph queries. They also specialize the recently studied matrix-vector query model. Our work is exploratory and broad, and we provide new upper and lower bounds for a wide variety of problems, spanning linear algebra, statistics, and graphs. Many of our results are nearly tight, and we use diverse techniques from linear algebra, randomized algorithms, and communication complexity.

2012 ACM Subject Classification Theory of computation → Stochastic approximation

Keywords and phrases Query complexity, property testing, vector-matrix-vector, linear algebra, statistics, graph parameter estimation

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.26

Category RANDOM

Funding *David P. Woodruff*: D. Woodruff would like to thank support in part by the Office of Naval Research (ONR) grant N00014-18-1-2562.

1 Introduction

In the past few decades, there has been a significant amount of research on query-based algorithms, motivated by compressed sensing, streaming, sketching, distributed methods, graph parameter estimation, and property testing [14, 23, 26, 43, 45]. Most of this work focuses on local queries that only access a small portion of the unknown data at a time. For example, prior work on graph parameter estimation has considered *degree* queries (which output the degree of a vertex v), *edge existence* queries (which answer whether a pair $\{u, v\}$ forms an edge), and *neighbor* queries (which provide the ith neighbor of a vertex v). Not surprisingly, such queries have limited utility for certain problems. Even estimating the number of edges in a graph is known to require a polynomial number of edge existence, degree, and neighbor queries [24, 25].

This has led researchers to consider queries that still reveal a small amount of information, while being more global in nature. For example, *bipartite independent set* queries (which indicate whether or not there is at least one edge between two disjoint sets of vertices) can be used to estimate the number of edges with only polylog(n) queries [9, 19]. Similarly, *cut* queries (which provide the number of edges crossing a graph cut) can be used to find the

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka; Article No. 26; pp. 26:1–26:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

exact minimum cut in a graph [39, 35]. Augmenting edge existence, degree, and neighbor queries with access to an edge sampling oracle (which provides a uniformly random edge) leads to elegant algorithms for estimating the number of certain subgraphs (e.g., triangles or cliques) [5], which was a major open problem (without edge sampling) until recently [21, 40].

As the diversity of queries increases (along with the range of applicable problems), it is natural to wonder whether there is a more general framework for understanding the power and limitations of query-based algorithms. In this work, we initiate the study of querying a matrix through bilinear forms, which generalizes the above mentioned queries and several more (sometimes with an $O(\log n)$ factor overhead). Formally, let $\boldsymbol{M}$ be an $n \times n$ matrix over a field $\mathbb{F}$. We consider vector-matrix-vector queries, which we call $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries for short. Given a pair of vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}^n$, these queries return the value of $\boldsymbol{u}^\mathrm{T}\boldsymbol{M}\boldsymbol{v}$ over $\mathbb{F}$. For graph applications, we often let the matrix $\boldsymbol{M}$ be the adjacency matrix of a graph. We later explain how to simulate standard graph queries with $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries. Allowing $\boldsymbol{M}$ to take values in other fields enables us to consider a greater variety of linear algebra, statistics, and data analytic problems. The underlying field $\mathbb{F}$ will play an important role in our results, where working over $\mathbb{F}_2$ or $\mathbb{R}$ will change the query complexity of certain problems. We assume that the entries have $O(\log n)$ bit-complexity, and therefore, the output of one $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ query provides only $O(\log n)$ bits of information. We strive for algorithms using a subquadratic number of queries, which allows us to solve the problem without trivially learning the whole matrix. Unless we specify otherwise, we allow the queries to be randomized and adaptive.

From a practical point of view, algorithms based on $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries would most likely be useful in the context of specialized hardware or distributed environments. Computing a query only requires a weighted sum of entries of $\boldsymbol{M}$, and hence, it would be easy to execute in a massively parallel fashion. For example, if each processor handled a single row, then the local memory would be bounded by $O(n \log n)$ for storing $\boldsymbol{u}$ and $\boldsymbol{v}$. In a shared-nothing system, the number of communication rounds would be proportional to the number of queries. Similarly, in a streaming environment where single entries of $\boldsymbol{M}$ are changed at each step, the memory would be $O(\log n)$ times the number of queries. Working over a finite field $\mathbb{F}$ would reduce the memory overhead to $O(\log |\mathbb{F}|)$.

That being said, our focus is on the theoretical aspects of the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ query model. We consider many problems, spanning linear algebra, statistics, and graph properties. Part of our motivation comes from finding algorithms that are query-efficient in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model, while surpassing lower bounds for more restricted models. For example, we consider properties that depend on the whole matrix (e.g., having low rank, being unitary or doubly stochastic) or the entire graph (e.g., being a perfect matching or a star). As these are global properties, it is intuitively challenging to verify them using local queries without simply learning the whole matrix or graph. Overall, the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ query model opens up many theoretical directions, and it facilitates new connections between linear algebra, randomized algorithms, and communication complexity.

1.1 Related work and other queries

The $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model provides a unifying lens and generalizes many previously studied queries. We give a brief overview of how to simulate other models and mention other related work.

- **Standard Graph Queries.** To gain intuition about $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries, we note that if $\boldsymbol{M}$ is the adjacency matrix of a graph, then a single query over a large field (e.g., $\mathbb{Q}$ or $\mathbb{R}$) provides the exact edge count. It is also easy to show that $O(\log n)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries suffice to simulate degree, edge existence, neighbor, or edge sampling queries

(see Section A.1 for details). Therefore, $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries achieve a variety of previous results with only an $O(\log n)$ factor overhead, such as estimating the number of cliques of different sizes [2, 5, 21, 22, 40], the number of stars [27], and the minimum vertex cover [37].

- **Independent Set Queries.** Another line of work considers *independent set* oracles for graphs (which return whether a given set of vertices induces an independent set or contains at least one edge), mostly in the context of estimating the number of edges in a graph [9, 17, 18, 19]. Interestingly, bipartite independent set queries are known to be much stronger than independent set queries [9, 17]. A special case of bipartite independent set query (where one of the bipartition sets is a singleton) has been used for testing k-colorability of graphs [10], and high degree vertex discovery [44]. While these algorithms are randomized and approximate, prior work also studies exact graph learning problems [1, 3, 4]. When $\boldsymbol{M}$ is a binary matrix over a large enough field (e.g., $\mathbb{Q}$ or $\mathbb{R}$), then $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries generalize both independent and bipartite independent set queries by taking $\boldsymbol{u}$ and $\boldsymbol{v}$ to be indicator vectors for the sets. In particular, the power of the bipartite version motivates allowing $\boldsymbol{u}$ and $\boldsymbol{v}$ to differ in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model.
- **Fine-Grained Complexity.** Independent set queries are partially motivated by studying the complexity of decision vs. counting problems [18, 19]. While we do not know of a (natural) use of $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries in this area, future work could consider using our algorithms for a similar complexity-theoretic reduction. Our model could also be extended to tensors, where queries are k-linear forms, analogous the generalization to k-partite independent set queries for counting k-cliques, which has applications to k-SUM and related problems [19].
- **Cut Queries.** Another global graph query model considers *cut* queries (which provide the number of edges in a graph $G = (V, E)$ crossing a cut $(S, V \setminus S)$). It is known that $\widetilde{O}(n)$ cut queries suffice to exactly compute a minimum cut in a graph, and $\widetilde{O}(n^{5/3})$ suffice to compute an s-t cut [39]. This has also been extended to multigraphs [35]. We can directly simulate cut queries via indicator vectors $\boldsymbol{u} = \mathbf{1}_S$ and $\boldsymbol{v} = \mathbf{1}_{\{V \setminus S\}}$, when $\boldsymbol{M}$ is the adjacency matrix of the graph. As the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model is more general than cut queries, it an interesting open question whether a sublinear number of queries suffice for these problems.
- **Matrix-Vector Queries.** A similar but more powerful query model considered by previous work involves matrix-vector queries [42]. In this case, the queries return a vector of n values $\boldsymbol{v}^\mathrm{T}\boldsymbol{M}$ or $\boldsymbol{M}\boldsymbol{v}$ when given a vector $\boldsymbol{v} \in \mathbb{F}^n$. We study many of the same problems as this prior work. In some cases, we show that certain problems (such as determining if a matrix is symmetric or diagonal) have constant query complexity in both models, even though $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries reveal much less information than matrix-vector queries. Previous work also considers lower bounds for the operator norm in the matrix-vector model [13]. There is also work on the query complexity of computing PCA in this model [41]. Finally, we provide examples where matrix-vector queries are more powerful because there are lower bounds for $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries (see, e.g., Section 3.1).
- $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ **Data Structures.** A complementary line of work considers the data structure complexity of $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries [15, 16, 20, 30, 36]. More precisely, the goal is to preprocess $\boldsymbol{M}$ using a small amount space so that the value of $\boldsymbol{u}^\mathrm{T}\boldsymbol{M}\boldsymbol{v}$ can be obtained with a small query time (e.g., in the cell-probe model or natural restrictions of that model). Since there are connections between such data structures and challenging complexity theoretic problems (e.g., matrix rigidity, see [20, 36]), it is an outstanding question to further explore whether our results have implications for $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ data structures or vice versa.

Table 1 Our upper and lower bounds on the query complexity in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model for $n \times n$ matrices and constant success probability. Results hold over any field unless stated otherwise.

Linear Algebra Problems		
Schatten p-norm	$\Omega(\sqrt{n})$ for $p \in [0, 4)$, const. factor approx. over $\mathbb{R}$	Theorem 2
	$\Omega(n^{1-2/p})$ for $p \geq 4$, const. factor approx. over $\mathbb{R}$	Theorem 2
Rank testing	$\Omega(k^2)$ to distinguish rank k vs. $k+1$ over $\mathbb{F}_p$	Theorem 3
	$\Omega(n^{2-O(\varepsilon)})$ for $(1 \pm \epsilon)$ approx. over $\mathbb{R}$, non-adaptive	Theorem 4
Trace estimation	$\Omega(n/\log n)$ and $O(n)$ for entries in $\{0, 1, 2, \ldots, n^3\}$	Theorem 5
Diagonal matrix	$O(1)$	Theorem 6
Symmetric matrix	$O(1)$	Theorem 7
Unitary matrix	$\Omega(n/\log n)$ and $O(n)$ for randomized queries over $\mathbb{C}$	Theorem 8
	$\Omega(n^2/\log n)$ for deterministic queries over $\mathbb{C}$	Theorem 10
Statistics Problems		
All ones column	$\Omega(n/\log n)$ and $O(n)$ over $\mathbb{R}$	Section 4.1
Two identical columns	$\Omega(n)$ and $O(n \log n)$ over $\mathbb{F}_2$	Section 4.2
	$O(n)$ over $\mathbb{R}$	Section 4.2
Column-wise majority	$\Theta(n^2)$ over $\mathbb{F}_2$	Corollary 15
Permutation matrix	$O(1)$ over $\mathbb{R}$	Theorem 16
	$\Omega(n)$ over $\mathbb{F}_2$	Theorem 17
Doubly stochastic matrix	$O(1)$ over $\mathbb{R}$	Theorem 18
Negative entry detection	$\Omega(n^2/\log n)$ over $\mathbb{R}$	Theorem 19
Graph Problems		
Triangle detection	$\Omega(n^2/\log n)$	Theorem 20
Star graph	$O(1)$ over $\mathbb{R}$	Theorem 21
Local graph queries	$O(\log n)$	Lemma 22

1.2 Our Results

We provide new upper and lower bounds on the query complexity of various problems in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model. Table 1 summarizes our results. Many of the bounds are nearly tight: for some problems $O(1)$ queries suffice, and for others, either $\widetilde{\Theta}(n)$ or $\widetilde{\Theta}(n^2)$ are necessary and sufficient. For brevity, we defer formal definitions to the relevant subsections. Here we highlight some interesting results.

Linear Algebra Problems. Section 3.1 provides lower bounds for approximately computing many matrix norms, such as the trace norm, Frobenius norm, and operator norm (in general, we study Schatten p-norms; see Section 3.1 for the definition). To prove this result, we develop a general simulation result that allows us to establish lower bounds for adaptive $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries by reducing them to lower bounds for non-adaptive *entry-wise* queries. The key idea is that such a simulation result holds whenever the input matrix distribution is rotationally invariant (under row permutations). Then, we utilize known sketching lower bounds for matrix norms that identify a hard distribution that is rotationally invariant [31].

We give constant-query algorithms for testing if a matrix is diagonal (Section 3.4) or symmetric (Section 3.5). While these algorithms are fairly straightforward, they exhibit the power of $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries to efficiently test for global properties of the matrix. We prove nearly-matching bounds for testing if a matrix is orthonormal (over $\mathbb{R}$) or unitary (over $\mathbb{C}$). The lower bound uses an encoding of information via the Hadamard matrix.

Statistics Problems. Turning to other matrix problems, we consider properties of one or more columns (our results also hold for rows, by symmetry of the query model). For example, Section 4.1 and Section 4.2 provide nearly matching upper and lower bounds for testing if there is an all ones column or two identical columns. Many of our lower bounds follow from communication complexity via Disjointness. While this may be technically simple, we note that our reductions require certain gadgets that seem to be new in the context of query complexity; for example, see our lower bounds for permutation matrices (Theorem 17). This also led us to study negative entry detection in its own right, because a lower bound of $\Omega(n^2/\log n)$ from Theorem 19 essentially provides the reason why certain results for binary matrices (e.g., graphs) cannot be generalized.

Graph Problems. Our upper bound on permutation matrices (Theorem 16) gives a constant-query algorithm for detecting whether a graph is a perfect matching. We also provide a constant-query upper bound for testing whether a graph is a star on n vertices (Theorem 21). Both of these are global properties that would be difficult to verify using standard graph queries. They also complement previous results for learning hidden matchings or other structures using independent set queries [3, 4]. As mentioned previously, being able to simulate local graph queries with $O(\log n)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries gives rise to a number of results on graph parameter estimation in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model (see, e.g., [2, 5, 21, 22, 27, 40, 37]).

Organization. We start with preliminaries in Section 2. We provide results for linear algebra problems in Section 3, for statistics problems in Section 4, and for graph problems in Section 5. We conclude in Section 6.

2 Preliminaries

We use capital bold letters ($\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{M}, \ldots$) to represent matrices, lower-case bold letters ($\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y}, \ldots$) to represent column vectors. We use non-bold lower-case letters ($x, y, \ldots$) to represent strings. For a matrix $\boldsymbol{M}$, we let M_{ij} denote the entry in i^{th} row and j^{th} column. For a vector $\boldsymbol{v}$, we use v_i to denote the ith entry. For a string x, we use x_i to denote the ith entry. We use $\mathbb{F}$ to represent arbitrary fields, and use $\mathbb{F}_p$ to represent the finite field with p elements where p is prime, and $\mathbb{R}$ to denote the reals. We use $G = (V, E)$ to represent a simple graph, where V denotes the set of vertices and E denotes the set of edges. We query the adjacency matrix. Some of our lower bounds use the communication complexity of Disjointness, where Alice has $x \in \{0,1\}^n$, Bob has $y \in \{0,1\}^n$, and they decide if there exists an index i with $x_i = y_i = 1$. The randomized communication complexity is $\Omega(n)$ [29, 38]. We also use the following result: if x and y contain exactly $n/4$ ones, then the randomized complexity is still $\Omega(n)$ [7, 28].

3 Linear Algebra Problems

3.1 Lower Bounds for Approximating Matrix Norms

Say that a distribution over matrices $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ is *orthonormal and rotationally invariant* if all rows of each $\boldsymbol{X}$ in the support are orthonormal and the distribution remains the same under any permutation of the rows of $\boldsymbol{X}$. We will consider distributions over matrices $\boldsymbol{M}$ formed by fixing a diagonal matrix $\boldsymbol{\Sigma}$, sampling two matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ from orthonormal and rotationally invariant distributions, and letting $\boldsymbol{M} = \boldsymbol{X}\boldsymbol{\Sigma}\boldsymbol{Y}^{\mathrm{T}}$. At a high level, are interested in algorithms for computing functions of the singular values $\boldsymbol{\Sigma}$, which remain invariant over matrices in such distributions.

Our first goal is to prove a structural result relating $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries to entry-wise queries of $\boldsymbol{M}$. Then, we use this reduction to prove new lower bounds. To do so, we utilize known streaming lower bounds, and we take advantage of the fact that these lower bounds are based on hard distributions that are orthonormal and rotationally invariant. Recall that $[s] = \{1, 2, \ldots, s\}$ and that $\boldsymbol{e}_i \in \{0,1\}^n$ denotes the ith standard basis vector.

▶ **Lemma 1.** *Let $\boldsymbol{M} = \boldsymbol{X}\boldsymbol{\Sigma}\boldsymbol{Y}^\mathrm{T}$ be a random $n \times n$ real-valued matrix, where $\boldsymbol{\Sigma}$ is diagonal, and $\boldsymbol{X}$ and $\boldsymbol{Y}$ are sampled from orthonormal and rotationally invariant distributions. Any $s \leq n$ deterministic, adaptive queries in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model can be simulated by s^2 non-adaptive entry-wise queries to the values of $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_j$ for $i, j \in [s]$.*

Proof. We proceed by induction on the number of queries $s \geq 1$. For the base case, consider a query $\boldsymbol{u}_1\boldsymbol{M}\boldsymbol{v}_1$, where $\boldsymbol{u}_1, \boldsymbol{v}_1$ are arbitrary unit vectors. Observe that $\boldsymbol{u}_1^\mathrm{T}\boldsymbol{X}$ and $\boldsymbol{Y}^\mathrm{T}\boldsymbol{v}_1$ are random unit vectors, and moreover, they follow the same distribution as $\boldsymbol{e}_1^\mathrm{T}\boldsymbol{X}$ and $\boldsymbol{Y}^\mathrm{T}\boldsymbol{e}_1$, respectively. Since $\boldsymbol{M} = \boldsymbol{X}\boldsymbol{\Sigma}\boldsymbol{Y}^\mathrm{T}$, we see that the values of $\boldsymbol{u}_1^\mathrm{T}\boldsymbol{M}\boldsymbol{v}_1$ and $\boldsymbol{e}_1^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_1$ are identically distributed as well.

Suppose the lemma holds for any $s-1$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries. Consider a sequence of s queries

$$\boldsymbol{u}_1^\mathrm{T}\boldsymbol{M}\boldsymbol{v}_1,\ \boldsymbol{u}_2^\mathrm{T}\boldsymbol{M}\boldsymbol{v}_2,\ \ldots\ ,\ \boldsymbol{u}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{v}_s, \tag{1}$$

for unit vectors $\boldsymbol{u}_i, \boldsymbol{v}_i$ for $i \in [s]$ that may depend adaptively on the previous queries. Assume without loss of generality that $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_s$ and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s$ are respectively linearly independent. For the final query vectors $\boldsymbol{u}_s$ and $\boldsymbol{v}_s$, decompose them as

$$\boldsymbol{u}_s = \boldsymbol{a}_s + \boldsymbol{b}_s \qquad \text{and} \qquad \boldsymbol{v}_s = \boldsymbol{c}_s + \boldsymbol{d}_s,$$

where

$$\boldsymbol{a}_s \in \mathsf{span}\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{s-1}\} \qquad \text{and} \qquad \boldsymbol{c}_s \in \mathsf{span}\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{s-1}\},$$

and where $\boldsymbol{a}_s$ is orthogonal to $\boldsymbol{b}_s$, and $\boldsymbol{c}_s$ is orthogonal to $\boldsymbol{d}_s$.

Invoking the inductive hypothesis, this decomposition implies that $\boldsymbol{a}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{c}_s$ can be simulated using $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_j$ for $i, j \in [s-1]$. Furthermore, by the orthogonality assumptions, we have that $\boldsymbol{b}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s$ follows the same distribution as $\boldsymbol{e}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_s$, even conditioned on the previous queries.

It remains to argue about $\boldsymbol{a}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s$ and $\boldsymbol{b}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{c}_s$. We begin with the former, noting that the latter follows by a symmetric argument. Let $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{s-1}$ denote an orthonormal basis for $\mathsf{span}\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{s-1}\}$. Considering the expansion of $\boldsymbol{a}_s$ in this basis, we observe that, by linearity, it suffices to simulate

$$\boldsymbol{w}_1^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s,\ \boldsymbol{w}_2^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s,\ \ldots\ ,\ \boldsymbol{w}_{s-1}^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s \tag{2}$$

using only the information from $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_s$ for $i \in [s-1]$. To establish this, consider any vector $\boldsymbol{w}_i$ for $i \in [s-1]$. By assumption, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are drawn from orthonormal and rotationally invariant distributions. Since $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_s$ form an orthonormal basis, we have that $\boldsymbol{w}_i^\mathrm{T}\boldsymbol{X}$ is a random unit vector following the same distribution as $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{X}$. Moreover, by orthogonality, for any $i \geq 2$, the distribution of $\boldsymbol{w}_i^\mathrm{T}\boldsymbol{X}$ remains the same as $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{X}$ even conditioned on

$$\boldsymbol{w}_1^\mathrm{T}\boldsymbol{X},\ \boldsymbol{w}_2^\mathrm{T}\boldsymbol{X},\ \ldots\ ,\ \boldsymbol{w}_{i-1}^\mathrm{T}\boldsymbol{X}.$$

an analogous argument implies that $\boldsymbol{Y}^\mathrm{T}\boldsymbol{d}_s$ follows the same distribution as $\boldsymbol{Y}^\mathrm{T}\boldsymbol{e}_s$, even conditioned on the previous queries. Therefore, we have that $\boldsymbol{w}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s$ is identically distributed as $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_s$. As this holds for all $i \in [s-1]$, the queries in Eq. (2) can be simulated by $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_s$ for $i \in [s-1]$. By symmetry, a similar result holds for simulating $\boldsymbol{b}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{c}_s$. Therefore, we have shown that all s deterministic queries in Eq. (1) can be simulated by the s^2 entry-wise non-adaptive queries to $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_j$ for $i, j \in [s]$, as desired. ◀

We use this structural result to prove lower bounds for computing certain matrix norms by applying sketching lower bounds due to Li, Nguyen, and Woodruff [31]. For $p \in (0, \infty)$, the *Schatten p-norm* of a real matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ with singular values $\sigma_1, \ldots, \sigma_n$ is defined as

$$\|\boldsymbol{M}\|_p = \left(\sum_{i=1}^{n} \sigma_i^p \right)^{1/p}.$$

By convention, the Schatten 0-norm is the rank of the matrix, and the Schatten ∞-norm equals the largest singular value (a.k.a., operator norm). We have the following result for the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model.

▶ **Theorem 2.** *Let $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ be a matrix. For any value $p \in [0, 4)$, computing a constant-factor approximation to the Schatten p-norm of $\boldsymbol{M}$ requires $\Omega(\sqrt{n})$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries. For $p \geq 4$, computing a constant-factor approximation to the Schatten p-norm of $\boldsymbol{M}$ requires $\Omega(n^{1-2/p})$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries. Both results hold for randomized, adaptive queries with constant success probability.*

We sketch the proof of this theorem, which now follows directly from previous results. Before applying Lemma 1, we use Yao's principle [46] to show that it suffices to consider deterministic query algorithms for distributions over input matrices. Also, the query vectors can be taken to be unit vectors without loss of generality, as the algorithm can rescale the results. Then, we note that the previous lower bounds use hard distributions that are orthonormal and rotationally invariant [31]. As a result, the distribution of matrices $\boldsymbol{M}$ satisfies the conditions of Lemma 1.

The previous results hold over the *bilinear sketching model*, where the sketches correspond to an $r \times n$ matrix $\boldsymbol{U}$ and an $s \times n$ matrix $\boldsymbol{V}$, and the goal is to approximate $\|\boldsymbol{M}\|_p$ up to a constant factor using $\boldsymbol{U}\boldsymbol{M}\boldsymbol{V}^\mathrm{T}$. Applying Lemma 1, we see that any algorithm making s queries in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model corresponds to a bilinear sketch with both matrices being $s \times n$. Moreover, as the conclusion of the lemma only uses entry-wise queries, the corresponding matrices consist of the $s \times s$ identity matrix in the upper left-hand corner, while the rest of the matrix is all zeroes. The lower bound on bilinear sketches implies

- $s^2 = \Omega(n)$ for approximating the Schatten p-norm with $p \in [0, 4)$
- $s^2 = \Omega(n^{2-4/p})$ for approximating the Schatten p-norm with $p \geq 4$.

Taking a square root leads to the bounds in Theorem 2.

The above provides separations between the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ and matrix-vector models [42]. Indeed, it is known that there exist non-trivial bilinear sketching matrices for approximating the Schatten p-norm whenever p is an even integer. Denoting such sketching matrices as $\boldsymbol{U}$ and $\boldsymbol{V}$, it suffices for $\boldsymbol{U}$ and $\boldsymbol{V}^\mathrm{T}$ to each have $O(n^{1-2/p})$ rows [31] to approximate the Schatten p-norm up to a constant factor. Observe that the Schatten p-norm of a matrix $\boldsymbol{M}$ is the same as the Schatten $p/2$-norm of the matrix $\boldsymbol{M}\boldsymbol{M}^\mathrm{T}$. Thus, if p is an integer multiple of 4, then in the matrix-vector model one can first compute $\boldsymbol{U}\boldsymbol{M}$ and then compute $\boldsymbol{M}^\mathrm{T}\boldsymbol{V}$, and then multiply these together to obtain $\boldsymbol{U}\boldsymbol{M}\boldsymbol{M}^\mathrm{T}\boldsymbol{V}$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the corresponding sketching matrices for the Schatten $p/2$-norm. The total cost is $O(n^{1-4/p})$ queries in the matrix-vector model.

On the other hand, Theorem 2 implies that $\Omega(n^{1-2/p})$ queries are necessary in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model, thus providing a separation for integers $p \geq 4$ which are multiples of 4. We also directly get an $\Omega(n)$ lower bound for approximating the operator norm up to a constant factor, using the $\Omega(n^2)$ lower bound bound for general sketches in [33]. For recent work on actually finding the top eigenvector and solving a linear system in the matrix-vector model in the high accuracy regime, see [13].

3.2 Rank Testing

Given a matrix $\boldsymbol{M} \in \mathbb{F}^{n\times n}$, a natural problem is to determine the rank of $\boldsymbol{M}$. We first consider matrices over a finite field $\mathbb{F}_p$ for a prime p.

▶ **Theorem 3.** *Given a matrix $\boldsymbol{M} \in \mathbb{F}_p^{n\times n}$ and an integer k, at least $\Omega(k^2)$ adaptive queries are necessary to decide the rank whether the rank of $\boldsymbol{M}$ is k or $k+1$ with constant probability.*

Proof. We reduce this problem to a communication complexity problem. Alice holds a matrix $\boldsymbol{A} \in \mathbb{F}_p^{n\times n}$ and Bob holds a matrix $\boldsymbol{B} \in \mathbb{F}_p^{n\times n}$, where $\boldsymbol{M} = \boldsymbol{A}+\boldsymbol{B}$ and $\mathsf{rank}(\boldsymbol{M}) \in \{k, k+1\}$. Corollary 23 in [32] implies that the randomized communication complexity is $\Omega(k^2 \log p)$ to determine whether the rank of $\boldsymbol{M}$ is k or $k+1$. Alice and Bob can simulate the query algorithm using $O(\log p)$ bits of communication per query. Let $q(n,k)$ be the query complexity of this problem in the $\mathsf{u}^\mathsf{T}\mathsf{Mv}$ model. Then $q(n,k)\log p = \Omega(k^2 \log p)$, and we conclude that $q(n,k) = \Omega(k^2)$. ◀

Now consider the real-valued version of rank testing with $\boldsymbol{M} \in \mathbb{R}^{n\times n}$. It is known that if we want to compute the rank of $\boldsymbol{M}$ up to a factor of $(1 \pm \epsilon)$, then this requires $\Omega(n^{2-O(\varepsilon)})$ space in the streaming model [6]. Assadi et. al. [6] has shown that even for some special matrices of which the entries are only in $\{-1, 0, 1\}$, there exists an $\Omega(n^{2-O(\varepsilon)})$ space lower bound for $(1+\varepsilon)$-approximation of the rank. Notice that for $\mathsf{u}^\mathsf{T}\mathsf{Mv}$ queries, if we choose $\boldsymbol{u} = (1, 3, 3^2, \ldots, 3^{m-1})^\mathrm{T}$ and $\boldsymbol{v} = (1, 3^m, 3^{2m}, \ldots, 3^{m(m-1)})^\mathrm{T}$, then we can exactly reconstruct $\boldsymbol{M}$ using the value of $\boldsymbol{u}^\mathrm{T}\boldsymbol{M}\boldsymbol{v}$. Therefore, we assume that the matrix and the query vectors have integral values bounded by a polynomial in n. Under this assumption, we prove the following theorem:

▶ **Theorem 4.** *Given a matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, if we restrict that the entry of query vectors can be chosen only from $\{0, 1, 2, \ldots, n^c\}$ for some constant c, then $\Omega(n^{2-O(\varepsilon)})$ non-adaptive queries are necessary to obtain a $(1+\varepsilon)$-estimation of $\mathsf{rank}(\boldsymbol{M})$.*

Proof. Let $q(n)$ be the number of $\mathsf{u}^\mathsf{T}\mathsf{Mv}$ queries sufficient to estimate the rank up to a factor of $(1 \pm \epsilon)$. Consider a streaming model with updates of the form $\boldsymbol{u}_1^\mathrm{T}\boldsymbol{M}\boldsymbol{v}_1$, $\boldsymbol{u}_2^\mathrm{T}\boldsymbol{M}\boldsymbol{v}_2$, $\ldots$, $\boldsymbol{u}_{q(n)}^\mathrm{T}\boldsymbol{M}\boldsymbol{v}_{q(n)}$, where $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are the ith queries made in the $\mathsf{u}^\mathsf{T}\mathsf{Mv}$ model for $i = 1, 2, \ldots, q(n)$. We can store these queries using $O(q(n) \cdot \log n)$ bits of space (as the matrix and vector entries are polynomially bounded). Using the previous results of [6], we see that $\Omega(n^{2-O(\varepsilon)})$ bits of space are necessary. This implies that $q(n)O(\log n) = \Omega(n^{2-O(\varepsilon)})$, and hence, $q(n) = \Omega(n^{2-O(\varepsilon)}/\log n) = \Omega(n^{2-O(\varepsilon)})$. ◀

3.3 Trace Estimation

Estimating the trace of a matrix presents a simple problem where $\mathsf{u}^\mathsf{T}\mathsf{Mv}$ queries are just as powerful as matrix-vector queries, even though the latter obtains much more information per query. Sun et. al. [42] proves an $\Omega(n/\log n)$ lower bound for trace estimation of symmetric matrix, of which the entries are in $\{0, 1, 2, \ldots, n^3\}$, using matrix-vector queries. Since $\boldsymbol{M}\boldsymbol{v}$ contains all information of $\boldsymbol{u}^\mathrm{T}\boldsymbol{M}\boldsymbol{v}$, it is also a lower bound for the $\mathsf{u}^\mathsf{T}\mathsf{Mv}$ model. Of course, n queries suffice to obtain all the diagonal elements of matrix $\boldsymbol{M}$, i.e., $\mathrm{tr}(\boldsymbol{M}) = \sum_{i=1}^n M_{ii} = \sum_{i=1}^n \boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_i$, where $\boldsymbol{e}_i$ is the ith standard basis vector. Thus, for trace estimation, we obtain an $\Omega(n/\log n)$ lower bound and an $O(n)$ upper bound. We formalize this as the following theorem.

▶ **Theorem 5.** *Let $\boldsymbol{M}$ be an $n \times n$ matrix over $\mathbb{R}$ with entries in $\{0, 1, 2, \ldots, n^3\}$. Assume the query vectors have entries in $\{0, 1, 2, \ldots, n^c\}$ for a constant $c > 0$. Computing a constant factor approximation to the trace $tr(\boldsymbol{M})$ has query complexity between $\Omega(n/\log n)$ and $O(n)$.*

3.4 Deciding if a Matrix is Diagonal

In this section, we show that over any field $\mathbb{F}$, $O(\log(\frac{1}{\varepsilon}))$ queries suffice to test whether a matrix is a diagonal matrix with error probability at most $\varepsilon \in (0,1)$. Actually, it is equivalent that 1 $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ query can test with constant error probability.

For each test, we randomly and uniformly choose a subset S of $[n] = \{1, 2, 3, \ldots, n\}$ with size $|S| = \frac{n}{2}$. We select a subset G of size $|G| = 2$ from $\mathbb{F}$. Construct the query vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ as follows. For each $i \in [n]$, if $i \in S$, then let u_i be randomly and uniformly sampled from G, and $v_i = 0$; otherwise let v_i be randomly and uniformly sampled from G and $u_i = 0$. If $\boldsymbol{u}^\mathrm{T}\boldsymbol{M}\boldsymbol{v} = 0$, then output 'Success', otherwise output 'Fail'. The whole algorithm outputs 'Success' iff every test outputs 'Success'. The proof of Theorem 6 is presented in Appendix A.

▶ **Theorem 6.** *Let $\boldsymbol{M}$ be an $n \times n$ matrix over any field $\mathbb{F}$. Then with $O(\log(\frac{1}{\varepsilon}))$ queries, one can test whether $\boldsymbol{M}$ is a diagonal matrix with probability at least $1 - \varepsilon$.*

3.5 Deciding if a Matrix is Symmetric

Sun et. al. [42] has shown an $O(\log(\frac{1}{\varepsilon}))$ query upper bound in the matrix-vector model to test whether an $n \times n$ matrix $\boldsymbol{M}$ is symmetric with probability $1 - \epsilon$. We can simulate their method in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model. To do so, repeat the following process $O(\log(\frac{1}{\varepsilon}))$ times: choose two random vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ and test whether $\boldsymbol{u}^\mathrm{T}\boldsymbol{M}\boldsymbol{v} = \boldsymbol{v}^\mathrm{T}\boldsymbol{M}\boldsymbol{u}$. Using the previous results [42], we see that the error probability is at most ε. We formalize this as follows:

▶ **Theorem 7.** *Let $\boldsymbol{M}$ be an $n \times n$ matrix over any field $\mathbb{F}$. Then with $O(\log(\frac{1}{\varepsilon}))$ queries, one can test whether $\boldsymbol{M}$ is a symmetric matrix with probability at least $1 - \varepsilon$.*

3.6 Deciding if a Matrix is Unitary

The results on query complexity in this subsection also apply for testing if a matrix is orthonormal over $\mathbb{R}$, since orthonormal is a special case of unitary.

3.6.1 Randomized Queries

Given an $n \times n$ complex matrix $\boldsymbol{M}$, a single matrix-vector query can determine whether $\boldsymbol{M}$ is unitary with probability one [42]. Hence in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model, n randomized queries suffice, by obtaining the entries of the vector $\boldsymbol{M}\boldsymbol{v}$ using $\boldsymbol{u} = \boldsymbol{e}_i$ for $1 \le i \le n$.

Now we show that the $O(n)$ algorithm is nearly optimal by proving a lower bound $\Omega(n/\log n)$ in random case.

▶ **Theorem 8.** *Let $\boldsymbol{M}$ be an $n \times n$ matrix over $\mathbb{C}$. Then to determine whether $\boldsymbol{M}$ is a unitary matrix with a constant probability, the lower bound of query complexity is $\Omega(n/\log n)$ and the upper bound is $O(n)$.*

Proof. WLOG, let $n = 2^k$, and then this problem can be reduced to Disjointness. Suppose Alice has a string $x \in \{0,1\}^n$, and Bob has a string $y \in \{0,1\}^n$. Moreover, $\boldsymbol{x}$ and $\boldsymbol{y}$ both contain exactly $\frac{n}{4}$ ones, i.e. $|\{i \in [n] \mid x_i = 1\}| = \frac{n}{4}$ and $|\{i \in [n] \mid y_i = 1\}| = \frac{n}{4}$. Now Alice and Bob want to find whether there exists an index i such that $x_i = y_i = 1$. The communication complexity of this problem is $\Omega(n)$ [7, 28]. Now we show a protocol of the communication. First, let's recall one construction of a Hadamard matrix.

▶ **Definition 9.** *Let $\boldsymbol{H}_1 = \begin{bmatrix} 1 \end{bmatrix}$, and $\boldsymbol{H}_{2^k} = \begin{bmatrix} \boldsymbol{H}_{2^{k-1}} & \boldsymbol{H}_{2^{k-1}} \\ \boldsymbol{H}_{2^{k-1}} & -\boldsymbol{H}_{2^{k-1}} \end{bmatrix}$ be a Hadamard matrix, then we define $\boldsymbol{G}_{2^k} = \frac{1}{\sqrt{2^k}}\boldsymbol{H}_{2^k}$ for any $k \ge 1$.*

By the definition, $\boldsymbol{G}_{n/4}^{*}\boldsymbol{G}_{n/4} = \boldsymbol{I}_{n/4}$, which means $\boldsymbol{G}_{n/4}$ is a unitary matrix. Also, we denote the element of row i and column j of matrix $\boldsymbol{G}_{n/4}$ by $g_{i,j}$. Then Alice constructs an $n \times n$ matrix $\boldsymbol{X}$ with the following method. Let a_i denote the ith smallest position of string x of which the value is 1. For example, if $x = 00100010$, then $a_1 = 3, a_2 = 7$. Then Alice fills exactly $(n/4) \times (n/4)$ elements of matrix $\boldsymbol{X}$, i.e.

$$\boldsymbol{X}_{a_i,a_j} = \begin{cases} g_{i,j} & ,i \neq j \\ g_{i,j} - 1 & ,i = j, \end{cases} \quad \text{where } 1 \leq i,j \leq n/4.$$

Other elements of $\boldsymbol{X}$ are all 0s. Alice then constructs another matrix $\boldsymbol{X}'$, which is the same as $\boldsymbol{X}$ except that

$$\boldsymbol{X}'_{a_i,a_j} = \begin{cases} -g_{i,j} & ,i \neq j \\ -g_{i,j} - 1 & ,i = j, \end{cases} \quad \text{where } 1 \leq i,j \leq n/4.$$

Let $\boldsymbol{X}_1 = \boldsymbol{X}$ and $\boldsymbol{X}_2 = \boldsymbol{X}'$. Bob uses the similar method to construct matrices $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ using his string y. If x and y are not intersected, i.e. there does not exist an index k, such that $x_k = y_k = 1$, then the four matrices $\boldsymbol{M}_{ij} = \boldsymbol{X}_i + \boldsymbol{Y}_j + \boldsymbol{I}$ are all unitary for $1 \leq i,j \leq 2$, where $\boldsymbol{I}$ is the identity matrix. However, if x and y intersects, then there exists an index k such that $x_k = y_k = 1$, so there exists $i,j \in \{1,2\}$, such that the element of kth row and k^{th} column of matrix $\boldsymbol{M}_{ij}$ equals

$$(-\frac{1}{\sqrt{n/4}} - 1) + (-\frac{1}{\sqrt{n/4}} - 1) + 1 = -1 - \frac{2}{\sqrt{n/4}} < -1,$$

so $\boldsymbol{M}_{ij}$ is not unitary.

Therefore, Alice and Bob can compute $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}_{ij}\boldsymbol{v}$ by sending $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{X}_i\boldsymbol{v}$ and $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{Y}_j\boldsymbol{v}$, which take $O(\log n)$ bits by one communication. Assume that $q(n)$ queries can determine whether an $n \times n$ matrix is unitary, since Disjointness requires $\Omega(n)$ bits, $q(n)O(\log n) = \Omega(n)$, which demonstrates that $q(n) = \Omega(n/\log n)$. We summarize the above results and obtain the following theorem. ◀

3.6.2 Deterministic Queries

For deterministic case, a trivial upper bound is $O(n^2)$ by retrieving all the entries of matrix $\boldsymbol{M}$ one by one. Now we show a strong lower bound $\Omega(n^2/\log n)$, which demonstrates that the trivial algorithm can nearly perform the best.

▶ **Theorem 10.** *Let $\boldsymbol{M}$ be an $n \times n$ matrix over $\mathbb{C}$. Then to determine whether $\boldsymbol{M}$ is a unitary matrix in deterministic case, the lower bound of query complexity is $\Omega(n^2/\log n)$.*

Proof. We reduce the problem to Disjointness. Without loss of generality, let $n = 2^k$. One can calculate easily that there are $n^- = \frac{n(n-1)}{2}$ (-1)s in matrix $\boldsymbol{H}_n$, and $n^+ = \frac{n(n+1)}{2}$ 1s. We denote the element of row i and column j of matrix $\boldsymbol{H}_n$ by $h_{i,j}$. Then, let $\boldsymbol{Z}$ be the matrix of $n \times n$, which satisfies that

$$Z_{i,j} = \begin{cases} -1, & h_{i,j} = -1 \\ 0, & h_{i,j} = 1. \end{cases}$$

Denote the positions in which elements are 1s in $\boldsymbol{H}_n$ by 1, 2, 3, ..., n^+ respectively. Now Alice holds a string x and Bob holds a string y, where $x, y \in \{0,1\}^{n^+}$. Each of them has exactly $\frac{n^+}{2}$ 1s. In the deterministic case, it requires $\Omega(n^+) = \Omega(n^2)$ bits of communication to decide whether the two strings intersect. Alice then constructs a matrix $\boldsymbol{X}$ of $n \times n$, and initially, all the entries of $\boldsymbol{X}$ are zero. Next, for each i, where $1 \leq i \leq n^+$, if $x_i = 1$, then Alice fills in 1 at position i in $\boldsymbol{X}$. Bob constructs a matrix $\boldsymbol{Y}$ with the same method. Let $\boldsymbol{M} = \frac{1}{\sqrt{n}}(\boldsymbol{X} + \boldsymbol{Y} + \boldsymbol{Z})$. Then x and y do not intersect if and only if $\boldsymbol{M}$ is unitary. To exchange $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{v}$ or $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{v}$ needs only $O(\log n)$ bits, so the lower bound is $\Omega(n^2/\log n)$. ◀

4 Statistics Problems

4.1 All Ones Column

Let $\boldsymbol{M} \in \{0,1\}^{n\times n}$ be a binary matrix. Sun et. al. [42] show a lower bound of $\Omega(n/\log n)$ for matrix-vector queries over $\mathbb{R}$ when restricting the entries in the query vector to $[n^c] = \{1, 2, \ldots, n^c\}$ for some constant c. This lower bound can be applied directly to the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model. The following theorem shows that this is tight (up to logarithmic factors) and the proof is presented in Appendix A.

▶ **Theorem 11.** *Given a matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$ over $\mathbb{R}$, then $O(n)$ queries suffice to test whether there exists an all ones column in $\boldsymbol{M}$ with probability one.*

4.2 Identical Columns

Let $\boldsymbol{M} \in \{0,1\}^{n\times n}$. Rearrange $\boldsymbol{M}$ in the following way: $\boldsymbol{M} = \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \cdots & \boldsymbol{c}_n \end{bmatrix}$. Our task is to determine whether there exists i, j, such that $1 \le i < j \le n$ and $\boldsymbol{c}_i = \boldsymbol{c}_j$.

We consider the lower bound of query complexity over $\mathbb{F}_2$ first.

▶ **Theorem 12.** *Let $\boldsymbol{M} \in \{0,1\}^{n\times n}$ be a binary matrix over $\mathbb{F}_2$. Let ε be a real number such that $0 < \varepsilon < 1$ and $n \ge 2(1 + \log \frac{n^2}{\varepsilon})$, then $\Omega(n)$ queries are necessary to detect whether there exist two identical columns in $\boldsymbol{M}$ with probability at least $1 - \varepsilon$.*

Proof. We reduce this problem to Disjointness. Assume Alice has a string $x \in \{0,1\}^{n-1}$, and Bob has a string $y \in \{0,1\}^{n-1}$. Now Alice could construct a matrix $\boldsymbol{X} \in \{0,1\}^{\frac{n}{2}\times n}$, where

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x} & \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_{\frac{n}{2}-1} \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}^{\mathrm{T}}.$$

We denote the jth element of vector $\boldsymbol{a}_i$ as a_{ij}. For each a_{ij}, when $x_j = 1$, we let $a_{ij} = 1$; and when $x_j = 0$, we let a_{ij} be a random variable drawn from a uniform distribution in $\{0,1\}$. Bob constructs $\boldsymbol{Y}$ by the same method. Then let

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{bmatrix}$$

be an $n \times n$ matrix. If x and y intersect, then the corresponding column of $\boldsymbol{M}$ is all ones. Since the last column of $\boldsymbol{M}$ is also all ones, $\boldsymbol{M}$ contains two identical columns. If x and y do not intersect, then for every two columns, the probability that they are identical is at most $\frac{1}{2^{\frac{n}{2}-1}}$. By a union bound, the probability that there exist two identical columns is less than $\binom{n}{2} \times \frac{1}{2^{\frac{n}{2}-1}} \le \frac{n^2}{2^{\frac{n}{2}-1}} \le \varepsilon$, since $n \ge 2(1 + \log \frac{n^2}{\varepsilon})$. Alice and Bob must communicate $\Omega(n)$ bits, and sending $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{v}$ and $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{v}$ each need only one bit over $\mathbb{F}_2$, so $\Omega(n)$ queries are necessary to detect two identical columns. ◀

For the upper bounds over $\mathbb{F}_2$ and $\mathbb{R}$ we have the following theorem.

▶ **Theorem 13.** *Let $\boldsymbol{M} \in \{0,1\}^{n\times n}$ be a binary matrix.*

- *$O(n\log(n/\varepsilon))$ queries over $\mathbb{F}_2$ suffice to detect two identical columns with probability $1-\varepsilon$.*
- *$O(n)$ queries over $\mathbb{R}$ suffice to detect two identical columns with probability one.*

Proof. We choose a random n-dimensional vector $\boldsymbol{u}$, where each u_i is independent. Over $\mathbb{R}$, let u_i be chosen from a standard normal distribution $N(0,1)$; and over $\mathbb{F}_2$, let u_i be chosen uniformly from $\{0,1\}$. Notice that n queries suffice to obtain $\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle$ for $1 \le i \le n$, where $\boldsymbol{c}_i$ is the ith column of $\boldsymbol{M}$. If there are two identical columns $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$, then $\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle = \langle \boldsymbol{u}, \boldsymbol{c}_j \rangle$ always holds.

Now we analyze the probability that $\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle = \langle \boldsymbol{u}, \boldsymbol{c}_j \rangle$ holds for two columns that are not equal. For convenience, let $\boldsymbol{v} = \boldsymbol{c}_i - \boldsymbol{c}_j$. Since $\boldsymbol{c}_i \neq \boldsymbol{c}_j$, we know that $\boldsymbol{v} \neq \boldsymbol{0}$. Assume $v_k \neq 0$ for some index k such that $1 \le k \le n$. When querying over $\mathbb{R}$, we have that

$$\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle - \langle \boldsymbol{u}, \boldsymbol{c}_j \rangle = \langle \boldsymbol{u}, \boldsymbol{c}_i - \boldsymbol{c}_j \rangle = \langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^{n} u_i v_i = u_k v_k + \sum_{i \neq k} u_i v_i.$$

Since $u_k \sim N(0,1)$ and $v_k \neq 0$, we have that $u_k v_k + \sum_{i \neq k} u_i v_i = 0$ with probability 0, which means that $\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle \neq \langle \boldsymbol{u}, \boldsymbol{c}_j \rangle$ with probability one. Therefore, $O(n)$ queries suffice over $\mathbb{R}$ to detect identical columns with probability one.

Working over the field $\mathbb{F}_2$, we see that $u_k v_k + \sum_{i \neq k} u_i v_i = 0$ with probability $\frac{1}{2}$. This means that $\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle = \langle \boldsymbol{u}, \boldsymbol{c}_j \rangle$ with probability $\frac{1}{2}$. If we choose $\log(n^2/\varepsilon) = O(\log(n/\varepsilon))$ independent vectors $\boldsymbol{u}$, then this equality holds for every $\boldsymbol{u}$ with probability $\frac{\varepsilon}{n^2}$. Since there are $\binom{n}{2} \le n^2$ pairs (i,j), the overall error probability is less than $\frac{\varepsilon}{n^2} \cdot n^2 = \varepsilon$ by a union bound. Therefore, the query complexity in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model over $\mathbb{F}_2$ is $O(n \log(n/\varepsilon))$. ◀

4.3 Majority

Given an $n \times n$ matrix $\boldsymbol{M}$ over $\mathbb{F}_2$, we consider computing the *column-wise majority* of $\boldsymbol{M}$. That is, for each column, we compute whether it contains at least $n/2$ ones or not. We prove that $\Theta(n^2)$ queries are necessary and sufficient, even for randomized algorithms.

▶ **Theorem 14.** *Let $\boldsymbol{M} \in \mathbb{F}_2^{n \times n}$ be a binary matrix. Computing the column-wise majority of $\boldsymbol{M}$ requires $\Omega(n^2)$ queries, even for constant success probability.*

Proof. We reduce this problem to Disjointness. Assume Alice has n binary strings of length n, i.e. $x_1, x_2, \ldots, x_n$, each of which contains exactly $\frac{n}{4}$ 1s. Bob has n binary strings of length n, i.e. $y_1, y_2, \ldots, y_n$, each of which contains exactly $\frac{n}{4}$ 1s as well. We define $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ as follows:

$$f(x,y) = \begin{cases} 0, & x \text{ and } y \text{ have non-empty intersection,} \\ 1, & \text{otherwise.} \end{cases}$$

By a direct sum theorem in communication complexity, $\Omega(n^2)$ bits of communication are required to decide $(f(x_1,y_1), f(x_2,y_2), \ldots, f(x_n,y_n))$ simultaneously [34]. Let $\boldsymbol{x}_i$ be the corresponding n-dimensional column vector of string x_i. Also, let $\boldsymbol{y}_i$ be the corresponding column vector of string y_i. Alice and Bob construct matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, where

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_n \end{bmatrix}.$$

Let $\boldsymbol{M} = \boldsymbol{X} + \boldsymbol{Y}$. Then x_i and y_i intersect if and only if the majority of ith column of $\boldsymbol{M}$ is 0 since the elements are over $\mathbb{F}_2$. Furthermore, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}$ can be computed by the communication of $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{v}$ and $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{v}$, each communication requiring one bit. Thus, the number of queries needed to decide the majority of every column is $q(n) = \Omega(n^2)$. ◀

▶ **Corollary 15.** *Given a matrix $\boldsymbol{M} \in \{0,1\}^{n \times n}$ over $\mathbb{F}_2$, then the query complexity of computing the column-wise majority is $\Theta(n^2)$, for both deterministic and randomized queries.*

4.4 Permutation Matrix

A matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$ is a *permutation matrix* if each column and each row contains exactly one entry equal to 1. We consider the query complexity over both $\mathbb{R}$ and $\mathbb{F}_2$, which are very different.

We observe checking if a graph G is a perfect matching is equivalent to checking if the adjacency matrix is a permutation matrix. This also holds for the bipartite version: for a graph on $2n$ vertices, let $M_{ij} = 1$ when the ith vertex on the left is connected to the jth vertex on the right. The following theorem states that $O(1)$ queries suffice over the reals to check whether $\boldsymbol{M}$ is a permutation matrix with constant probability.

▶ **Theorem 16.** *Let $\boldsymbol{M} \in \{0,1\}^{n\times n}$ be a binary matrix over $\mathbb{R}$. Then, $O(1)$ queries suffice to check whether $\boldsymbol{M}$ is a permutation matrix with constant probability.*

Proof. Using a single query $\boldsymbol{u} = \boldsymbol{v} = (1,1,\ldots,1)^{\mathrm{T}}$, we first verify that $\boldsymbol{M}$ contains exactly n ones. Assume this holds. Also, assume without loss of generality that n is even.

We first describe an algorithm to test with constant probability whether each column contains a single one. Reversing the roles of columns and rows will establish the same for rows. The algorithm repeats the following process a constant number of times. Randomly select a subset $A \subseteq [n]$ of exactly $n/2$ columns. Let $\boldsymbol{u}$ be the all ones vector, and let $\boldsymbol{v} = \mathbf{1}_A$ and $\boldsymbol{v}' = \mathbf{1}_{A\setminus[n]}$ be the indicator vectors for A and its complement. Reject if either $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} \neq n/2$ or $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}' \neq n/2$.

If $\boldsymbol{M}$ is a permutation matrix, then $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}' = n/2$ holds. If $\boldsymbol{M}$ is not a permutation matrix, there must be a pair of columns (or rows), one with all zeros, and one with at least two ones. Suppose column c contains all zeros, and column c' contains at least two ones. With constant probability in choosing A, we have $c \in A$ and $c' \notin A$ or vice versa. Conditioned on this, we claim that either $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} \neq n/2$ or $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}' \neq n/2$ with constant probability as well.

To see this, consider randomly partitioning the $n-2$ columns (excluding c and c') into two groups of size $\frac{n}{2}-1$. Let s_1 and s_2 be the number of ones in Groups 1 and 2, respectively. Without loss of generality, assume $s_1 \leq s_2$. Now, consider adding c and c' to the two groups, conditioned on them being separated. If c' is in Group 2, then Group 2 will have more ones than Group 1. Thus, one of the groups must not have $n/2$ ones, and our algorithm rejects with constant probability. ◀

Interestingly, the query complexity depends on the field. If $\boldsymbol{M} \in \{0,1\}^{n\times n}$ is over $\mathbb{F}_2$, then $O(1)$ queries are far from enough.

▶ **Theorem 17.** *Let $\boldsymbol{M} \in \mathbb{F}_2^{n\times n}$ be a matrix. Then, $\Omega(n)$ queries are necessary to determine whether $\boldsymbol{M}$ is a permutation matrix with constant probability.*

Proof. We reduce this problem to Disjointness. Alice holds a string $x \in \{0,1\}^n$ and Bob holds a string $y \in \{0,1\}^n$. Now Alice constructs a $3n \times 3n$ matrix

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{A}_n \end{bmatrix} \qquad \text{where} \qquad \boldsymbol{A}_i = \begin{cases} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & \text{if } x_i = 0 \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \text{if } x_i = 1. \end{cases}$$

Bob constructs a $3n \times 3n$ matrix

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{B}_n \end{bmatrix} \qquad \text{where} \qquad \boldsymbol{B}_i = \begin{cases} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \text{if } y_i = 0 \\ \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} & \text{if } y_i = 1. \end{cases}$$

Let $\boldsymbol{M} = \boldsymbol{A} + \boldsymbol{B}$, with addition over $\mathbb{F}_2$. Then $\boldsymbol{M}$ is a permutation matrix if and only if x and y are disjoint. Thus, the query complexity is $\Omega(n)$ since $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}$ and $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{v}$ are both a single bit. ◀

4.5 Doubly Stochastic Matrix

A non-negative real-valued matrix is *doubly stochastic* if all rows and columns sum to one.

▶ **Theorem 18.** *Let $\boldsymbol{M} \in (\mathbb{R}^+ \cup \{0\})^{n \times n}$ be a non-negative real matrix. Then $O(1)$ queries suffices to check whether $\boldsymbol{M}$ is doubly stochastic with constant probability.*

Proof. First, check whether the sum of all entries is n by choosing $\boldsymbol{u} = \boldsymbol{v} = (1, 1, \ldots, 1)^{\mathrm{T}}$ and checking whether $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = n$. Assume equality holds. If $\boldsymbol{M}$ is not doubly stochastic, then some column c (or row r) has sum > 1 and another column c' (or row r' respectively) has sum < 1. Partition the columns (rows) into two groups of size $\frac{n}{2}$. By the argument in the proof of Theorem 16, two groups sum to different values with constant probability. ◀

4.6 Matrix with Negative Entries

In our previous result for doubly stochastic matrices, we assumed that all the entries are non-negative. This assumption is necessary. If we allow negative entries in a matrix, then even checking whether or not there exists a negative entry requires $\Omega(n^2/\log n)$ queries.

▶ **Theorem 19.** *Let $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ be a matrix. Then $\Omega(n^2/\log n)$ queries are necessary to test if $\boldsymbol{M}$ contains a negative entry using query vectors with entries in $\{0, \pm 1, \pm 2, \ldots, \pm n^c\}$ for a constant c.*

Proof. We reduce this problem to Disjointness. Alice holds a bit-string x with size n^2, and Bob holds a bit-string y with the same size. Alice and Bob construct $n \times n$ matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, where

$$A_{ij} = \begin{cases} 1, & x_{(i-1)n+j} = 0 \\ 0, & x_{(i-1)n+j} = 1 \end{cases} \qquad \text{and} \qquad B_{ij} = \begin{cases} 0, & y_{(i-1)n+j} = 0 \\ -1, & y_{(i-1)n+j} = 1 \end{cases}$$

Let $\boldsymbol{M} = \boldsymbol{A} + \boldsymbol{B}$, and notice that $\boldsymbol{M}$ contains negative entries if and only if x and y intersect. If the query complexity is $q(n)$, then by the Disjointness lower bound, $q(n) \log n = \Omega(n^2)$. ◀

5 Graph Problems

5.1 Triangle Detection

Triangle detection task means that a simple graph G is given in the form of adjacency matrix $\boldsymbol{M} \in \{0, 1\}^{n \times n}$, where n is the number of vertices in G, and we want to decide whether there exists a triangle, i.e. there exists $1 \leq i < j < k \leq n$, such that $M_{ij} = M_{jk} = M_{ki} = 1$. The following theorem shows a lower bound on the number of $\mathsf{u}^{\mathsf{T}}\mathsf{M}\mathsf{v}$ queries to detect a triangle.

▶ **Theorem 20.** *Given a simple graph G consisting of n vertices in the form of its adjacency matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$, then even with a constant probability, $\Omega(n^2/\log n)$ queries are necessary to determine whether there exists a triangle in G.*

Proof. We reduce this problem to a communication complexity problem, that is [8], given a graph G with n vertices, where Alice holds some edges of G, and Bob holds the remaining edges of G, then $\Omega(n^2)$ bits of communication is required to determine whether there exists a triangle in G, even in the random case with a constant probability.

Now suppose the graph is G, and its adjacency matrix is $\boldsymbol{M} \in \{0,1\}^{n\times n}$. Alice holds some edges represented by the matrix $\boldsymbol{X}$, and Bob holds the remaining edges represented by the matrix $\boldsymbol{Y}$. Obviously $\boldsymbol{M} = \boldsymbol{X} + \boldsymbol{Y}$. Then Alice and Bob can communicate by sending $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{v}$ and $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{v}$, and $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}$ can be obtained immediately since $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{v}+\boldsymbol{u}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{v}$. Assume $q(n)$ queries can determine whether there exists a triangle, then $q(n)\log n = \Omega(n^2)$. Thus $q(n) = \Omega(n^2/\log n)$. ◀

5.2 Deciding if a Graph is a Star

Star is a special kind of tree where there exists one vertex adjacent to all the other vertices.

▶ **Theorem 21.** *$\boldsymbol{M} \in \{0,1\}^{n\times n}$ is the adjacency matrix of a simple graph G. Then $O(1)$ queries suffice to determine whether G is a star with constant probability.*

The proof of Theorem 21 is shown in Appendix A.

6 Conclusion

In this paper, we undertook an exploratory study of a new query model that considers querying a matrix through vector-matrix-vector queries. We provided new algorithms and lower bounds for problems spanning three domains: linear algebra, statistics, and graphs. For many of our results, we showed nearly matching bounds on the query complexity, sometimes up to logarithmic factors. We also demonstrated that many previously studied queries can be viewed as special cases or variants of the $\mathsf{u}^{\mathsf{T}}\mathsf{M}\mathsf{v}$ model, and therefore, $\mathsf{u}^{\mathsf{T}}\mathsf{M}\mathsf{v}$ queries provide a unified way to study the query complexity of various graph and matrix problems.

In terms of open questions, an interesting direction would be to identify cases where $\mathsf{u}^{\mathsf{T}}\mathsf{M}\mathsf{v}$ queries are much more efficient than previously studied models. Some options include: determining the minimum cut more efficiently than cut queries [39, 35] or estimating subgraph counts (e.g., triangles) more efficiently than local graph queries [6, 21, 40]. It could also be interesting to study the generalization of our model to k-linear forms (i.e., querying a k-tensor by specifying k vectors), comparing against k-partite independent set queries for counting k-cliques [12, 11, 19].

References

1 Hasan Abasi and Bshouty Nader. On Learning Graphs with Edge-Detecting Queries. In *Algorithmic Learning Theory*, pages 3–30, 2019.

2 Maryam Aliakbarpour, Amartya Shankha Biswas, Themis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, 80(2):668–697, 2018. doi:10.1007/s00453-017-0287-3.

3 Noga Alon and Vera Asodi. Learning a Hidden Subgraph. *SIAM Journal on Discrete Mathematics*, 18(4):697–712, 2005.

4 Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. Learning a hidden matching. *SIAM Journal on Computing*, 33(2):487–501, 2004.

5 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In *Proc. 10th Innovations in Theoretical Computer Science Conference (ITCS)*, 2019.

6 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1723–1742. SIAM, 2017.

7 Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

8 Ziv Bar-Yossef, Ravi Kumar, and D Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 623–632. Society for Industrial and Applied Mathematics, 2002.

9 P. Beame, S. Har-Peled, S. Natarajan Ramamoorthy, C. Rashtchian, and M. Sinha. Edge estimation with independent set oracles. In Anna R. Karlin, editor, *Proc. Innov. Theo. Comp. Sci.* (ITCS), volume 94 of *LIPIcs*, pages 38:1–38:21, 2018.

10 I. Ben-Eliezer, T. Kaufman, M. Krivelevich, and D. Ron. Comparing the strength of query types in property testing: The case of k-colorability. *Computational Complexity*, 22(1):89–135, 2013. doi:10.1007/s00037-012-0048-2.

11 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Hyperedge estimation using polylogarithmic subset queries. *CoRR*, abs/1908.04196, 2019. arXiv:1908.04196.

12 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Triangle estimation using tripartite independent set queries. In Pinyan Lu and Guochuan Zhang, editors, *Proc. 30th Annu. Internat. Sympos. Algorithms Comput.* (ISAAC), volume 149 of *LIPIcs*, pages 19:1–19:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ISAAC.2019.19.

13 Mark Braverman, Elad Hazan, Max Simchowitz, and Blake E. Woodworth. The gradient complexity of linear regression. *CoRR*, abs/1911.02212, 2019. arXiv:1911.02212.

14 Clément L Canonne. A survey on distribution testing: your data is big. but is it blue? In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 22:63, pages 1–1, 2015.

15 Diptarka Chakraborty, Lior Kamma, and Kasper Green Larsen. Tight Cell Probe Bounds for Succinct Boolean Matrix-Vector Multiplication. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1297–1306, 2018.

16 Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation Beats Richness: New Data-Structure Lower Bounds. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1013–1020, 2018.

17 Xi Chen, Amit Levi, and Erik Waingarten. Nearly optimal edge estimation with independent set queries. In Shuchi Chawla, editor, *Proc. 31st ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 2916–2935. SIAM, 2020. doi:10.1137/1.9781611975994.177.

18 Holger Dell and John Lapinskas. Fine-grained reductions from approximate counting to decision. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proc. 50th Annu. ACM Sympos. Theory Comput.* (STOC), pages 281–288. ACM, 2018. doi:10.1145/3188745.3188920.

19 Holger Dell, John Lapinskas, and Kitty Meeks. Approximately counting and sampling small witnesses using a colourful decision oracle. In Shuchi Chawla, editor, *Proc. 31st ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 2201–2211. SIAM, 2020. doi:10.1137/1.9781611975994.135.

20 Zeev Dvir, Alexander Golovnev, and Omri Weinstein. Static Data Structure Lower Bounds Imply Rigidity. In *Proc. 51st Annual ACM SIGACT Symp. on Theory of Computing (STOC)*, pages 967–978, 2019.

21 T. Eden, A. Levi, D. Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. *SIAM J. Comput.*, 46(5):1603–1646, 2017. doi:10.1137/15M1054389.

22 T. Eden, D. Ron, and C. Seshadhri. On Approximating the Number of k-cliques in Sublinear Time. *CoRR*, abs/1707.04858, July 2017. arXiv:1707.04858.

23 Yonina C Eldar and Gitta Kutyniok. *Compressed sensing: theory and applications*. Cambridge university press, 2012.

24 U. Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM J. Comput.*, 35(4):964–984, 2006. doi:10.1137/s0097539704447304.

25 O. Goldreich and D. Ron. Approximating average parameters of graphs. *Random Struct. Algo.*, 32(4):473–493, 2008. doi:10.1002/rsa.v32:4.

26 Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017.

27 M. Gonen, D. Ron, and Y. Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM J. Discrete Math*, 25(3):1365–1411, 2011. doi:10.1137/100783066.

28 Johan Håstad and Avi Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007.

29 Bala Kalyanasundaram and Georg Schintger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

30 Kasper Green Larsen and Ryan Williams. Faster Online Matrix-Vector Multiplication. In *Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2182–2189, 2017.

31 Yi Li, Huy L Nguyen, and David P Woodruff. On approximating matrix norms in data streams. *SIAM Journal on Computing*, 48(6):1643–1697, 2019.

32 Yi Li, Xiaoming Sun, Chengu Wang, and David P Woodruff. On the communication complexity of linear algebraic problems in the message passing model. In *International Symposium on Distributed Computing*, pages 499–513. Springer, 2014.

33 Yi Li and David P. Woodruff. Tight bounds for sketching the operator norm, schatten norms, and subspace embeddings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, pages 39:1–39:11, 2016.

34 Marco Molinaro, David P Woodruff, and Grigory Yaroslavtsev. Beating the direct sum theorem in communication complexity with implications for sketching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1738–1756. SIAM, 2013.

35 Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted Min-Cut: Sequential, Cut-Query and Streaming Algorithms. In *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, 2020.

36 Sivaramakrishnan Natarajan Ramamoorthy and Cyrus Rashtchian. Equivalence of Systematic Linear Data Structures and Matrix Rigidity. In *Proc. 11th Innovations in Theoretical Computer Science Conference (ITCS)*, 2020.

37 K. Onak, D. Ron, M. Rosen, and R. Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proc. 23rd ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 1123–1131, 2012. URL: http://dl.acm.org/citation.cfm?id=2095116.2095204.

38 AA Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106:385–390, 1992.

39 Aviad Rubinstein, Tselil Schramm, and S Matthew Weinberg. Computing exact minimum cuts without knowing the graph. In *Proc. 9th Innovations in Theoretical Computer Science Conference (ITCS)*, 2018.

40 C. Seshadhri. A simpler sublinear algorithm for approximating the triangle count. *CoRR*, abs/1505.01927, May 2015. arXiv:1505.01927.

41 Max Simchowitz, Ahmed El Alaoui, and Benjamin Recht. Tight query complexity lower bounds for PCA via finite sample deformed wigner law. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1249–1259, 2018.

42 Xiaoming Sun, David P Woodruff, Guang Yang, and Jialin Zhang. Querying a matrix through matrix-vector products. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, 2019.

43 Dan Wang and Zhu Han. *Sublinear algorithms for big data applications*. Springer, 2015.

44 J. Wang, E. Lo, and M. L. Yiu. Identifying the most connected vertices in hidden bipartite graphs using group testing. *IEEE Tran. Knowl. Data Eng.*, 25(10):2245–2256, 2013. doi: 10.1109/TKDE.2012.178.

45 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

46 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science*, pages 222–227. IEEE, 1977.

A Remaining Proofs

Proof of Theorem 6. We show that for each query, if $\boldsymbol{M}$ is a diagonal matrix, then the test will always succeed; if not, then the test will fail with constant probability. Then by error reduction, $O(\log(\frac{1}{\varepsilon}))$ queries suffice to achieve error probability at most ε.

For each query, we choose $\boldsymbol{u}$, $\boldsymbol{v}$ as the above algorithm describes. Therefore,

$$\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = \sum_{i\in S, j\in[n]-S} u_i v_j M_{ij}.$$

If $\boldsymbol{M}$ is diagonal, then $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = 0$ always holds. If $\boldsymbol{M}$ is not diagonal, then we claim that $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}$ is non-zero with constant probability. In this case, there exists an off-diagonal element $M_{k\ell} \neq 0$ with $k \neq \ell$. With probability at least $\frac{1}{4}$, $k \in S$ and $\ell \in [n] \setminus S$ simultaneously. Conditioning on this event, let $t_i = \sum_{j\in[n]-S} v_j M_{ij}$, and rewrite $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}$ as

$$\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = \sum_{i\in S} u_i \sum_{j\in[n]-S} v_j M_{ij} = \sum_{i\in S} u_i t_i.$$

Since

$$t_p = \sum_{j\in[n]\setminus S} v_j M_{kj} = v_\ell M_{k\ell} + \sum_{j\in[n]\setminus S\setminus\{\ell\}} v_j M_{kj} \triangleq v_\ell M_{k\ell} + T,$$

and $M_{k\ell} \neq 0$, $v_\ell M_{k\ell}$ has two different possible values. Moreover, at most one choice satisfies $v_\ell M_{k\ell} + T = 0$. So $t_k \neq 0$ with probability at least $\frac{1}{2}$. Now under the condition that $t_k \neq 0$,

$$\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = \sum_{i\in S} u_i t_i = u_k t_k + \sum_{i\in S\setminus\{k\}} u_i t_i \triangleq u_k t_k + R.$$

Since $t_k \neq 0$, by the same argument, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} \neq 0$ with probability at least $\frac{1}{2}$. Combining all of these events, the test fails with probability at least $\frac{1}{16}$, which completes the proof. ◀

Proof of Theorem 11. We construct a random vector $\boldsymbol{u} \in \mathbb{R}^n$, where each entry u_i is independent and follows the standard Gaussian distribution. Let $\boldsymbol{e}_i$ denote the n dimensional vector with i^{th} entry 1 and all other entries 0s, and let $\boldsymbol{e} = \sum_{i=1}^n \boldsymbol{e}_i$ be the all ones n-dimensional vector. Also, let $\boldsymbol{c}_i$ denote the ith column of matrix $\boldsymbol{M}$. Since we have

$$\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{e}_i = \boldsymbol{u}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \cdots & \boldsymbol{c}_n \end{bmatrix} \boldsymbol{e}_i = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{c}_i = \sum_{j=1}^n u_j c_{ij},$$

if we compute the sum of all entries of $\boldsymbol{u}$, i.e., $s = \sum_{i=1}^n u_i$, then when $\boldsymbol{c}_i$ is an all ones column, all $c_{ij} = 1$, so $s = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{e}_i$. Otherwise,

$$s - \boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{e}_i = \sum_{1\le j\le n, c_{ij}=0} u_j.$$

The above quantity equals to 0 with probability 0, which means $s \neq \boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{e}_i$ with probability one. By querying $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{e}_i$ for $1 \le i \le n$, and comparing the result to s, we can detect whether there is an all ones column with probability one using $O(n)$ queries. ◀

Proof of Theorem 21. First, check whether $\boldsymbol{M}$ contains exact $2(n-1)$ ones. If not, $\boldsymbol{M}$ is obviously not a star. Now we assume that $\boldsymbol{M}$ contains exact $2(n-1)$ ones, which means G contains $(n-1)$ edges. Then equally divide the vertices into 2 groups of size $\frac{n}{2}$ randomly and uniformly. We only need to check whether the sum of degrees in one group is $\frac{n}{2}$, and another $\frac{3n}{2}-2$. If this is true, the algorithm should report that $\boldsymbol{G}$ is a star. Otherwise, the algorithm reports that $\boldsymbol{G}$ is not a star. We prove this algorithm has a constant error probability.

If G is a star, then the sum of degrees in one group is $\frac{n}{2}$, and another $\frac{3n}{2}-2$.

If G is not a star, we claim that there exists two vertices v_1 and v_2 with different degrees, which satisfies $|\deg(v_1)-\deg(v_2)| < n-2$.

Since G is not a star, then the degree of any vertex can be at most $n-2$.

- If there exists a vertex v_1 with degree $n-2$, then when n is large (e.g. $n > 10$), there must exists another vertex v_2 with degree 1. Therefore,

$$|\deg(v_1)-\deg(v_2)| = (n-2)-1 = n-3 < n-2.$$

- If there does not exists a vertex with degree $n-2$, then the degree of all vertices are in $\{0,1,2,\ldots,n-3\}$. When n is large enough (e.g. $n > 10$), there must exist two vertices v_1 and v_2 with different degrees, and

$$|\deg(v_1)-\deg(v_2)| \le (n-3)-0 = n-3 < n-2.$$

Now with probability at least $\frac{1}{2}$, v_1 and v_2 are in the different groups. Without loss of generality, assume $\deg(v_1) > \deg(v_2)$. Conditioned on this, we can decompose the random partition procedure into 2 steps. First, we randomly and uniformly partition the other $n-2$ vertices except v_1 and v_2 into 2 groups with size $\frac{n}{2}-1$. Assume in Group 1 the sum of degrees in these vertices is s_1 and in Group 2 the sum is s_2. Without loss of generality, assume $s_1 \le s_2$. The second step is to place v_1 in Group 1, v_2 in Group 2, or v_2 in Group 2, v_1 in Group 1 both with probability $\frac{1}{2}$.

If we have that

$$\begin{aligned} s_1+\deg(v_1) &= \tfrac{3n}{2}-2, & s_2+\deg(v_2) &= \tfrac{n}{2},\\ s_1+\deg(v_2) &= \tfrac{n}{2}, & s_2+\deg(v_1) &= \tfrac{3n}{2}-2 \end{aligned}$$

hold simultaneously, then $\deg(v_1)-\deg(v_2) = n-2$, a contradiction.

If

$$\begin{aligned} s_1+\deg(v_1) &= \tfrac{n}{2}, & s_2+\deg(v_2) &= \tfrac{3n}{2}-2,\\ s_1+\deg(v_2) &= \tfrac{n}{2}, & s_2+\deg(v_1) &= \tfrac{3n}{2}-2, \end{aligned}$$

then $\deg(v_1) = \deg(v_2)$, a contradiction.

Also, since $s_1 \le s_2, \deg(v_1) > \deg(v_2)$ and $\frac{3n}{2}-2 \ge \frac{n}{2}$, it is impossible that $s_1+\deg(v_2) = \frac{3n}{2}-2$ and $s_2+\deg(v_1) = \frac{n}{2}$ both holds.

So with probability at least $\frac{1}{2}$, the sums of the two groups will not be $\frac{3n}{2}-2$ and $\frac{n}{2}$.

Overall, with probability at least $\frac{1}{4}$, the sums of the two groups will not be $\frac{3n}{2}-2$ and $\frac{n}{2}$. ◀

A.1 Local Graph Queries and Estimating Subgraph Counts

The following four local graph queries can be implemented by $O(\log n)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries.

▶ **Lemma 22.** *Given the adjacency matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$ of a simple graph $G = (V, E)$, then the following four queries:*

- ***Degree query*** *i: the degree of vertex i.*
- ***Neighbor query*** *(i, j): the jth neighbor of vertex i.*
- ***Pair query*** *(i, j): whether the edge (i, j) exists.*
- ***Edge-sample query****: sample an edge e uniformly at random from E.*

can be implemented by $O(\log n)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries.

As one application, we mention the problem of counting subgraphs. Given the adjacency matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$ of a simple graph G, we want to estimate the number of occurrences of H in G, where H is a given subgraph (such as a triangle). Assadi et. al. [5] shows that with

$$\widetilde{O}\left(\min\left\{m, \frac{m^{\rho(H)}}{\#H}\right\}\right)$$

of the above four standard graph queries, we can obtain a $(1\pm\varepsilon)$-approximation to the number of occurrences of H in G with high probability. Here, $\#H$ is the number of occurrences of H in G, m is the number of edges, and $\rho(H)$ is the fractional edge-cover of H. Also, the $\widetilde{O}(\cdot)$ notation ignores ε and $\log n$ terms, as well as the size of graph H. By Lemma 22, the four standard graph queries can be implemented by $O(\log n)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries. Therefore, we derive the following result.

▶ **Proposition 23.** *Given the adjacency matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$ of a simple graph G and an arbitrary small target graph H, $\widetilde{O}\left(min\left\{m, \frac{m^{\rho(H)}}{\#H}\right\}\right)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries suffice to obtain a $(1 \pm \varepsilon)$-approximation to the number of occurrences of H in G with high probability.*

We briefly compare this to work on independent set queries [12, 11, 19]. Proposition 23 achieves a general result for $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries, whereas estimating triangles or other subgraphs with bipartite independent set queries is an open question. Moreover, estimating larger subgraphs seems to require higher-order queries (e.g., tripartite independent set queries). This suggests that, as expected, $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries may be more powerful for a variety of problems.