The Johnson-Lindenstrauss Transform Itself Preserves Differential Privacy

Jeremiah Blocki[*]   Avrim Blum[†]   Anupam Datta[‡]   Or Sheffet[§]

Carnegie Mellon University

{jblocki@cs, avrim@cs, danupam@andrew, osheffet@cs}.cmu.edu

August 21, 2012

Abstract

This paper proves that an "old dog", namely – the classical Johnson-Lindenstrauss transform, "performs new tricks" – it gives a novel way of preserving differential privacy. We show that if we take two databases, D and D', such that (i) $D' - D$ is a rank-1 matrix of bounded norm and (ii) all singular values of D and D' are sufficiently large, then multiplying either D or D' with a vector of iid normal Gaussians yields two statistically close distributions in the sense of differential privacy. Furthermore, a small, deterministic and *public* alteration of the input is enough to assert that all singular values of D are large.

We apply the Johnson-Lindenstrauss transform to the task of approximating cut-queries: the number of edges crossing a $(S, \bar{S})$-cut in a graph. We show that the JL transform allows us to *publish a sanitized graph* that preserves edge differential privacy (where two graphs are neighbors if they differ on a single edge) while adding only $O(|S|/\epsilon)$ random noise to any given query (w.h.p). Comparing the additive noise of our algorithm to existing algorithms for answering cut-queries in a differentially private manner, we outperform all others on small cuts ($|S| = o(n)$).

We also apply our technique to the task of estimating the variance of a given matrix in any given direction. The JL transform allows us to *publish a sanitized covariance matrix* that preserves differential privacy w.r.t bounded changes (each row in the matrix can change by at most a norm-1 vector) while adding random noise of magnitude independent of the size of the matrix (w.h.p). In contrast, existing algorithms introduce an error which depends on the matrix dimensions.

1 Introduction

The celebrated Johnson Lindenstrauss transform [JL84] is widely used across many areas of Computer Science. A very non-exhaustive list of related applications include metric and graph embeddings [Bou85, LLR94], computational speedups [Sar06, Vem05], machine learning [BBV06, Sch00], information retrieval [PRT+98], nearest-neighbor search [Kle97, IM98, AC06], and compressed sensing [BDDW08]. This paper unveils a new application of the Johnson Lindenstrauss transform – it also preserves differential privacy.

Consider a scenario in which a trusted curator gathers personal information from n individuals, and wishes to release statistics about these individuals to the public without compromising any individual's privacy. *Differential privacy* [DMNS06] provides a robust guarantee of privacy for such data releases. It guarantees that for any two neighboring databases (databases that differ on the details of any single individual), the curator's distributions over potential outputs are statistically close (see formal definition in Section 2). By itself, preserving differential privacy isn't hard, since the curator's answers to users' queries can be so noisy that they obliterate any useful data stored in the database. Therefore, the key research question in this field is to provide tight *utility and privacy tradeoffs*.

The most basic technique that preserves differential privacy and gives good utility guarantees is to add relatively small Laplace or Gaussian noise to a query's true answer. This simple technique lies at the core of an overwhelming majority of algorithms that preserve differential privacy. In fact, many differentially private algorithms follow a common outline. They take an existing algorithm and revise it by adding such random noise each time the algorithm operates on the sensitive data. Proving that the revised algorithm preserves differential privacy is almost immediate, because differential privacy is composable. On the other hand, providing good bounds on the revised algorithm's utility follows from bounding the overall noise added to the algorithm, which is often difficult. This work takes the complementary approach. We show that an existing algorithm preserves differential privacy provided we slightly alter the input in a reversible way. Our analysis of the algorithm's utility is immediate, whereas privacy guarantees require a non-trivial proof.

We prove that by multiplying a given database with a vector of iid normal Gaussians, we can output the result while preserving differential privacy (assuming the database has certain properties, see "our technique"). This technique is no other than the Johnson-Lindenstrauss transform, and it's guaranteed to preserve w.h.p the L_2 norm of the given database up to a small multiplicative factor. Therefore, whenever answers to users' queries can be formalized as the length of the product between the given database and a query-vector, utility bounds are straight-forward.

For example, consider the case where our input is composed of n points in $\mathbb{R}^d$ given as a $n \times d$ matrix. We define two matrices as neighbors if they differ on a single row and the norm of the difference is at most 1.[1] Under this notion of neighbors, a simple privacy preserving mechanism allows us to output the mean of the rows in A, but what about the covariance matrix $A^\mathsf{T} A$? We prove that the JL transform gives a (ϵ, δ)-differentially private algorithm that outputs a sanitized covariance matrix. Furthermore, for *directional variance queries*, where users give a unit-length vector x and wish to know the variance of A along x (see definition in Section 2), we give utility bounds that are *independent of d and n*. In contrast, all other differentially private algorithms that answer directional variance queries have utility guarantees that depend on d or n. Observe that

---

[1]This notion of neighboring inputs, also considered in [MM09, HR12], is somewhat different than the typical notion of privacy, allowing any individual to change her attributes arbitrarily.

our utility guarantees are somewhat weaker than usual. Recall that the JL lemma guarantees that w.h.p lengths are preserved up to a small multiplicative error, so for each query our algorithm's estimation has w.h.p small multiplicative error and additional additive error.

A special case of directional variance queries is *cut-queries* of a graph. Suppose our database is a graph G and users wish to know how many edges cross a $(S, \bar{S})$-cut. Such a query can be formalized by the length of the product $E_G \mathbf{1}_S$, where E_G is the *edge-matrix* of G and $\mathbf{1}_S$ is the indicator vector of S (see Section 2). We prove that the JL transform allows us to publish a perturbed Laplacian of G while preserving (ϵ, δ)-differential privacy, w.r.t two graphs being neighbors if they differ only on a single edge. Comparing our algorithm to existing algorithms, we show that we add (w.h.p) $O(|S|)$ random noise to the true answer (alternatively: w.h.p we add only constant noise to the query $\frac{\mathbf{1}_S^\mathsf{T} E_G^\mathsf{T} E_G \mathbf{1}_S}{\mathbf{1}_S^\mathsf{T} \mathbf{1}_S}$). In contrast, all other algorithms add noise proportional to the number of vertices (or edges) in the graph.

Our technique. It is best to demonstrate our technique on a toy example. Assume D is a database represented as a $\{0, 1\}^n$-vector, and suppose we sample a vector Y of n iid normal Gaussians and publish $X = Y^\mathsf{T} D$. Our output is therefore distributed like a Gaussian random variable of 0 mean and variance $\sigma^2 = \|D\|^2$. Assume a single entry in D changes from 0 to 1 and denote the new database as D'. Then $X' = Y^\mathsf{T} D'$ is distributed like a Gaussian of 0-mean and variance $\lambda^2 = \|D\|^2 + 1$. Comparing $\mathsf{PDF}_X(x) = (2\pi\sigma^2)^{-1/2} \exp(-x^2/(2\sigma^2))$ to $\mathsf{PDF}_{X'}(x) = (2\pi\lambda^2)^{-1/2} \exp(-x^2/(2\lambda^2))$ we have that $\forall x$, $\sqrt{\lambda^2/\sigma^2} \mathsf{PDF}_{X'}(x) \geq \mathsf{PDF}_X(x) \geq \exp(-\frac{x^2}{2\sigma^2} \cdot \frac{1}{\lambda^2}) \mathsf{PDF}_{X'}(x)$. Using concentration bounds on Gaussians we deduce that if $\lambda^2 > \sigma^2 = \Omega(\log(1/\delta)/\epsilon)$, then w.p $\geq 1 - \delta$ both PDFs are within multiplicative factor of $e^{\pm\epsilon}$. We now repeat this process r times (setting ϵ, δ accordingly) s.t. the JL lemma assures that (after scaling) w.h.p we output a vector of norm $(1 \pm \eta)\|D\|^2$ for a given η. We get utility guarantees for publishing the number of ones in D while preserving (ϵ, δ)-differential privacy.

Keeping with our toy example, one step remains – to convert the above analysis so that it will hold for any database, and not only databases with $w \overset{\text{def}}{=} \log(1/\delta)/\epsilon$ many ones. One way is to append the data with w one entries, but observe: this ends up in outputting $X + N$ where N is random Gaussian noise! In other word, appending the data with ones makes the above technique worse (noisier) than the classical technique of adding random Gaussian noise. Instead, what we do is to "translate the database". We apply a simple *deterministic* affine transformation s.t. D turns into a $\{\sqrt{\frac{w}{n}}, 1\}^n$-vector. Applying the JL algorithm to the translated database, we output a vector whose norm squared is $\approx (1 \pm \eta)(\|D\|^2 + w)$. Clearly, users can subtract w from the result, and we end up with ηw additive random noise (in addition to the multiplicative noise).[2]

It is tempting to think the above analysis suffices to show that privacy is also preserved in the multidimensional case. After all, if we multiply the edge matrix of a graph G with a vector of iid normal Gaussians, we get a vector with each entry distributed like a Gaussian; and if we replace G with a neighboring G', we affect only two entries in this vector. Presumably, applying the previous analysis to both entries suffices to prove we preserve differential privacy. But this intuition is false. Multiplying E_G with a random vector does not result in n independent Gaussians, but rather in one multivariate Gaussian. This is best illustrated with an example. Suppose G is a graph and S is a subset of nodes s.t. no edge crosses the $(S, \bar{S})$-cut. Therefore $E_G \mathbf{1}_S$ is the zero-vector, and no

---

[2]Observe that in this toy example, our $O(\log(1/\delta)/\epsilon)$ noise bound is still worse than the noise bound of $O(\sqrt{\log(1/\delta)}/\epsilon)$ one gets from adding Gaussian noise. However, in the applications detailed in Sections 3 and 4, the idea of changing the input will be the key ingredient in getting noise bounds that are independent of n and d.

matter what random projection we pick, $Y^\mathsf{T} E_G \mathbf{1}_S = 0$. In contrast, by adding a single edge that crosses the $(S, \bar{S})$-cut, we get a graph G' s.t. $\mathbf{Pr}[Y^\mathsf{T} E_{G'} \mathbf{1}_S \neq 0] = 1$.

Organization. Next we detail related work. Section 2 details important notations and important preliminaries. In Sections 3 and 4 we convert the above univariate intuition to the multivariate Gaussian case. Section 3 describes our results for graphs and cut-queries, and in Section 3.2 we compare our method to other algorithms. Section 4 details the result for directional queries (the general case), then a comparison with other algorithms. Even though there are clear similarities between the analyses in Sections 3 and 4, we provide both because the graph case is simpler and analogous to the univariate Gaussian case. Suppose G and G' are two graphs without and with a certain edge resp., then G induces the multivariate Gaussian with the "smaller" variance, and G' induces the multivariate Gaussian with the "larger" variance. In contrast, in the general case there's no notion of "smaller" and "larger" variances. Also, the noise bound in the general case is larger than the one for the graph case, and the theorems our analysis relies on are more esoteric. Section 5 concludes with a discussion and open problems.

1.1 Related Work

Differential privacy was developed through a series of papers [DN03, DMNS06, CDM+05, BDMN05]. Dwork et al [DMNS06] gave the first formal definition and the description of the basic Laplace mechanism. Its Gaussian equivalent was defined in [DKM+06]. Other mechanisms for preserving differential privacy include the Exponential Mechanism of McSherry and Talwar [MT07, BLR08]; the recent Multiplicative Weights mechanism of Hardt and Rothblum [HR10] and its various extensions [HLM10, GHRU11, GRU12]; the Median Mechanism [RR10] and a boosting mechanism of Dwork et al [DRV10]. In addition, the classical Randomized Response (see [War65]) preserves differential privacy as discussed in recent surveys [DS10, Dwo11]. The task of preserving differential privacy when the given database is a graph or a social network was studied by Hay et al [HLMJ09] who presented a privacy preserving algorithm for publishing the degree distribution in a graph. They also introduce multiple notions of neighboring graphs, one of which is for the change of a single edge. Nissim et al [NRS07] (see full version) studied the case of estimating the number of triangles in a graph, and Karwa et al [KRSY11] extended this result to other graph structures. Gupta et al [GRU12] studied the case of answering (S, T)-cut queries, for two disjoint subsets of nodes S and T. All latter works use the same notion of neighboring graphs as we do. In differential privacy it is common to think of a database as a matrix, but seldom one gives utility guarantees for queries regarding global properties of the input matrix. Blum et al [BDMN05] approximate the input matrix with the PCA construction by adding $O(d^2)$ noise to the input. The work of McSherry and Mironov [MM09] (inspired by the Netflix prize competition) defines neighboring databases as a change in a single entry, and introduces $O(k^2)$ noise while outputting a rank-k approximation of the input. The work of Hardt and Roth [HR12] gives a low-rank approximation of a given input matrix while adding $\min\{\sqrt{d}, \sqrt{n}\}$ noise by following the elegant framework of Halko et al [HMT11]. According to [HR12], a recent and not-yet-published work of Kapralov, McSherry and Talwar preserves rank-1 approximations of a given PSD matrix with error $O(n)$.

The body of work on the JL transform is by now so extensive that only a book may survey it properly [Vem05]. In the context of differential privacy, the JL lemma has been used to reduce dimensionality of an input prior to adding noise or other forms of privacy preservation. Blum et al [BLR08] gave an algorithm that outputs a sanitized dataset for learning large-margin classifiers

by appealing to JL related results of [BBV06]. Hardt and Roth [HR12] gave a privacy preserving version of an algorithm of [HMT11] that uses randomize projections onto the image space of a given matrix. Blum and Roth [**?**] used it to reduce the noise added to answering sparse queries. The way the JL lemma was applied in these works is very different than the way we use it.

2 Basic Definitions, Preliminaries and Notations

Privacy and utility. In this work, we deal with two types of inputs: $[0, 1]$-weighted graphs over n nodes and $n \times d$ real matrices. (We treat $w_{a,b} = 0$ as no edge between a and b). Trivially extending the definition in [NRS07, KRSY11], two weighted n-nodes graphs G and G' are called *neighbors* if they differ on the weight of a single edge (a, b). Like in [HR12], two $n \times d$-matrices are called *neighbors* if all the coordinates on which A and A' differ lie on a single row i, s.t. $\|A_{(i)} - A'_{(i)}\|^2 \le 1$, where $A_{(i)}$ denotes the i-th row of A.

Definition 2.1. *An algorithm* ALG *which maps inputs into some range* $\mathcal{R}$ *maintains* (ϵ, δ)*-differential privacy if for all pairs of neighboring inputs* $\mathcal{I}, \mathcal{I}'$ *and for all subsets* $\mathcal{S} \subset \mathcal{R}$ *it holds that*

$$\mathbf{Pr}[\mathsf{ALG}(\mathcal{I}) \in \mathcal{S}] \le e^\epsilon \mathbf{Pr}[\mathsf{ALG}(\mathcal{I}') \in \mathcal{S}] + \delta$$

For each type of input we are interested in answering a different type of query. For graphs, we are interesting in *cut-queries*: given a nonempty subset S of the vertices of the graph, we wish to know what is the total weight of edges crossing the $(S, \bar{S})$-cut. We denote this as $\Phi_G(S) = \sum_{u \in S, v \notin S} w_{u,v}$.

Definition 2.2. *We say an algorithm* ALG *gives a* (η, τ, ν)*-approximation for cut queries, if for every nonempty* S *it holds that*

$$\mathbf{Pr}\left[(1 - \eta)\Phi_G(S) - \tau \le \mathsf{ALG}(S) \le (1 + \eta)\Phi_G(S) + \tau\right] \ge 1 - \nu$$

For $n \times d$ matrices, we are interested in *directional variance queries*: given a unit-length direction x, we wish to know what's the variance of A along the x direction: $\Phi_A(x) = x^\mathsf{T} A^\mathsf{T} A x$. (Our algorithm normalizes A s.t. the mean of its n rows is 0.)

Definition 2.3. *We say an algorithm* ALG *gives a* (η, τ, ν)*-approximation for directional variance queries, if for every unit-length vector* x *it holds that*

$$\mathbf{Pr}\left[(1 - \eta)\Phi_A(x) - \tau \le \mathsf{ALG}(x) \le (1 + \eta)\Phi_A(x) + \tau\right] \ge 1 - \nu$$

Some Linear Algebra. Given a $m \times n$ matrix M its Singular Value Decomposition (SVD) is $M = U\Sigma V^\mathsf{T}$ where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are unitary matrices, and Σ has non-zero values only on its main diagonal. Furthermore, there are exactly $rank(M)$ positive values on the main diagonal, denoted $\sigma_1(M) \ge \ldots \ge \sigma_{rank(M)}(M)$, called the *singular values*. This allows us to write M as the sum of $rank(M)$ rank-1 matrices: $M = \sum_{i=1}^{rank(M)} \sigma_i u_i v_i^\mathsf{T}$. Because Σ has non-zero values only on its main diagonal, the notation Σ^i denotes a matrix whose non-zero values lie only on the main diagonal and are $\sigma_1^i(M), \sigma_2^i(M), \ldots, \sigma_{rank(M)}^i(M)$. Using the SVD, it is clear that if M is of full-rank, then $M^{-1} = V\Sigma^{-1}U^\mathsf{T}$, and that if $n = m = rank(M)$ then $\det(M) = \prod_{i=1}^n \sigma_i(M)$. Furthermore, even when M is not full-rank, the SVD allows us to use similar notation to denote the generalizations of the inverse and of the determinant: The Moore-Penrose inverse of M is $M^\dagger = V\Sigma^{-1}U^\mathsf{T}$; and

4

the pseudo-determinant of M is $\tilde{\det}(M) = \prod_{i=1}^{rank(M)} \sigma_i(M)$. A $n \times n$ symmetric matrix is called *positive semidefinite* (PSD) if it holds that $x^\mathsf{T} M x \geq 0$ for every $x \in \mathbb{R}^n$. Given two PSDs M and N we denote the fact that $(N - M)$ is PSD by $M \preceq N$. For further details, see [HJ90].

Gaussian distribution. Given a r.v. X, we denote by $X \sim \mathcal{N}(\mu, \sigma^2)$ the fact that X has normal distribution with mean μ and variance σ^2. Recall that $\mathsf{PDF}_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-(x-\mu)^2/2\sigma^2)$. We repeatedly apply the *linear combination* rule: for any two i.i.d normal random variables s.t. $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$, we have that their linear combination $Z = aX + bY$ is distributed according to $Z \sim \mathcal{N}(a\mu_X + b\mu_Y, a^2\sigma_X^2 + b^2\sigma_Y^2)$. This in turn allows us to identify a random variable $R \sim \mathcal{N}(0, \sigma^2)$ with the random variable $\sigma R'$, where $R' \sim \mathcal{N}(0,1)$. Classic concentration bounds on Gaussians give that $Pr[|x - \mu|^2 > \log(1/\delta)\sigma^2] \leq 2\delta$.

The multivariate normal distribution is the multi-dimension extension of the univariate normal distribution. $X \sim \mathcal{N}(\mu, \Sigma)$ denotes a m-dimensional multivariate r.v. whose mean is $\mu \in \mathbb{R}^m$, and variance is the PSD matrix $\Sigma = \mathbf{E}\left[(X - \mu)(X - \mu)^\mathsf{T}\right]$. If Σ has full rank (Σ is positive definite) then $\mathsf{PDF}_X(x) = \frac{1}{\sqrt{(2\pi)^m \det(\Sigma)}} \exp(-\frac{1}{2} x^\mathsf{T} \Sigma^{-1} x)$, a well defined function. If Σ has non-trivial kernel space then PDF_X is technically undefined (since X is defined only on a subspace of volume 0, yet $\int_{\mathbb{R}^m} \mathsf{PDF}_X(x)dx = 1$). However, if we restrict ourselves only to the subspace $\mathcal{V} = (Ker(\Sigma))^\perp$, then $\mathsf{PDF}_X^\mathcal{V}$ is defined over $\mathcal{V}$ and $\mathsf{PDF}_X^\mathcal{V}(x) = \frac{1}{\sqrt{(2\pi)^{rank(\Sigma)} \tilde{\det}(\Sigma)}} \exp(-\frac{1}{2} x^\mathsf{T} \Sigma^\dagger x)$. From now on, we omit the superscript from the PDF and refer to the above function as the PDF of X. Observe that using the SVD, we can denote $\Sigma = U \operatorname{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_r^2, 0, \ldots, 0) U^\mathsf{T}$, and so $\mathcal{V}$ is the subspace spanned by the first r rows of U. The multivariate extension of the linear combination rule is as follows. If A is a $n \times m$ matrix, then the multivariate r.v. $Y = AX$ is distributed as though $Y \sim \mathcal{N}(A\mu, A\Sigma A^\mathsf{T})$. For further details regarding multivariate Gaussians see [Mil64].

Finally, we conclude these Gaussian preliminaries with the famous Johnson-Lindenstrauss Lemma, our main tool in this paper.

Theorem 2.4 (The Johnson Lindenstrauss transform [JL84])**.** *Fix any $0 < \eta < 1/2$. Let M be a $r \times m$ matrix whose entries are iid samples from $\mathcal{N}(0,1)$. Then $\forall x \in \mathbb{R}^m$.*

$$\mathbf{Pr}_M\left[(1-\eta)\|x\|^2 \leq \frac{1}{r}\|Mx\|^2 \leq (1+\eta)\|x\|^2\right] \geq 1 - 2\exp(-\eta^2 r/8)$$

Laplacians and edge-matrices. An undirected weighted graph $G = (V(G), E(G))$ can be represented in various ways. One representation is by the *adjacency matrix* A, where $A_{u,v} = w_{u,v}$. Another way is by the $\binom{n}{2} \times n$ *edge matrix* of the graph, E_G. We assume that the vertices of G are ordered arbitrarily, and for each pair of vertices $\{u, v\}$ where $u < v$, there exists a row in E_G. The entries of E_G are

$$\left(E_G\right)_{(\{u,v\}, x)} = \left\{\sqrt{w_{u,v}}, \text{ if } u \sim_G v \text{ and } x = u \; ; \qquad -\sqrt{w_{u,v}}, \text{ if } u \sim_G v \text{ and } x = v \; ; \qquad 0, \text{ o/w}\right\}$$

where $u \sim_G v$ denotes that (u, v) is an edge in G. Alternatively, one can represent G using the *Laplacian* of the graph $L_G = E_G^\mathsf{T} E_G$. Formally, the matrix L_G is the matrix whose diagonal entries are $(L_G)_{u,u} = \sum_{x \sim_G u} w_{x,u}$ and non diagonal entries are $(L_G)_{u,v} = -w_{u,v}$. It is simple to verify that for any x, the following equality holds: $x^\mathsf{T} L_G x = \sum_{u \sim_G v} w_{u,v}(x_u - x_v)^2$. As a corollary, if we take any nonempty $S \subsetneq V(G)$ and denote its $\{0,1\}^n$-indicator vector as $\mathbf{1}_S$, then $\mathbf{1}_S^\mathsf{T} L_G \mathbf{1}_S = \|E_G \mathbf{1}_S\|^2 = \sum_{u \in S, v \notin S} w_{u,v} = \Phi_G(S)$.

Additional notations. We denote by e_a the indicator vector of a. We denote by $e_{a,b} = e_a - e_b$. It follows that the $n \times n$ matrix $L_{a,b} = e_{a,b} e_{a,b}^\top$ is the matrix whose projection over coordinates a, b is $\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$, while every other entry is 0. We also denote $E_{a,b}$ as the $\binom{n}{2} \times n$ matrix, whose rows are all zeros except for the row indexed by the (a,b) pair, which is $e_{a,b}^\top$. Observe: $L_{a,b} = e_{a,b} e_{a,b}^\top = E_{a,b}^\top E_{a,b}$.

3 Publishing a Perturbed Laplacian

3.1 The Johnson-Lindenstrauss Algorithm

We now show that the Johnson Lindenstrauss transform preserves differential privacy. We first detail our algorithm, then analyze it.

Algorithm 1: Outputting the Laplacian of a Graph while Preserving Differential Privacy

Input: A n-node graph G, parameters: $\epsilon, \delta, \eta, \nu > 0$
Output: A Laplacian of a graph $\tilde{L}$

1 Set $r = \frac{8 \ln(2/\nu)}{\eta^2}$, and $w = \frac{\sqrt{32 r \ln(2/\delta)}}{\epsilon} \ln(4r/\delta)$
2 For every $u \neq v$, set $w_{u,v} \leftarrow \frac{w}{n} + \left(1 - \frac{w}{n}\right) w_{u,v}$.
3 Pick a matrix M of size $r \times \binom{n}{2}$, whose entries are iid samples of $\mathcal{N}(0,1)$.
4 **return** $\tilde{L} = \frac{1}{r} E_G^\top M^\top M E_G$

Algorithm 2: Approximating $\Phi_G(S)$

Input: A non empty $S \subsetneq V(G)$, parameters n, w and Laplacian $\tilde{L}$ from Algorithm 1.

return $R(S) = \frac{1}{1 - \frac{w}{n}} \left(\mathbf{1}_S^\top \tilde{L} \mathbf{1}_S - w \frac{s(n-s)}{n} \right)$

Theorem 3.1. *Algorithm 1 preserves (ϵ, δ)-differential privacy w.r.t to edge changes in G.*

Theorem 3.2. *For every $\eta, \nu > 0$ and a nonempty S of size s, Algorithm 2 gives a (η, τ, ν)-approximation for cut queries, for $\tau = O\left(s \cdot \frac{\sqrt{\ln(1/\delta) \ln(1/\nu)}}{\epsilon} \left(\ln(1/\delta) + \ln(\ln(1/\nu)/\eta^2)\right)\right)$.*

Clearly, once Algorithm 1 publishes $\tilde{L}$, any user interested in estimating $\Phi_G(S)$ for some nonempty $S \subsetneq V(G)$ can run Algorithm 2 on her own. Also, observe that w is independent of n, which we think of as large number, so we assume thoughout the proofs of both theorems that both $\frac{w}{n}, \frac{1}{w}$ are $< 1/2$. Now, the proof of Theorem 3.2 is immediate from the JL Lemma.

Proof of Theorem 3.2. Let us denote G as the input graph for Algorithm 1, and H as the graph resulting from the changes in edge-weights Algorithm 1 makes. Therefore,

$$L_H = L_{\frac{w}{n} K_n} + L_{(1 - \frac{w}{n})G} = \frac{w}{n} L_{K_n} + \left(1 - \frac{w}{n}\right) L_G$$

Fix S. The JL Lemma (Theorem 2.4) assures us that w.p. $\geq 1 - \nu$ we have

$$(1 - \eta) \mathbf{1}_S^\top L_H \mathbf{1}_S \leq \mathbf{1}_S^\top \tilde{L} \mathbf{1}_S \leq (1 + \eta) \mathbf{1}_S^\top L_H \mathbf{1}_S$$

The proof now follows from basic arithmetic and the value of w.

$$
\begin{aligned}
R(S) \quad &\leq \frac{1}{1-\frac{w}{n}}\left((1+\eta)\mathbf{1}_S^\mathsf{T} L_H \mathbf{1}_S - w\frac{s(n-s)}{n}\right) \\
&= \frac{1}{1-\frac{w}{n}}\left((1+\eta)\frac{w}{n}s(n-s) + (1+\eta)(1-\frac{w}{n})\mathbf{1}_S^\mathsf{T} L_G \mathbf{1}_S - w\frac{s(n-s)}{n}\right) \\
&\leq (1+\eta)\Phi_G(S) + \frac{1}{1-\frac{w}{n}}\eta w \cdot s = (1+\eta)\Phi_G(S) + \tau
\end{aligned}
$$

where $\tau \leq 2\eta w \cdot s$. The lower bound is obtained exactly the same way. $\square$

Comment. The guarantee of Theorem 3.2 is not to be mistaken with a weaker guarantee of providing a good approximation to *most* cut-queries. Theorem 3.2 guarantees that any set of k predetermined cuts is well-approximated by Algorithm 2, assuming Algorithm 1 sets $\nu < 1/2k$. In contrast, giving a good approximation to most cuts can be done by a very simple (and privacy preserving) algorithm: by outputting the number of edges in the graph (with small Laplacian noise). Afterall, we expect a cut to have $\frac{m}{\binom{n}{2}}s(n-s)$ edges crossing it.

We turn our attention to the proof of Theorem 3.1. We fix any two graphs G and G', which differ only on a single edge, (a,b). We think of (a,b) as an edge in G' which isn't present in G, and in the proof of Theorem 3.1, we identify G with the manipulation Algorithm 1 performs over G, and assume that the edge (a,b) is present in both graphs, only it has weight $\frac{w}{n}$ in G, and weight 1 in G'. Clearly, this analysis carries on for a smaller change, when the edge (a,b) is present in both graphs but with different weights. (Recall, we assume all edge weights are bounded by 1.)

Now, the proof follows from assuming that Algorithm 1 outputs the matrix $O = ME_G$, instead of $\tilde{L} = \frac{1}{r}O^\mathsf{T}O$. (Clearly, outputting O allows one to reconstruct $\tilde{L}$.) Observe that O is composed of r identically distributed rows: each row is created by sampling a $\binom{n}{2}$-dimensional vector Y whose entries $\sim \mathcal{N}(0,1)$, then outputting $Y^\mathsf{T}E_G$. Therefore, we prove Theorem 3.1 by showing that each row maintain (ϵ_0, δ_0)-differential privacy, for the right parameters ϵ_0, δ_0. To match standard notion, we transpose row vectors to column vectors, and compare the distributions $E_G^\mathsf{T}Y$ and $E_{G'}^\mathsf{T}Y$.

Claim 3.3. *Set $\epsilon_0 = \frac{\epsilon}{\sqrt{4r\ln(2/\delta)}}, \delta_0 = \frac{\delta}{2r}$. Then,*

$$
\forall x, \quad \mathsf{PDF}_{E_G^\mathsf{T}Y}(x) \leq \quad e^{\epsilon_0}\mathsf{PDF}_{E_{G'}^\mathsf{T}Y}(x) \tag{1}
$$

Denote $S = \{x : \mathsf{PDF}_{E_G^\mathsf{T}Y}(x) \geq e^{-\epsilon_0}\mathsf{PDF}_{E_{G'}^\mathsf{T}Y}(x)\}$. Then

$$
\mathbf{Pr}[S] \geq 1 - \delta_0 \tag{2}
$$

Proof of Theorem 3.1 based on Claim 3.3. Apply the composition theorem of [DRV10] for r iid samples each preserving (ϵ_0, δ_0)-differential privacy. $\square$

To prove Claim 3.3, we denote $X = E_G^\mathsf{T}Y$ and $X' = E_{G'}^\mathsf{T}Y$. From the preliminaries it follows that X is a multivariate Gaussian distributed according to $\mathcal{N}(0, E_G^\mathsf{T}I_{\binom{n}{2}\times\binom{n}{2}}E_G) = \mathcal{N}(0, L_G)$, and similarly, $X' \sim \mathcal{N}(0, L_{G'})$. In order to analyze the two distributions, $\mathcal{N}(0, L_G)$ and $\mathcal{N}(0, L_{G'})$, we now discuss several of the properties of L_G and $L_{G'}$, then turn to the proof of Claim 3.3.

First, it is clear from definition that the all ones vector, $\mathbf{1}$, belongs to the kernel space of E_G and $E_{G'}$, and therefore to the kernel space of L_G and $L_{G'}$. Next, we establish a simple fact.

Fact 3.4. *If G is a graph s.t. for every $u \neq v$ we have that $w_{u,v} > 0$, then $\mathbf{1}$ is the only vector in the kernel space of E_G and L_G.*

Proof. Any non-zero $x \perp \mathbf{1}$ has at least one positive coordinate and one negative coordinate, thus the non-negative sum $\|E_G x\|^2 = x^\intercal L_G x = \sum_{u \neq v} w_{u,v}(x_u - x_v)^2$ is strictly positive. $\qquad\square$

Therefore, the kernel space of both L_G and of $L_{G'}$ is exactly the 1-dimensional span of the $\mathbf{1}$ vector (for every possible outcome y of Y we have that $E_G^\intercal y \cdot \mathbf{1} = E_{G'}^\intercal y \cdot \mathbf{1} = 0$). Alternatively, both X and X' have support which is exactly $\mathcal{V} = \mathbf{1}^\perp$. Hence, we only need to prove the inequalities of Claim 3.3 for $x \in \mathcal{V}$. Secondly, observe that $L_{G'} = L_G + (1 - \frac{w}{n})L_{a,b}$. Therefore, it holds that for every $x \in \mathbb{R}^n$ we have $x^\intercal L_{G'} x = x^\intercal L_G x + (1 - \frac{w}{n})(x_a - x_b)^2 \geq x^\intercal L_G x$. In other words, $L_G \preceq L_{G'}$, a fact that yields several important corollaries.

We now introduce notation for the Singular Value Decomposition of both L_G and $L_{G'}$. We denote $E_G^\intercal = U\Sigma V^\intercal$ and $E_{G'}{}^\intercal = U'\Lambda V'^\intercal$, resulting in $L_G = U\Sigma^2 U^\intercal, L_{G'} = U'\Lambda^2 U'^\intercal, L_G^\dagger = U\Sigma^{-2}U^\intercal$ and $L_{G'}^\dagger = U'\Lambda^{-2}U'^\intercal$. We denote the singular values of L_G as $\sigma_1^2 \geq \ldots \geq \sigma_{n-1}^2 > \sigma_n^2 = 0$, and the singular values of $L_{G'}$ as $\lambda_1^2 \geq \ldots \geq \lambda_{n-1}^2 > \lambda_n^2 = 0$. Weyl's inequality allows us to deduce the following fact. Its and other facts' proofs are in Appendix A.

Fact 3.5. *Since $L_G \preceq L_{G'}$ then for every i we have that $\lambda_i^2 \geq \sigma_i^2$.*

In addition, since Algorithm 1 alters the input graphs s.t. the complete graph $\frac{w}{n}L_{K_n}$ is contained in G, then it also holds that $\frac{w}{n}L_{K_n} \preceq L_G$, and so Fact 3.5 gives that for every $1 \leq i \leq n - 1$ we have that $\sigma_i^2 \geq w = \frac{w}{n} \cdot n$. (It is simple to see that the eigenvalues of K_n are $\{n, n, \ldots, n, 0\}$.) Furthermore, as $L_{G'} = L_G + (1 - \frac{w}{n})L_{a,b}$ and the singular values of $L_{a,b}$ are $\{2, 0, 0, \ldots, 0\}$, then we have that

$$\sum_i \lambda_i^2 = tr(L_{G'}) \leq tr(L_G) + tr\left((1 - \frac{w}{n})L_{a,b}\right) \leq \sum_i \sigma_i^2 + 2$$

Another fact we can deduce from $L_G \preceq L_{G'}$, is the following.

Fact 3.6. *Since the kernels of L_G and of $L_{G'}$ are identical, then for every x it holds that $x^\intercal L_{G'}^\dagger x \leq x^\intercal L_G^\dagger x$. Symbolically, $L_G \preceq L_{G'} \Rightarrow L_{G'}^\dagger \preceq L_G^\dagger$.*

Having established the above facts, we can turn to the proof of privacy.

Proof of Claim 3.3. We first prove the upper bound in (1). As mentioned, we focus only on $x \in \mathcal{V} = \mathbf{1}^\perp$, where

$$\mathsf{PDF}_{E_G^\intercal Y}(x) = \left((2\pi)^{n-1}\tilde{\det}(L_G)\right)^{-1/2} \exp(-\frac{1}{2}x^\intercal L_G^\dagger x)$$

$$\mathsf{PDF}_{E_{G'}^\intercal Y}(x) = \left((2\pi)^{n-1}\tilde{\det}(L_{G'})\right)^{-1/2} \exp(-\frac{1}{2}x^\intercal L_{G'}^\dagger x)$$

As noted above, we have that for every x it holds that $x^\intercal L_{G'}^\dagger x \leq x^\intercal L_G^\dagger x$, so $\exp(-\frac{1}{2}x^\intercal L_G^\dagger x) \leq \exp(-\frac{1}{2}x^\intercal L_{G'}^\dagger x)$. It follows that for every x we have that $\frac{\mathsf{PDF}_{E_G^\intercal Y}(x)}{\mathsf{PDF}_{E_{G'}^\intercal Y}(x)} \leq \left(\frac{\tilde{\det}(L_{G'})}{\tilde{\det}(L_G)}\right)^{1/2} = \left(\prod_{i=1}^{n-1}\frac{\lambda_i^2}{\sigma_i^2}\right)^{1/2}$.
Denoting $\Delta_i = \lambda_i^2 - \sigma_i^2 \geq 0$, and recalling that $\sum_i \Delta_i \leq 2$ and that $\forall i, \sigma_i^2 \geq w$ it holds that

$$\frac{\mathsf{PDF}_{E_G^\intercal Y}(x)}{\mathsf{PDF}_{E_{G'}^\intercal Y}(x)} \leq \sqrt{\prod_{i=1}^{n-1}\left(1 + \frac{\Delta_i}{\sigma_i^2}\right)} \leq \exp\left(\frac{1}{2w}\sum_i \Delta_i\right) \leq e^{\frac{1}{w}} \leq e^{\frac{\epsilon}{\sqrt{4r\ln(2/\delta)}}} = e^{\epsilon_0}$$

8

We now turn to the lower bound of (2). We start with analyzing the term $x^\mathsf{T} L_G^\dagger x$ that appears in $\mathsf{PDF}_{E^\mathsf{T} Y}(x)$. Again, we emphasize that $x \in \mathcal{V}$, justifying the very first equality below.

$$
\begin{aligned}
x^\mathsf{T} L_G^\dagger x &= x^\mathsf{T} L_G^\dagger L_{G'} L_{G'}^\dagger x = x^\mathsf{T} L_G^\dagger \left(L_G + (1 - \frac{w}{n}) L_{ab} \right) L_{G'}^\dagger x \\
&= x^\mathsf{T} L_{G'}^\dagger x \; + \; (1 - \frac{w}{n}) x^\mathsf{T} L_G^\dagger L_{a,b} L_{G'}^\dagger x \\
&= x^\mathsf{T} L_{G'}^\dagger x \; + \; (1 - \frac{w}{n}) x^\mathsf{T} L_G^\dagger e_{a,b} \; \cdot \; e_{a,b}^\mathsf{T} L_{G'}^\dagger x
\end{aligned}
$$

Therefore, if we show that

$$
\mathbf{Pr}_{x \sim E_G^\mathsf{T} Y} \left[x^\mathsf{T} L_G^\dagger e_{a,b} \; \cdot \; e_{a,b}^\mathsf{T} L_{G'}^\dagger x > \frac{2}{1 - \frac{w}{n}} \epsilon_0 \right] < \delta_0 \tag{3}
$$

then it holds that w.p. $> 1 - \delta_0$ we have

$$
\frac{\mathsf{PDF}_{E_G^\mathsf{T} Y}(x)}{\mathsf{PDF}_{E_{G'}^\mathsf{T} Y}(x)} \geq 1 \cdot \exp\left(-\frac{1}{2} x^\mathsf{T} (L_G^\dagger - L_{G'}^\dagger) x \right) \geq \exp\left(-\frac{1 - \frac{w}{n}}{2} x^\mathsf{T} L_G^\dagger e_{a,b} \; \cdot \; e_{a,b}^\mathsf{T} L_{G'}^\dagger x \right) \geq e^{-\epsilon_0}
$$

which proves the lower bound of (2). We turn to proving (3).

Denote $term_1 = e_{a,b}^\mathsf{T} L_G^\dagger x$ and $term_2 = e_{a,b}^\mathsf{T} L_{G'}^\dagger x$. Since $x = E_G^\mathsf{T} y$ where $y \sim Y$ then $term_i$ is distributed like $vec_i^\mathsf{T} Y$ where $vec_1 = E_G L_G^\dagger e_{a,b}$ and $vec_2 = E_G L_{G'}^\dagger e_{a,b}$. The naïve bound, $\|vec_1\| \leq \|E_G\| \|L_G^\dagger\| \|e_{a,b}\|$ gives a bound on the size of vec_1 which is dependent on the ratio $\frac{\sigma_1}{\sigma_{n-1}^2}$. We can improve the bound, on both $\|vec_1\|$ and $\|vec_2\|$, using the SVD of E_G and $E_{G'}$.

$$
\begin{aligned}
\|vec_1\| &= \|E_G L_G^\dagger e_{a,b}\| = \|V \Sigma U^\mathsf{T} U \Sigma^{-2} U^\mathsf{T} e_{a,b}\| = \|V \Sigma^{-1} U^\mathsf{T} e_{a,b}\| \\
&\leq \|V\| \|\Sigma^{-1}\| \|U\| \|e_{a,b}\| = 1 \cdot \sigma_{n-1}^{-1} \cdot 1 \cdot \sqrt{2} = \frac{\sqrt{2}}{\sqrt{w}} \\
\|vec_2\| &= \|E_G L_{G'}^\dagger e_{a,b}\| = \|(E_{G'} - (1 - \frac{w}{n}) E_{a,b}) L_{G'}^\dagger e_{a,b}\| < \|E_{G'} L_{G'}^\dagger e_{a,b}\| + \|E_{a,b} L_{G'}^\dagger e_{a,b}\| \\
&\stackrel{(*)}{\leq} \lambda_{n-1}^{-1} \cdot \sqrt{2} + \|E_{a,b} L_{G'}^\dagger e_{a,b}\| \stackrel{(**)}{=} \frac{\sqrt{2}}{\sqrt{w}} + e_{a,b}^\mathsf{T} L_{G'}^\dagger e_{a,b} \\
&\leq \frac{\sqrt{2}}{\sqrt{w}} + \frac{2}{w} = \frac{\sqrt{2}}{\sqrt{w}} \left(1 + \frac{\sqrt{2}}{\sqrt{w}} \right)
\end{aligned}
$$

where the bound in $(*)$ is derived just like in vec_1 (using $E_{G'} L_{G'}^\dagger e_{a,b} = V' \Lambda U'^\mathsf{T} U' \Lambda^{-2} U'^\mathsf{T} e_{a,b}$), and the equality in $(**)$ follows from the fact that all coordinates in the vector $E_{a,b} L_{G'}^\dagger e_{a,b}$ are zero, except for the coordinate indexed by the (a, b) pair.

We now use the fact that $term_1$ and $term_2$ are both linear combinations of i.i.d $\mathcal{N}(0, 1)$ random variables. Therefore for $i = 1, 2$ we have that $term_i \sim \mathcal{N}(0, \|vec_i\|^2)$ so $\mathbf{Pr}[|term_i| > \sqrt{\log(2/\delta_0)} \|vec_i\|] \leq e^{-\frac{\|vec_i\|^2 \log(2/\delta_0)}{\|vec_i\|^2}} < \frac{\delta_0}{2}$. It follows that w.p $> 1 - \delta_0$ both $|term_1| < \sqrt{\log(2/\delta_0)} \sqrt{\frac{2}{w}}$ and $|term_2| \leq \sqrt{\log(2/\delta_0)} \sqrt{\frac{4}{w}}$, so $term_1 \cdot term_2 \leq \sqrt{8} \log(2/\delta_0)/w$. Plugging in the value of w, we have that $\mathbf{Pr}[term_1 \cdot term_2 \leq 2\epsilon_0] \geq 1 - \delta_0$ which concludes the proof of (3) and of Claim 3.3. $\square$

3.2 Discussion and Comparison with Other Algorithms

Recently, Gupta et al [GRU12] have also considered the problem of answering cut-queries while preserving differential privacy, examining both an iterative database construction approach (e.g., based on the multiplicative-weights method) and a randomized-response approach. Here, we compare this and other methods to our algorithm. We compare them along several axes: the dependence on n and s (number of vertices in G and in S resp.), the dependence on ϵ, and the dependence on k – the number of queries answered by the mechanism. Other parameters are omitted. The bottom line is that for a long non-adaptive query sequence, our approach dominates in the case that $s = o(n)$. The results are summarized in Table 1.

Note, comparing the dependence on k for interactive and non-interactive mechanisms is not straight-forward. In general, non-interactive mechanisms are more desirable than interactive mechanisms, because interactive mechanisms require a central authority that serves as the only way users can interact with the database. However, interactive mechanisms can answer k *adaptively chosen* queries. In order for non-interactive mechanisms to do so, they have to answer correctly on $\min\{\exp(O(k)), 2^n\}$ queries. This is why outputting a sanitized database is often considered a harder task than interactively answering user queries. We therefore compare answering k adaptively chosen queries for interactive mechanisms, and k *predetermined* queries for non-interactive mechanism.

3.2.1 Our Algorithm

Clearly, our algorithm is non-interactive. As such, if we wish to answer correctly w.h.p. a set of k *predetermined* queries, we set $\nu' = \nu/k$, and deduce that the amount of noise added to each query is $O(s\sqrt{\log(k)}/\epsilon)$. So, if we wish to answer all 2^n cut queries correctly, our noise is set to $\tilde{O}(s\sqrt{n}/\epsilon)$. An interesting observation is that in such a case we aim to answer all 2^n queries, we generate a iid normal matrix of size $r \times n$ where $r > n$. Therefore, we now apply the JL transform to *increase* the dimensionality of the problem rather than decreasing it. This clearly sets privacy preserving apart from all other applications of the JL transform.

In addition, we comment that our algorithm can be implemented in a *distributed* fashion, where node i repeats the following procedure r times (where r is the number of rows in the matrix picked by Algorithm 1): First, i picks $n - i - 1$ iid samples from $\mathcal{N}(0, 1)$ and sends the j-th sample, x_j, to node $i + j$. Once node i receives $i - 1$ values from nodes $1, 2, \ldots, i - 1$, it outputs the weighted sum $\sum_{j \neq i} (-1)^{\{j < i\}} x_j \left(\sqrt{\frac{w}{n}} + w_{i,j}(1 - \sqrt{\frac{w}{n}}) \right)$ (where $(-1)^{\{j<i\}}$ denotes -1 if $j < i$, or 1 otherwise).

3.2.2 Naïvely Adding Laplace Noise

The most basic of all differentially private mechanisms is the classical Laplace mechanism which is interactive. A user poses a cut-query S and the mechanism replies with $\Phi_G(S) + Lap(0, \epsilon^{-1})$ (since the global sensitivity of cut-queries is 1). The composition theorem of [DRV10] assures us that for k queries we preserve $(O(\sqrt{k}\epsilon), \delta)$-privacy. As a result, the mechanism completely obfuscates the true answer if $k \geq n^4$ and even for $k = n^2$ has noise proportional to n/ϵ.

3.2.3 The Randomized Response Mechanism

The "Randomized Response" algorithm perturbs the edges of a graph in a way that allows us to publish the result and still preserve privacy. Given G, the Randomized Response algorithm

constructs a weighted graph H where for every $u, v \in V(G)$, the weight of the edge (u, v) in H, denoted $w'_{u,v}$, is chosen independently to be either 1 or -1. Each edge picks its weight independently, s.t. $\mathbf{Pr}[w'_{u,v} = 1] = \frac{1 + \epsilon w_{u,v}}{2}$ and $\mathbf{Pr}[w'_{u,v} = -1] = \frac{1 - \epsilon w_{u,v}}{2}$. Clearly, this algorithm maintains ϵ-differential edge privacy: two neighboring graphs differ on a single edge, (a, b), and obviously

$$\mathbf{Pr}[w'_{a,b} = 1 \mid w_{a,b} = 1] \leq (1 + \epsilon)\mathbf{Pr}[w'_{a,b} = 1 \mid w_{a,b} = 0]$$

In addition, it is also evident that for every nonempty $S \subsetneq V(G)$, we have that $\mathbf{E}[\sum_{u \in S, v \in \bar{S}} w'_{u,v}] = \epsilon \sum_{u \in S, v \notin S} w_{u,v} = \epsilon \Phi_G(S)$, yet the variance of this r.v. is $\Omega(s(n - s))$. Therefore, a classical Hoeffding-type bound gives that for any nonempty $S \subsetneq V(G)$ we have that for every $0 < \nu < 1/2$,

$$\mathbf{Pr}\left[\left| \frac{1}{\epsilon} \sum_{u \in S, v \in \bar{S}} w'_{u,v} - \Phi_G(S) \right| > \frac{\sqrt{2\log(1/\nu)s(n-s)}}{\epsilon} \right] \leq 2\nu$$

Observe that while $\sqrt{s(n-s)}$ is a comparable with s when $s = \Omega(n)$, there are cuts (namely, cuts with $s = O(1)$) where $\sqrt{\frac{n-s}{s}} = \Omega(\sqrt{n})$. More generally, the additive noise of Randomized Response is a factor $\sqrt{n/s}$ worse than our algorithm. We comment that the Randomized Response algorithm can also be performed in a distributed fashion, and in contrast to our algorithm, it has no multiplicative error. In addition, the above analysis holds for any linear combination of edge, not just the $s(n-s)$ potential edges that cross the $(S, \bar{S})$ cut. So given $E' \subset E(G)$ it is possible to approximate $\sum_{e \in E'} w_e$ up to $\pm \frac{\sqrt{|E'|\log(1/\nu)}}{\epsilon}$ w.p. $\geq 1 - 2\nu$. In particular, for queries regarding an (S, T)-cut (where S, T are two disjoint subsets of vertices) we can estimate the error up to $\pm \frac{\sqrt{|S||T|\log(1/\nu)}}{\epsilon}$. We also comment that the version of Randomized Response presented here differs slightly from the version of [GRU12]. In particular, it is possible to address their concern regarding outputting a sanitized graph with non-negative weights by an affine transformation taking $\{-1, 1\} \to \{0, 1\}$.

3.2.4 Exponential Mechanism / BLR

The exponential mechanism [MT07, BLR08] is a non-interactive privacy preserving mechanism, which is typically intractable. To implement it for cut-queries one needs to (a) specify a range of potential outputs and (b) give a scoring function over potential outputs s.t. a good output's score is much higher than all bad outputs' scores.

One such set of potential outputs is derived from edge-sparsifiers. Given a graph G we say that H is an edge-sparsifier for G if for any nonempty $S \subsetneq V(G)$ it holds that $\Phi_H(S) \in (1 \pm \eta)\Phi_G(S)$. There's a rich literature on sparsifiers (see [BK96, ST04, SS08]), and the current best known construction [BSS09] gives a (weighted) sparsifier with $O(n/\eta^2)$ edges with all edge-weights $\leq$ poly(n). By describing every edge's two endpoints and weight, we have that such edge-sprasifiers can be described using $O(n \log(n))$ bits (omitting dependence on η). Thus, the set of all sparsifiers is bounded above by $\exp(O(n \log(n)))$. Given an input graph G and a weighted graph H, we can score H using $q(G, H) = \max_S \{\min_{\alpha: |\alpha - 1| \leq \eta} |\Phi_H(S)/\alpha - \Phi_G(S)|\}$. Observe that if we change G to a neighboring graph G', then the score changes by at most 1.

Putting it all together, we have that given input G the exponential mechanism gives a score of $e^{-\epsilon q(G,H)/2}$ to each possible output. The edge-sparsifier of G gets score of 1, whereas every graph

11

with $q(G, H) > \tau$ gets a score of $e^{-\epsilon\tau/2}$. So if we wish to claim we output a graph whose error is $> \tau$ w.p. at most ν, then we need to set $\exp(n\log(n) - \epsilon\tau/2) \le \nu$. It follows that τ is proportional to $n\log(n)/\epsilon$. Note however that the additive error of this mechanism is independent of the number of queries it answers correctly.

We comment that even though we managed to find a range of size $2^{O(n\log(n))}$, it is possible to show that the range of the mechanism has to be $2^{\Omega(n)}$. (Fix $\alpha < 1/2$ and think of a set of inputs $\mathcal{G}$ where each $G \in \mathcal{G}$ has $n/2$ vertices with degree n^α and $n/2$ vertices with degree $n^{2\alpha}$. Preserving all cuts of size 1 up to $(1 \pm \eta)$ requires our output to have vertices of degree $> (1 - \eta)n^{2\alpha}$ and vertices of degree $< (1 + \eta)n^\alpha$. Therefore, by representing vertices of high- and low-degree using a binary vector, there exists an injective mapping of balanced $\{0, 1\}^n$-vectors onto the set of potential outputs.) Thus, unless one can devise a scoring function of lower sensitivity, the exponential mechanism is bounded to have additive error proportional to n/ϵ.

3.2.5 The Multiplicative Weights Mechanism

The very elegant Multiplicative Weights mechanism of Hardt and Rothblum [HR10] can be adapted as well for answering cut queries. In the Multiplicative Weights mechanism, a database is represented by a histogram over all N "types" of individuals that exist in a certain universe. In our case, each pair of vertices is a type, and each entry in the database is an edge detailing its weight. Thus, $N = \binom{n}{2}$ and the database length $= |E|$,[3] and each query S corresponds to taking a dot-product between this histogram the $\binom{n}{2}$-length binary vector indicating the edges that cross the cut. Plugging these parameters into the main theorem of [HR10], we get an adaptive mechanism that answers k queries with additive noise of $\tilde{O}(\sqrt{|E|}\log(k)/\epsilon)$.

We should mention that the Multiplicative Weights mechanism, in contrast to ours, always answers correctly with no multiplicative error and can deal with k adaptively chosen queries. Furthermore, it allows one to answer any linear query on the edges, not just cut-queries and in particular answer (S, T)-cut queries. However, its additive error is bigger than ours, and should we choose to set $k = 2^n$ (meaning, answering all cut-queries) then its additive error becomes $\tilde{O}(n\sqrt{|E|}/\epsilon)$ (in contrast to our $O(s\sqrt{n}/\epsilon)$).

Gupta et al [GRU12] have improved on the bounds on the Multiplicative Weights mechanism by generalizing it as a "Iterative Database Construction" mechanism, and providing a tighter analysis of it. In particular, they have reduced the dependency on ϵ to $1/\sqrt{\epsilon}$. Overall, their additive error is $\tilde{O}(\sqrt{|E|\log(k)}/\sqrt{\epsilon})$, which for the case of all cut-queries is $\tilde{O}(\sqrt{n|E|}/\epsilon)$.

4 Publishing a Covariance Matrix

4.1 The Algorithm

In this section, we are concerned with the question of allowing users to estimate the covariance of a given sample data along an arbitrary direction x. We think of our input as a $n \times d$ matrix A, and we maintain privacy w.r.t to changing the coordinates of a single row s.t. a vector v of size 1 is added to $A_{(i)}$. We now detail our algorithm for publishing the covariance matrix of A. Observe that in addition to the variance, we can output $\mu = \frac{1}{n}A^\mathsf{T}\mathbf{1}$, the mean of all samples in A, in a differentially

---

[3] Observe that it is not possible to assume $|E| = O(n)$ using sparsifiers, because sparsifiers output a *weighted* graph with edge-weights $O(n)$. Since the Multiplicative Weights mechanism views the database as a histogram the overall resolution of the problem remains roughly n^2 in the worst case.

Method	Additive Error for any k	Additive Error for all Cuts	Multi-plicative Error?	Inter-active?	Tract-able?	Comments
Laplace Noise [DMNS06]	$O(\sqrt{k}/\epsilon)$	$O(2^{n/2}\epsilon)$	✗	✓	✓	
Randomized Response	$O(\sqrt{sn\log(k)}/\epsilon)$	$O(n\sqrt{s}/\epsilon)$	✗	✗	✓	Can be distributed; answers (S,T)-cut queries
Exponential Mechanism [MT07, BLR08]	$O(n\log(n)/\epsilon)$	$O(n\log(n)/\epsilon)$	✓	✗	✗	Error ind. of k
MW [HR10] IDC [GRU12]	$\tilde{O}(\sqrt{|E|}\log(k)/\epsilon)$ $\tilde{O}(\sqrt{|E|\log(k)}/\epsilon)$	$\tilde{O}(n\sqrt{|E|}/\epsilon)$ $\tilde{O}(\sqrt{n|E|}/\epsilon)$	✗	✓	✓	Answers (S,T)-cut queries
JL	$O(s\sqrt{\log(k)}/\epsilon)$	$\tilde{O}(s\sqrt{n})/\epsilon)$	✓	✗	✓	Can be distributed

Table 1: Comparison between mechanisms for answering cut-queries. ϵ – privacy parameter; n and $|E|$ – number of vertices and edges resp.; s – number of vertices in a query; k – number of queries.

private manner by adding random Gaussian noise. (We merely output $\tilde{\mu} = \mu + \mathcal{N}(0, \frac{4\log(1/\delta)}{n^2\epsilon^2} I_{d\times d})$.) We denote by $I_{n\times d}$ the $n \times d$ matrix whose main diagonal has 1 in each coordinate and all other coordinates are 0.

Algorithm 3: Outputting a Covariance Matrix while Preserving Differential Privacy

Input: A $n \times d$ matrix A. Parameters $\epsilon, \delta, \eta, \nu > 0$.

1 Set $r = \frac{8\ln(2/\nu)}{\eta^2}$ and $w = \frac{16\sqrt{r\ln(2/\delta)}}{\epsilon} \ln(16r/\delta)$.

2 Subtract the mean from A by computing $A \leftarrow A - \frac{1}{n}\mathbf{1}\mathbf{1}^T A$.

3 Compute the SVD of $A = U\Sigma V^\mathsf{T}$.

4 Set $A \leftarrow U(\sqrt{\Sigma^2 + w^2 I_{n\times d}})V^\mathsf{T}$.

5 Pick a matrix M of size $r \times n$ whose entries are iid samples of $\mathcal{N}(0,1)$.

6 **return** $\tilde{C} = \frac{1}{r}A^\mathsf{T}M^\mathsf{T}MA$.

Algorithm 4: Approximating $\Phi_A(x)$

Input: A unit-length vector x, parameter w and a Covariance matrix $\tilde{C}$ from Algorithm 3.

return $R(x) = x^\mathsf{T}\tilde{C}x - w^2$.

Theorem 4.1. *Algorithm 3 preserves (ϵ, δ)-differential privacy.*

Theorem 4.2. *Algorithm 4 is a (η, τ, ν)-approximation for directional variance queries, where* $\tau = O\left(\frac{\ln(1/\delta)\ln(1/\nu)}{\epsilon^2\eta} \ln^2\left(\frac{\ln(1/\nu)}{\delta\eta^2}\right)\right)$.

Proof of Theorem 4.2. Again, the proof is immediate from the JL Lemma, and straight-forward arithmetics give that for every x w.p. $\geq 1 - \nu$ we have that

$$(1 - \eta)\Phi_A(x) - \eta w^2 \leq R(x) \leq (1 + \eta)\Phi_A(x) + \eta w^2$$

so $\tau = \eta w^2$. $\qquad\square$

Comment. We wish to clarify that Theorem 4.2 does *not* mean that we publish a matrix $\tilde{C}$ which is a low-rank approximation to $A^\mathsf{T}A$. It is also not a matrix on which one can compute an approximated PCA of A, *even if we set $\nu = 1/\text{poly}(d)$*. The matrix $\tilde{C}$ should be thought of as a "test-matrix" – if you believe A has high directional variance along some direction x then you can test your hypothesis on $\tilde{C}$ and (w.h.p) get the good approximated answer. However, we do not guarantee that the singular values of $A^\mathsf{T}A$ and of $\tilde{C}$ are close or that the eigenvectors of $A^\mathsf{T}A$ and $\tilde{C}$ are comparable. (See discussion in Section 5.)

Proof of Theorem 4.1. Fix two neighboring A and A'. We often refer to the gap matrix $A' - A$ as E. Observe, E is a rank-1 matrix, which we denote as the outer-product $E = e_i v^\mathsf{T}$ (e_i is the indicator vector of row i and v is a vector of norm 1). As such, the singular values of E are exactly $\{1, 0, \ldots, 0\}$.[4]

The proof of the theorem is composed of two stages. The first stage is the simpler one. We ignore step 4 of Algorithm 3 (shifting the singular values), and work under the premise that both A and A' have singular values no less than w. In the second stage we denote B and B' as the results of applying step 4 to A and A' resp., and show what adaptations are needed to make the proof follow through.

Stage 1.
We assume step 4 was not applied, and all singular values of A and A' are at least w.

As in the proof of Theorem 3.1, the proof follows from the assumption that Algorithm 3 outputs $O^\mathsf{T} = A^\mathsf{T}M$ (which clearly allows us to reconstruct $\tilde{C} = \frac{1}{r}O^\mathsf{T}O$). Again O^T is composed of r columns each is an iid sample from $A^\mathsf{T}Y$ where $Y \sim \mathcal{N}(0, I_{n\times n})$. We now give the analogous claim to Claim 3.3.

Claim 4.3. *Fix $\epsilon_0 = \frac{\epsilon}{\sqrt{4r\ln(2/\delta)}}$ and $\delta_0 = \frac{\delta}{2r}$. Denote $S = \{x : e^{-\epsilon_0}\mathsf{PDF}_{A'^\mathsf{T}Y}(x) \leq PDF_{A^\mathsf{T}Y}(x) \leq e^{\epsilon_0}\mathsf{PDF}_{A'^\mathsf{T}Y}(x)\}$. Then $\mathbf{Pr}[S] \geq 1 - \delta_0$.*

Again, the composition theorem of [DRV10] along with the choice of r gives that overall we preserve (ϵ, δ)-differential privacy. □

Proof of Claim 4.3. The proof mimics the proof of Claim 3.3, but there are two subtle differences. First, the problem is simpler notation-wise, because A and A' both have full rank due to Algorithm 3. Secondly, the problem becomes more complicated and requires we use some heavier machinery, because the singular values of A' aren't necessarily bigger than the singular values of A. Details follow.

First, let us formally define the PDF of the two distributions. Again, we apply the fact that $A^\mathsf{T}Y$ and $A'^\mathsf{T}Y$ are linear transformations of $\mathcal{N}(0, I_{n\times n})$.

$$\mathsf{PDF}_{A^\mathsf{T}Y}(x) = \frac{1}{\sqrt{(2\pi)^d \det(A^\mathsf{T}A)}} \exp(-\frac{1}{2}x^\mathsf{T}(A^\mathsf{T}A)^{-1}x)$$

$$\mathsf{PDF}_{A'^\mathsf{T}Y}(x) = \frac{1}{\sqrt{(2\pi)^d \det(A'^\mathsf{T}A')}} \exp(-\frac{1}{2}x^\mathsf{T}(A'^\mathsf{T}A')^{-1}x)$$

---

[4]For convenience, we ignore the part of the algorithm that subtracts the mean of the rows of A. Observe that if $E = A - A'$ then after subtracting the mean from each row, the difference between the two matrices is $\tilde{e_i}^\mathsf{T}v$ where $\tilde{e_i}$ is simply subtracting $1/n$ from each coordinate of e_i. Since $\|\tilde{e_i}\| < \|e_i\|$, this has no effect on the analysis.

Our proof proceeds as follows. First, we show

$$e^{-\epsilon_0/2} \leq \sqrt{\frac{\det(A'^{\mathsf{T}}A')}{\det(A^{\mathsf{T}}A)}} \leq e^{\epsilon_0/2} \tag{4}$$

Then we show that no matter whether we sample x from $A^{\mathsf{T}}Y$ or from $A'^{\mathsf{T}}Y$, we have that

$$\mathbf{Pr}_x \left[\frac{1}{2} \left| x^{\mathsf{T}} \left((A^{\mathsf{T}}A)^{-1} - (A'^{\mathsf{T}}A')^{-1} \right) x \right| \geq \epsilon_0/2 \right] \leq \delta_0 \tag{5}$$

Clearly, combining both (4) and (5) proves the claim.

Let us prove (4). Denote the SVD of $A = U\Sigma V^{\mathsf{T}}$ and $A' = U'\Lambda V'^{\mathsf{T}}$, where the singular values of A are $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d > 0$ and the singular values of A' are $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d > 0$. Therefore we have $A^{\mathsf{T}}A = V\Sigma^2 V^{\mathsf{T}}$, $A'^{\mathsf{T}}A' = V'\Lambda^2 V'^{\mathsf{T}}$ and also $(A^{\mathsf{T}}A)^{-1} = V\Sigma^{-2}V^{\mathsf{T}}$, $(A'^{\mathsf{T}}A')^{-1} = V'\Lambda^{-2}V'^{\mathsf{T}}$. Thus $\det(A^{\mathsf{T}}A) = \prod_{i=1}^{d} \sigma_i^2$ and $\det(A'^{\mathsf{T}}A') = \prod_{i=1}^{d} \lambda_i^2$.

This time, in order to bound the gap $\sum_i (\lambda_i^2 - \sigma_i^2)/\sigma_i^2$ it isn't sufficient to use the trace of the matrices. Instead, we invoke an application of Lindskii's theorem (Theorem 9.4 in [Bha07]).

Fact 4.4 (Linskii)**.** *For every k and every $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ we have that*

$$\sum_{j=1}^{k} \lambda_{i_j} \leq \sum_{j=1}^{k} \sigma_{i_j} + \sum_{i=1}^{k} sv_i(E)$$

where $\{sv_i(E)\}_{i=1}^{n}$ are the singular values of E sorted in a descending order.

As a corollary, because E has only 1 non-zero singular value, we denote $Big = \{i : \lambda_i > \sigma_i\}$ and deduce that $\sum_{i \in Big} \lambda_i - \sigma_i \leq 1$. Similarly, since the singular values of E and of $(-E)$ are the same, we have that $\sum_{i \notin Big} \sigma_i - \lambda_i \leq 1$. Using this, proving (4) is straight-forward:

$$\sqrt{\prod_i \frac{\lambda_i^2}{\sigma_i^2}} \leq \prod_{i \in Big} \left(1 + \frac{\lambda_i - \sigma_i}{\sigma_i} \right) \leq \exp \left(\frac{1}{w} \sum_{i \in Big} \lambda_i - \sigma_i \right) \leq e^{w^{-1}} \leq e^{\epsilon_0/2}$$

and similarly, $\sqrt{\prod_i \frac{\sigma_i^2}{\lambda_i^2}} \leq e^{\epsilon_0/2}$.

We turn to proving (5). We start with the following derivation.

$$\begin{aligned}
x^{\mathsf{T}}(A^{\mathsf{T}}A)^{-1}x - x^{\mathsf{T}}(A'^{\mathsf{T}}A')^{-1}x &= x^{\mathsf{T}}(A^{\mathsf{T}}A)^{-1}(A'^{\mathsf{T}}A')(A'^{\mathsf{T}}A')^{-1}x - x^{\mathsf{T}}(A'^{\mathsf{T}}A')^{-1}x = \\
&= x^{\mathsf{T}}(A^{\mathsf{T}}A)^{-1}((A+E)^{\mathsf{T}}(A+E))(A'^{\mathsf{T}}A')^{-1}x - x^{\mathsf{T}}(A'^{\mathsf{T}}A')^{-1}x \\
&= x^{\mathsf{T}}(A^{\mathsf{T}}A)^{-1}(A^{\mathsf{T}}E + E^{\mathsf{T}}A')(A'^{\mathsf{T}}A')^{-1}x
\end{aligned}$$

and using the SVD and denoting $E = e_i v^{\mathsf{T}}$, we get

$$\begin{aligned}
x^{\mathsf{T}}(A^{\mathsf{T}}A)^{-1}x - x^{\mathsf{T}}(A'^{\mathsf{T}}A')^{-1}x &= x^{\mathsf{T}} \left(V\Sigma^{-1}U^{\mathsf{T}} \right) e_i \cdot v^{\mathsf{T}} \left(V'\Lambda^{-2}V'^{\mathsf{T}} \right) x \\
&\quad + x^{\mathsf{T}} \left(V\Sigma^{-2}V^{\mathsf{T}} \right) v \cdot e_i^{\mathsf{T}} \left(U'\Lambda^{-1}V'^{\mathsf{T}} \right) x
\end{aligned}$$

So now, assume x is sampled from $A^{\mathsf{T}}Y$. (The case of $A'^{\mathsf{T}}Y$ is symmetric. In fact, the names A and A' are interchangeable.) That is, assume we've sampled y from $Y \sim \mathcal{N}(0, I_{n \times n})$ and we have

$x = A^\mathsf{T} y = V\Sigma U^\mathsf{T} y$ and equivalently $x = (A'^\mathsf{T} - E^\mathsf{T})y = V'\Lambda U'^\mathsf{T} y - ve_i^\mathsf{T} y$. The above calculation shows that

$$\left| x^\mathsf{T}(A^\mathsf{T} A)^{-1} x - x^\mathsf{T}(A'^\mathsf{T} A')^{-1} x \right| \le term_1 \cdot term_2 + term_3 \cdot term_4$$

where for $i = 1, 2, 3, 4$ we have $term_i = |vec_i \cdot y|$ and

$$vec_1 = U\Sigma V^\mathsf{T} V\Sigma^{-1} U e_i = e_i, \qquad\qquad \text{so } \|vec_1\| = 1$$

$$vec_2 = U'\Lambda^{-1} V'^\mathsf{T} v - e_i v^\mathsf{T} V'\Lambda^{-2} V'^\mathsf{T} v, \qquad\qquad \text{so } \|vec_2\| \le \frac{1}{\lambda_d} + \frac{1}{\lambda_d^2}$$

$$vec_3 = U\Sigma^{-1} V^\mathsf{T} v, \qquad\qquad \text{so } \|vec_3\| \le \frac{1}{\sigma_d}$$

$$vec_4 = e_i - e_i v^\mathsf{T} V'\Lambda^{-1} U'^\mathsf{T} e_i, \qquad\qquad \text{so } \|vec_4\| \le 1 + \frac{1}{\lambda_d}$$

Recall that all singular values, both of A and A', are greater than w and that $vec_i \cdot y \sim \mathcal{N}(0, \|vec_i\|^2)$, so w.p. $\ge 1 - \delta_0$ we have that for every i it holds that $term_i \le \sqrt{\ln(4/\delta_0)}\|vec_i\|$ so

$$\left| x^\mathsf{T}(A^\mathsf{T} A)^{-1} x - x^\mathsf{T}(A'^\mathsf{T} A')^{-1} x \right| \le 2\left(\frac{1}{w} + \frac{1}{w^2}\right)\ln(4/\delta_0) \le \frac{4\ln(4/\delta_0)}{w} \le \epsilon_0$$

this concludes the proof in our first stage.

Stage 2.

We assume step 4 was applied, and denote $B = U(\sqrt{\Sigma^2 + w^2 I})V^\mathsf{T}$ and $B' = U'(\sqrt{\Lambda^2 + w^2 I})V'^\mathsf{T}$. We denote the singular values of B and B' as $\sigma_1^B \ge \sigma_2^B \ge \ldots \ge \sigma_d^B$ and $\lambda_1^B \ge \lambda_2^B \ge \ldots \ge \lambda_d^B$ resp. Observe that by definition, for every i we have $(\sigma_i^B)^2 = \sigma_i^2 + w^2$ and $(\lambda_i^B)^2 = \lambda_i^2 + w^2$.

Again, we assume we output $O^\mathsf{T} = B^\mathsf{T} Y$, and compare $X = B^\mathsf{T} Y$ to $X' = B'^\mathsf{T} Y$. The theorem merely requires Claim 4.3 to hold, and they, in turn, depend on the following two conditions.

$$e^{-\epsilon_0/2} \le \sqrt{\frac{\det(B'^\mathsf{T} B')}{\det(B^\mathsf{T} B)}} \le e^{\epsilon_0/2} \tag{6}$$

$$\mathbf{Pr}_x \left[\frac{1}{2} \left| x^\mathsf{T} \left((B^\mathsf{T} B)^{-1} - (B'^\mathsf{T} B')^{-1}\right) x \right| \ge \epsilon_0/2 \right] \le \delta_0 \tag{7}$$

The second stage deals with the problem that now, the gap $\Delta = B' - B$ is not necessarily a rank-1 matrix. However, what we show is that all stages in the proof of Claim 4.3 either rely on the singular values or can be written as the sum of a few rank-1 matrix multiplications.

The easier part is to claim that Eq. (6) holds. The analysis is a simple variation on the proof of Eq. (4). Fact 4.4 still holds for the singular values of A and A'. Observe that $\lambda_i^B > \sigma_i^B$ iff $\lambda_i > \sigma_i$. And so we have

$$\sqrt{\prod_i \frac{(\lambda_i^B)^2}{(\sigma_i^B)^2}} \le \sqrt{\prod_{i \in Big} \frac{\lambda_i^2 + w^2}{\sigma_i^2 + w^2}} \le \sqrt{\prod_{i \in Big} \frac{\lambda_i^2}{\sigma_i^2}}$$

and the remainder of the proof follows.

16

We now turn to proving Eq. (7). We start with an observation regarding $A'^\mathsf{T}A$ and $B'^\mathsf{T}B'$.

$$A'^\mathsf{T}A' = (A+E)^\mathsf{T}(A+E) = A^\mathsf{T}A + A'^\mathsf{T}E + E^\mathsf{T}A$$
$$B^\mathsf{T}B = V(\Sigma^2 + w^2 I)V^\mathsf{T} = V\Sigma^2 V^\mathsf{T} + w^2 I = A^\mathsf{T}A + w^2 I$$
$$B'^\mathsf{T}B' = V'(\Lambda^2 + w^2 I)V'^\mathsf{T} = A'^\mathsf{T}A' + w^2 I$$
$$\Rightarrow \ B'^\mathsf{T}B' - B^\mathsf{T}B = A'^\mathsf{T}E + E^\mathsf{T}A$$

Now we can follow the same outline as in the proof of (5). Fix x, then:

$$
\begin{aligned}
x^\mathsf{T}(B^\mathsf{T}B)^{-1}x - x^\mathsf{T}(B'^\mathsf{T}B')^{-1}x \ &= x^\mathsf{T}(B^\mathsf{T}B)^{-1}(B'^\mathsf{T}B')(B'^\mathsf{T}B')^{-1}x - x^\mathsf{T}(B'^\mathsf{T}B')^{-1}x = \\
&= x^\mathsf{T}(B^\mathsf{T}B)^{-1}\left[B^\mathsf{T}B + A'^\mathsf{T}E + E^\mathsf{T}A\right](B'^\mathsf{T}B')^{-1}x - x^\mathsf{T}(B'^\mathsf{T}B')^{-1}x \\
&= x^\mathsf{T}(B^\mathsf{T}B)^{-1}\left[A'^\mathsf{T}E + E^\mathsf{T}A\right](B'^\mathsf{T}B')^{-1}x \\
&= x^\mathsf{T}(B^\mathsf{T}B)^{-1}(A^\mathsf{T} + E^\mathsf{T})e_i \ \cdot \ v^\mathsf{T}(B'^\mathsf{T}B')^{-1}x \\
&+ x^\mathsf{T}(B^\mathsf{T}B)^{-1}v \ \cdot \ e_i^\mathsf{T}\left(A' - E\right)(B'^\mathsf{T}B')^{-1}x
\end{aligned}
$$

It is straight-forward to see that the i-th spectral values of $(B^\mathsf{T}B)^{-1}A$ is $\frac{\sigma_i}{\sigma_i^2 + w^2} \leq \frac{1}{\sqrt{\sigma_i^2 + w^2}} \leq 1/w$, and similarly for the spectral values of $(B'^\mathsf{T}B')^{-1}A'$. We now proceed as before and partition the above sum into multiplications of pairs of terms where $term_i \leq |vec_i \cdot y|$, and y is sampled from $\mathcal{N}(0, I_{n\times n})$ and $x = B^\mathsf{T}y$:

$$
\begin{aligned}
x^\mathsf{T}(B^\mathsf{T}B)^{-1}x - x^\mathsf{T}(B'^\mathsf{T}B')^{-1}x \ &= y^\mathsf{T}\left[B(B^\mathsf{T}B)^{-1}(A^\mathsf{T} + E^\mathsf{T})e_i\right] \ \cdot \ \left[v^\mathsf{T}(B'^\mathsf{T}B')^{-1}B^\mathsf{T}\right]y \\
&+ y^\mathsf{T}\left[B(B^\mathsf{T}B)^{-1}v\right] \ \cdot \ \left[e_i^\mathsf{T}\left(A' - E\right)(B'^\mathsf{T}B')^{-1}B^\mathsf{T}\right]y
\end{aligned}
$$

Lastly, we need to bound all terms that contain the multiplication $(B'^\mathsf{T}B')^{-1}B^\mathsf{T}y$ in comparison to $(B'^\mathsf{T}B')^{-1}B'^\mathsf{T}y = B'^\dagger y$. For instance, take the $term = |vec^\mathsf{T}y|$ for $vec^\mathsf{T} = e_i^\mathsf{T}\left(A' - E\right)(B'^\mathsf{T}B')^{-1}B^\mathsf{T}$, and define it as $vec^\mathsf{T} = z^\mathsf{T}B^\mathsf{T}$. We can only bound $\|Bz\|$ using $\sigma_1^B/(\lambda_d^B)^2$, whereas we can bound $\|B'z\|$ with $1/\lambda_d^B < 1/w$. In contrast to before, we do not use the fact that $B^\mathsf{T}y = (B' - \Delta)^\mathsf{T}y$. Instead, we make the following derivations.

First, we observe that for every vector z we have that $\|B'z\| \geq \|A'z\|$ and $\|B'z\| \geq w\|z\|$. Using the fact that $B^\mathsf{T}B - B'^\mathsf{T}B' = -A'^\mathsf{T}E - E^\mathsf{T}A$, a simple derivation gives that $\|Bz\|^2 \leq (\|B'z\| + \|z\|)^2 \leq \left(1 + \frac{1}{w}\right)^2\|B'z\|^2$, and vice-versa. So if y is s.t. $\frac{|z^\mathsf{T}B^\mathsf{T}y|}{(1+\frac{1}{w})\|B'z\|} > Threshold$ then $\frac{|z^\mathsf{T}B^\mathsf{T}y|}{\|Bz\|} > Threshold$. Observe that $z^\mathsf{T}B^\mathsf{T}y$ is distributed like $\mathcal{N}(0, \|Bz\|^2) = \|Bz\|\mathcal{N}(0,1)$, and so we have that for every $\delta' > 0$

$$
\begin{aligned}
\mathbf{Pr}\left[|z^\mathsf{T}B^\mathsf{T}y| \geq \sqrt{\log(1/\delta')}\left(1 + \frac{1}{w}\right)\|B'z\|\right] &= \mathbf{Pr}\left[\left(\left(1 + \frac{1}{w}\right)\|B'z\|\right)^{-1}|z^\mathsf{T}B^\mathsf{T}y| \geq \sqrt{\log(1/\delta')}\right] \\
&\leq \mathbf{Pr}\left[(\|Bz\|)^{-1}|z^\mathsf{T}B^\mathsf{T}y| \geq \sqrt{\log(1/\delta')}\right] \leq \delta' \qquad \square
\end{aligned}
$$

Corollary. Using the definitions of r and w as in Algorithm 3 – the proof of Theorem 4.1 actually shows that in the case that A is a matrix with all singular values $\geq w$, then the following simple algorithm preserves (ϵ, δ)-differential privacy: pick a random $r \times n$ matrix M whose entries are iid normal Gaussians, and output $O = MA$. Furthermore, observe that if σ_d, the least singular value

17

of A, is bigger than, say, $10w$, then one can release $\sigma_d + Lap(1/\epsilon)$ then release $O = MA$. In such a case, users know that for any unit vector x w.p. $\geq 1 - \nu$ it holds that $\frac{1}{r}\|Ox\|^2 \leq (1 \pm \eta)\|Ax\|^2$.

Comment. Comparing Algorithms 1 and 3, we have that in $L_G = E_G^{\mathsf{T}}E_G$ we "translate" the spectral values by w, and in $A^{\mathsf{T}}A$ we "translated" the spectral values by w^2. This is an artifact of the ability to directly compare the spectal values of L_G and $L_{G'}$ in the first analysis, whereas in the second analysis we compare the spectral values of A and A' (vs. $A^{\mathsf{T}}A$ and $A'^{\mathsf{T}}A'$). This is why the noise bounds in the general case are $\tilde{O}(1/\epsilon\eta)$ times worse than for graphs.

4.2 Comparison with Other Algorithms

To the best of our knowledge, no previous work has studied the problem of preserving the variance of A in the same formulation as us. We deal with a scenario where users pose the directions on which they wish to find the variance of A. Other algorithms, that publish the PCA or a low-rank approximation of A without compromising privacy (see Section 1.1), provide users with specific directions and variances. These works are not comparable with our algorithm, as they give a different utility guarantee. For example, low-rank approximations aim at nullifying the projection of A in certain directions.

Here, we compare our method to the Laplace mechanism, the Multiplicative Weights mechanism and Randomized Response. The bottom line is clear: our method allows one to answer directional variance queries with additive noise which is independent of the given input. Other methods require we add random noise that depends on the size of the matrix, assuming we answer polynomially many queries.

Our notation is as follows. n denotes the number of rows in the matrix (number of individuals in the data), d denotes the number of columns in the matrix, and we assume each entry is at most 1. As before, ϵ denotes the privacy parameter and k denotes the number of queries. Observe that we (again) compare k *predetermined* queries for non-interactive mechanisms with k *adaptively chosen* queries for interactive ones. The remaining parameters are omitted from this comparison. Results are summarized in Table 2.

4.2.1 Our Algorithm

Our algorithm's utility is computed simply by plugging in $\nu = O(1/k)$ to Theorem 4.2, which gives a utility bound of $O(\log(k)/\epsilon^2)$.

4.2.2 Naïvely Adding Laplace Noise

Again, the simplest alternative is to answer each directional-variance query with $\Phi_x(A)+Lap(0, \epsilon^{-1})$. The composition theorem of [DRV10] assures us that for k queries we preserve $(O(\sqrt{k}\epsilon), \delta)$-differential privacy.

4.2.3 Randomized Response

We now consider a Randomized Response mechanism, similar to the Randomized Response mechanism of [GRU12]. We wish to output a noisy version of $A^{\mathsf{T}}A$, by adding some iid random noise to each entry of $A^{\mathsf{T}}A$. Since we call two matrices neighbors if they differ only on a single row, denote v as the difference vector on that row. It is simple to see that by adding v to some row in A, each entry in $A^{\mathsf{T}}A$ can change by at most $\|v\|_1$. Recall that we require $\|v\|_2 = 1$ and so $\|v\|_1 \leq \sqrt{d}$.

Method	Additive Error	Multi-plicative Error?	Inter-active?	Tract-able?
Laplace Noise [DMNS06]	$O(\sqrt{k}/\epsilon)$	✗	✓	✓
Randomized Re-sponse	$\tilde{O}(\sqrt{d\log(k)}/\epsilon)$	✗	✗	✓
MW [HR10] IDC [GRU12]	$\tilde{O}(d\sqrt{n}\log(k)/\epsilon)$ $\tilde{O}(d\sqrt{n\log(k)}/\epsilon)$	✗	✓	✓
JL	$O(\log(k)/\epsilon^2)$	✓	✗	✓

Table 2: Comparison between mechanisms for answering directional variance queries.

Therefore, we have that in order to preserve (ϵ, δ)-differential privacy, it is enough to add a random Gaussian noise of $\mathcal{N}(0, \frac{d\log(d)}{\epsilon^2})$ to each of the d^2 entries of $A^{\mathsf{T}}A$.

Next we give the utility guarantee of the Randomized Response scheme. Fix any unit length vector x. We think of the matrix we output as $A^{\mathsf{T}}A + N$, where N is a matrix of iid samples from $\mathcal{N}(0, \frac{d\log(d)}{\epsilon^2})$. Therefore, in direction x, we add to the true answer a random noise distributed like $x^{\mathsf{T}}Nx \sim \mathcal{N}(0, \left(\sum_{i,j} x_i^2 x_j^2\right) \frac{d\log(d)}{\epsilon^2}) = \mathcal{N}(0, \frac{d\log(d)}{\epsilon^2})$. So w.h.p the noise we add is within factor of $\tilde{O}(\sqrt{d}/\epsilon)$ for each query, and for k queries it is within factor of $\tilde{O}(\sqrt{d\log(k)}/\epsilon)$.

4.2.4 The Multiplicative Weights Mechanism

It is not straight-forward to adapt the Multiplicative Weights mechanism to answer directional variance queries. We represent $A^{\mathsf{T}}A$ as a histogram over its d^2 entries (so the size of the "universe" is $N = d^2$), but it is not simple to estimate what is the equivalent of number of individuals in this representation. We chose to take the pessimistic bound of nd^2, since this is the L_1 bound on the sum of entries in $A^{\mathsf{T}}A$, but we comment this is a highly pessimistic bound. It is fairly likely that the number of individuals in this representation can be set to only $O(d^2)$.

Plugging these parameters into the utility bounds of the Multiplicative Weights mechanism, we get a utility bound of $\tilde{O}(d\sqrt{n}\log(k)/\epsilon)$. Plugging them into the improved bounds of the IDC mechanism, we get $\tilde{O}(d\sqrt{n\log(k)}/\epsilon)$. Observe that even if replace the pessimistic bound of nd^2 with just d^2, these bounds depend on d.

5 Discussion and Open Problems

The fact that the JL transform preserves differential privacy is likely to have more theoretical and practical applications than the ones detailed in this paper. Below we detail a few of the open questions we find most compelling.

Error depedency on r. Our algorithm projects the edge-matrix of a given graph on r random directions, then publishes these projections. The value of r determines the probability we give a good approximation to a given cut-query, and provided that we wish to give a good approximation to all cut-queries, our analysis requires us to set $r = \Omega(n)$. But is it just an artifact of the analysis? Could it be that a better analysis gives a better bound on r? It turns out that the answer is "no". In fact, the direction on which we project the data now have high correlation with the published Laplacian. We demonstrate this with an example.

Assume our graph is composed of a single perfect matching between $2n$ nodes, where node i is matched with node $n + i$. Focus on a single random projection – it is chosen by picking $\binom{2n}{2}$ iid random values $x_{i,j} \sim \mathcal{N}(0, 1)$, and for the ease of exposition imagine that the values of the edges in the matching are picked first, then the values of all other pairs of vertices. Now, if we pick the value $x_{i,n+i}$ for the $\langle i, n+i \rangle$ edge, then node i is assigned $x_{i,n+i}$ while node $n + i$ is assigned $-x_{i,n+i}$. So regardless of the sign of $x_{i,n+i}$, *exactly one* of the two nodes $\{i, n+i\}$ is assigned the positive value $|x_{i,n+i}|$ and exactly one is assigned the negative value $-|x_{i,n+i}|$. Define S as the set of n nodes that are assigned the positive values and $\bar{S}$ as the set of n nodes that are assigned the negative values. The sum of weight crossing the $(S, \bar{S})$-cut is distributed like $(X + \frac{w}{n}Y)^2$ where $X = \sum_i |x_{i,n+i}|$ and $Y = \sum_i \sum_{j \neq n+i} x_{i,j}$. Indeed, Y is the sum of $n(n-1)$ random normal iid Gaussians, but X is the sum of n *absolute values* of Gaussians. So w.h.p. both X and Y are proportional to n. Therefore, in the direction of this particular random projection we estimate the $(S, \bar{S})$-cut as $\Omega((n \pm w)^2) = \Omega(n^2)$ rather than $O(n)$. (If X was distributed like the sum of n iid normal Gaussians, then the estimation would be proportional to $(\sqrt{n})^2 = n$.)

Assuming that the remaining $r - 1$ projections estimate the cut as $O(n)$, then by averaging over all r random projections our estimation of the $(S, \bar{S})$-cut is $\omega(n)$, as long as $r = o(n)$.

Error amplification or error detection. Having established that we do err on some cuts, we pose the question of error amplification. Can we introduce some error-correction scheme to the problem without increasing r significantly? Error amplification without increasing r will allow us to keep the additive error fairly small. One can view $\tilde{L}$ as a coding of answers to all 2^n cut-queries which is guaranteed to have at least $1 - \nu$ fraction of the code correct, in the sense that we get a (η, τ)-approximation to the true cut-query answer. As such, it is tempting to try some self-correcting scheme – like adding a random vector x to the vector $\mathbf{1}_S$, then finding the estimation to $x^\intercal L_G x$ and $(\mathbf{1}_s + x)^\intercal L_G x$ and inferring $\mathbf{1}_S^\intercal L_G \mathbf{1}_S$. We were unable to prove such scheme works due to the dot-product problem (see next paragraph) and to query dependencies.

A related question is of error detection: can we tell whether $\tilde{L}$ gives a good estimation to a cut query or not? One potential avenue is to utilize the trivial guess for $\Phi_G(S)$ – the expected value $\frac{m}{\binom{n}{2}} s(n - s)$ (we can release m via the Laplace mechanism). We believe this question is related to the problem of estimating the variance of $\{\Phi_G(S) \ : \ |S| = s\}$.

Edges between S and T. Our work assures utility only for cut-queries. It gives no utility guarantees for queries regarding $E(S, T)$, the set of edges connecting two disjoint vertex-subsets S and T. The reason is that it is possible to devise a graph where both $E(S, \bar{S})$ and $E(T, \bar{T})$ are large whereas $E(S, T)$ is fairly small. When $E(S, \bar{S})$ and $E(T, \bar{T})$ are big, the *multiplicative error* η given to both quantities might add too much noise to an estimation of $E(S, T)$.

The problem relates to the dot-product estimation of the JL transform. It is a classical result that if M is a distance-preserving matrix and u and v are two vectors s.t. $\|M(u + v)\|^2 \approx \|u + v\|^2$ and $\|M(u - v)\|^2 \approx \|u - v\|^2$ then it is possible to bound the difference $|Mu \cdot Mv - u \cdot v|$. But this bound is a function of $\|u\|$ and $\|v\|$, which in our case translates to a bound that depends on $\|E_G \mathbf{1}_S\|$ and $\|E_G \mathbf{1}_T\|$, both vectors of potentially large norms.

Other Versions of JL. The analysis in this works deals with the most basic JL transform, using normal Gaussians. We believe that qualitatively the same results should apply for other versions of the JL transform (e.g., with entries taken in $U_{[-1,1]}$). However, we are not certain whether the same results hold for *sparse* transforms (see [DKS10]).

Low rank approximation of a given matrix. The work of [HR12] gives a differentially private

algorithm that outputs a low-rank approximation of a given matrix A, while adding additive error $> \min\{\sqrt{d}, \sqrt{n}\}$. Our work, which introduces much smaller noise (independent of n and d), does not have such guarantees. Our algorithm could potentially be integrated into theirs. In particular, their algorithm is composed of two stages, and our technique greatly improves the first of the two. The crux of the second stage lies in devising a way to preserve differential privacy when multiplying a given (non-private) X with a private database A without introducing too large of an additive noise. Matrix multiplication via random projections might be such a way.

Integration with the Multiplicative Weights mechanism. When the interactive Multiplicative Weights mechanism is given a user's query, it considers two possible alternatives: answering according to a synthetic database, or answering according to the Laplace mechanism. It chooses the latter alternative only when the two answers are far apart. Its utility guarantees rely on applying the Laplace mechanism only a bounded number of times. An interesting approach might be to add a third alternative, of answering according to the perturbed Laplacian we output. Hopefully, if most updates can be "charged" to answers provided by the perturbed Laplacian, it will allow us to improve privacy parameters (noise dependency on n).

References

[AC06]   Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 557–563, New York, NY, USA, 2006. ACM.

[BBV06]   Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. Kernels as features: On kernels, margins, and low-dimensional mappings. *Machine Learning*, 65(1):79–94, 2006.

[BDDW08]   Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2008.

[BDMN05]   Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '05, pages 128–138, New York, NY, USA, 2005. ACM.

[Bha07]   R. Bhatia. *Perturbation Bounds for Matrix Eigenvalues (Classics in Applied Mathematics)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.

[BK96]   András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{o}(n^2)$ time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 47–55, New York, NY, USA, 1996. ACM.

[BLR08]   A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 609–618. ACM, 2008.

[Bou85]   J Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985.

[BSS09]   Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan spar-sifiers. In *STOC*, pages 255–262, 2009.

[CDM⁺05]   Shuchi Chawla, Cynthia Dwork, Frank Mcsherry, Adam Smith, and Larry Joseph Stockmeyer. Toward privacy in public databases. In *In TCC*, pages 363–385, 2005.

[DKM⁺06]   Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

[DKS10]   Anirban Dasgupta, Ravi Kumar, and Tamás Sarlos. A sparse johnson: Lindenstrauss transform. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 341–350, New York, NY, USA, 2010. ACM.

[DMNS06]   Cynthia Dwork, Frank Mcsherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *In Proceedings of the 3rd Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

[DN03]   Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

[DRV10]   Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60, 2010.

[DS10]   Cynthia Dwork and Adam Smith. Differential privacy for statistics: What we know and what we want to learn. *Journal of Privacy and Confidentiality*, 1(2):2, 2010.

[Dwo11]   Cynthia Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011.

[GHRU11]   Anupam Gupta, Moritz Hardt, Aaron Roth, and Jonathan Ullman. Privately releasing conjunctions and the statistical query barrier. In *STOC*, pages 803–812, 2011.

[GRU12]   Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In *TCC*, pages 339–356, 2012.

[HJ90]   Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[HLM10]   Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. *CoRR*, abs/1012.4763, 2010.

[HLMJ09]   Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, pages 169–178, 2009.

[HMT11]   Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[HR10]   M. Hardt and G.N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 61–70. IEEE, 2010.

[HR12]   Moritz Hardt and Aaron Roth. Beating randomized response on incoherent matrices. In *STOC*, 2012.

[IM98]   Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. pages 604–613, 1998.

[JL84]   W. Johnson and J. Lindenstauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 1984.

[Kle97]   Jon M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. pages 599–608, 1997.

[KRSY11]   Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *PVLDB*, 4(11):1146–1157, 2011.

[LLR94]   N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, pages 577–591, Washington, DC, USA, 1994. IEEE Computer Society.

[Mil64]   K.S. Miller. *Multidimensional Gaussian distributions*. SIAM series in applied mathematics. Wiley, 1964.

[MM09]   Frank McSherry and Ilya Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *KDD*, pages 627–636, 2009.

[MT07]   Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[NRS07]   K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing*, pages 75–84. ACM, 2007. Full version in: http://www.cse.psu.edu/~asmith/pubs/NRS07.

[PRT+98]   Christos H. Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, S. Vempala, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. pages 159–168. ACM press, 1998.

[RR10]   A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 765–774. ACM, 2010.

[Sar06]   Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pages 143–152, 2006.

[Sch00]   Leonard J. Schulman. Clustering for edge-cost minimization (extended abstract). In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00, pages 547–555, New York, NY, USA, 2000. ACM.

[SS08]   Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *STOC*, pages 563–568, 2008.

[ST04]   Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, pages 81–90, 2004.

[Vem05]   S.S. Vempala. *The Random Projection Method.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 2005.

[War65]   Stanley L. Warner. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *Journal of the American Statistical Association*, 60(309):63+, March 1965.

A   Facts from Linear Algebra

Below we prove the various facts from linear algebra that were mentioned in the body of the paper. We add the proofs, yet we comment that they are not new. In fact, existing literature [HJ90, Bha07] have documented proofs of the general theorems from which our facts are derived. Throughout this section, we denote the i-th eigenvalue (resp. the i-th singular value) of a given matrix M in a descending order, assuming all eigenvalues are real, as $ev_i(M)$ (resp. as $sv_i(M)$).

Proving Fact 3.5. The fact uses the max-min characterization of the singular values of a matrix.

Theorem A.1 (Courant-Fischer Min-Max Principle)**.** *For every matrix A and every $1 \leq i \leq n$, the i-th singular value of A satisfies:*

$$sv_i(A) = \max_{S:\dim(S)=i} \ \min_{x \in S: \ \|x\|=1} \langle Ax, x \rangle$$

Claim A.2 (Weyl Inequality)**.** *Let A and B be positive semidefinite matrices s.t. the matrix $E = B - A$ satisfies $x^{\mathsf{T}}Ex \geq 0$ for every x. Then for every $1 \leq i \leq n$ it holds that*

$$sv_i(A) \leq sv_i(B)$$

Proof. Let S_A be the i-dimensional subspace s.t. $sv_i(A) = \min_{x \in S_A: \ \|x\|=1} \langle Ax, x \rangle$. For every $x \in S_A$ we have that
$$\langle Ax, x \rangle = \langle Ax, x \rangle + 0 \leq \langle Ax, x \rangle + \langle Ex, x \rangle = \langle Bx, x \rangle$$
so $sv_i(A) \leq \min_{x \in S_A: \ \|x\|=1} \langle Bx, x \rangle$. Thus $sv_i(B) = \max_S \min_{x \in S: \ \|x\|=1} \langle Bx, x \rangle \geq sv_i(A)$. $\quad\square$

Fact 3.5 is a direct application of Claim A.2 to L_G and $L_{G'}$.

Proving Fact 3.6. The proof builds on the following two claims.

Claim A.3. *Let A be a positive-semidefinite matrix. If $x^{\mathsf{T}}Ax \geq x^{\mathsf{T}}x$ for every $x \in (Ker(A))^{\perp}$ then it also holds that $x^{\mathsf{T}}A^{\dagger}x \leq x^{\mathsf{T}}x$ for every $x \in (Ker(A))^{\perp}$.*

Proof. Denote the SVD of $A = V\Sigma^2 V^{\mathsf{T}} = \sum_{i=1}^{r} \sigma_i^2 v_i v_i^{\mathsf{T}}$, where v_i is the i-th column of V. Fix $x \in (Ker(A))^{\perp}$ and observe that x is span by the same r vectors $\{v_1, v_2, \ldots, v_r\}$, so we can write $x = \sum_{i=1}^{r} \alpha_i v_i$. Denote $y = V\Sigma^{-1}V^{\mathsf{T}}x = \sum_{i=1}^{r} \sigma_i^{-1} v_i v_i^{\mathsf{T}}x$. We have that $y = \sum_{i=1}^{r} \alpha_i \sigma_i^{-1} v_i$ so $y \in (Ker(A))^{\perp}$. Therefore $y^{\mathsf{T}}Ay \geq y^{\mathsf{T}}y$, but $y^{\mathsf{T}}y = x^{\mathsf{T}}A^{\dagger}x$ and $y^{\mathsf{T}}Ay = x^{\mathsf{T}}x$. $\quad\square$

Claim A.4. *Let A and B be two positive-semidefinite matrices s.t. $Ker(A) = Ker(B)$. Then if for every x we have that $x^\mathsf{T} A x \le x^\mathsf{T} B x$ then $x^\mathsf{T} A^\dagger x \ge x^\mathsf{T} B^\dagger x$.*

Proof. We denote the SVD $A = V\Sigma^2 V^\mathsf{T}$ and $B = W\Pi^2 W^\mathsf{T}$. Because we can split any vector x into the direct sum $x = x_0 + x_\perp$ where $x_0 \in Ker(A) = Ker(B)$ and $x_\perp \in (Ker(A))^\perp$, and since we have that the required inequality holds trivially for x_0, then we need to show it holds for $x_\perp$. Given any $z \in (Ker(A))^\perp$, set $y = V\Sigma^{-1}V^\mathsf{T}z$. We know that $y^\mathsf{T}Ay \le y^\mathsf{T}By$, and therefore

$$z^\mathsf{T}z = y^\mathsf{T}Ay \le y^\mathsf{T}By = z^\mathsf{T}\left(V\Sigma^{-1}V^\mathsf{T}W\Pi^2 W^\mathsf{T}V\Sigma^{-1}V^\mathsf{T}\right)z \overset{\text{def}}{=} z^\mathsf{T}Cz$$

The above proves that C is a positive semidefinite matrix whose kernel is exactly $Ker(A) = Ker(B)$, and so it follows from Claim A.3 that $z^\mathsf{T}z \ge z^\mathsf{T}C^\dagger z$. Let $I|_{Ker(C)^\perp}$ be the matrix which nullifies every element in $Ker(C)$, yet operates like the identity on $(Ker(C))^\perp$. One can easily check that $C^\dagger = V\Sigma V^\mathsf{T}W\Pi^{-2}W^\mathsf{T}V\Sigma V^\mathsf{T}$ by verifying that indeed $C^\dagger C = CC^\dagger = I|_{Ker(C)^\perp}$. So now, given x we denote $z = V\Sigma^{-1}V^\mathsf{T}x$ and apply the above to deduce $x^\mathsf{T}B^\dagger x = z^\mathsf{T}C^\dagger z \le z^\mathsf{T}z = x^\mathsf{T}A^\dagger x$. $\qquad\square$

Fact 3.6 is a direct application of Claim A.4 to L_G and $L_{G'}$.

Proving Fact 4.4. Much like Claim A.2 follows from the Courant-Fischer Min-Max principle, Lindskii's theorem follows from a generalization of this principle.

Theorem A.5 (Wielandt's Min-Max Principle.). *Let A be a $n \times n$ symmetric matrix. Then for every k and every k indices $1 \le i_1 < i_2 < \ldots < i_k \le n$ we have that*

$$\sum_{j=1}^{k} ev_{i_j}(A) = \max_{\substack{S_1 \subset S_2 \subset \ldots \subset S_k \\ \dim(S_j) = i_j}} \min_{\substack{x_j \in S_j: \\ x_j \ orthonormal}} \sum_{j=1}^{k}\langle Ax_j, x_j\rangle$$

Claim A.6 (Linskii's theorem.). *Let A and B be a $n \times n$ symmetric matrix. Denote $E = B - A$. Then for every k and every k indices $1 \le i_1 < i_2 < \ldots < i_k \le n$ we have that*

$$\sum_{j=1}^{k} ev_{i_j}(B) \le \sum_{j=1}^{k} ev_{i_j}(A) + \sum_{i=1}^{k} ev_i(E)$$

Proof. Fix $i_1 < i_2 < \ldots < i_k$ and let $T_1, T_2, \ldots, T_k$ the subspaces for which

$$\sum_{j=1}^{k} ev_{i_j}(B) = \min_{\substack{x_j \in T_j: \\ x_j \ orthonormal}} \sum_{j=1}^{k}\langle Bx_j, x_j\rangle$$

For every $v_1, v_2, \ldots, v_k$ orthonormal we have that $\sum_{j=1}^{k}\langle Bv_j, v_j\rangle = \sum_{j=1}^{k}\langle Av_j, v_j\rangle + \sum_{j=1}^{k}\langle Ev_j, v_j\rangle \le \sum_{j=1}^{k}\langle Av_j, v_j\rangle + \sum_{i=1}^{k} ev_i(E)$, so

$$\sum_{j=1}^{k} ev_{i_j}(B) \le \min_{\substack{x_j \in T_j: \\ x_j \ orthonormal}} \sum_{j=1}^{k}\langle Ax_j, x_j\rangle + \sum_{i=1}^{k} ev_i(E)$$

and clearly

$$\sum_{j=1}^{k} ev_{i_j}(A) = \max_{\substack{S_1 \subset S_2 \subset \ldots \subset S_k \\ \dim(S_j)=i_j}} \min_{\substack{x_j \in S_j: \\ x_j \text{ orthonormal}}} \sum_{j=1}^{k} \langle Ax_j, x_j \rangle \geq \min_{\substack{x_j \in T_j: \\ x_j \text{ orthonormal}}} \sum_{j=1}^{k} \langle Ax_j, x_j \rangle \qquad \square$$

Now, Fact 4.4 follows from Claim A.6, and from the following observation of Weilandt.[5] Given a $m \times n$ matrix M, the matrix $N = \begin{pmatrix} 0 & M \\ M^{\mathsf{T}} & 0 \end{pmatrix}$ is symmetric and has eigenvalues which are (in descending order) $\{sv_1(A), sv_2(A), \ldots, sv_m(A), 0, 0, \ldots, 0, -sv_m(A), -sv_{m-1}(A), \ldots, -sv_1(A)\}$.

---

[5]We thank Moritz Hardt for bringing this observation to our attention.