On Differentially Private Graph Sparsification and Applications

Raman Arora
Johns Hopkins University
arora@cs.jhu.edu

Jalaj Upadhyay
Rutgers University
jalaj.kumar.upadhyay@gmail.com

Abstract

In this paper, we study private sparsification of graphs. In particular, we give an algorithm that given an input graph, returns a sparse graph which approximates the spectrum of the input graph while ensuring differential privacy. This allows one to solve many graph problems privately yet efficiently and accurately. This is exemplified with application of the proposed meta-algorithm to graph algorithms for privately answering cut-queries, as well as practical algorithms for computing MAX-CUT and SPARSEST-CUT with better accuracy than previously known. We also give an efficient private algorithm to learn Laplacian eigenmap on a graph.

1 Introduction

Data from social and communication networks have become a rich source to gain useful insights into the social, behavioral, and information sciences. Such data is naturally modeled as observations on a graph, and encodes rich, fine-grained, and structured information. At the same time, due to the seamless nature of data acquisition, often collected through personal devices, the individual information content in network data is often highly sensitive. This raises valid privacy concerns pertaining the analysis and release of such data. We address these concerns in this paper by presenting a novel algorithm that can be used to publish a succinct differentially private representation of network data with minimal degradation in accuracy for various graph related tasks.

There are several notions of differential privacy one can consider in the setting described above. Depending on privacy requirements, one can consider edge level privacy that renders two graphs that differ in a single edge as in-distinguishable based on the algorithm's output; this is the setting studied in many recent works [9, 19, 25, 54]. Alternatively, one can require node-level privacy which preserves privacy of each node, which has been the focus in [10, 11, 28, 44]. In this paper, we consider settings where nodes are known public entities and the edges represent sensitive or private events and attributes relating two nodes.

In particular, we consider the following notion of differential privacy. We say that an algorithm that takes a graph as input and returns a graph, is differentially private if given any two graphs that differ in a single edge by weight at most one [1], the output of the algorithm does not change by much (see Section 3 for a formal definition).

Given the ubiquitous nature of the problem, several recent works have studied various graph problems within the framework of differential privacy. These include the works on differentially private algorithms for computing degree distribution [28, 27, 44], on subgraph counting [40, 45], on private MIN-CUT [22] and on estimating parameters of generative graphical models [39]. Each of the works referenced above present algorithms that are tailor-made for a specific graph problems. We, on the

[1] This is the standard neighboring relation used in Blocki et al. [9] (and references therein). The underlying reason for considering this privacy notion is as follows: since we do not restrict the weight on the graph, presence or absence of an edge, as required in standard edge-level privacy, can change the output drastically.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

other hand, are interested in understanding the problem more generally. We pose the following question: given an input graph, can we efficiently generate a succinct yet private representation that allows us to simultaneously solve multiple graph-related tasks accurately?

Two popular representations of graphs that succinctly capture several graph properties include *spectral sparsification* and *cut sparsification*. Spectral sparsification [52] is the following problem: given a graph $\mathcal{G}$ with n nodes, output a subgraph $\widetilde{\mathcal{G}}$ of $\mathcal{G}$ such that $(1-\varepsilon)L_{\mathcal{G}} \preceq L_{\widetilde{\mathcal{G}}} \preceq (1+\varepsilon)L_{\mathcal{G}}$, i.e.,

$$\forall x \in \mathbb{R}^n, \quad (1-\varepsilon)x^\top L_{\mathcal{G}} x \leq x^\top L_{\widetilde{\mathcal{G}}} x \leq (1+\varepsilon)x^\top L_{\mathcal{G}} x, \tag{1}$$

where $L_{\mathcal{G}}$ is the Laplacian of the graph $\mathcal{G}$ (see Definition 1). This is a generalization of cut sparsification [8], where x is restricted to binary vector. Spectral sparsification of a graph is a fundamental problem that has found application in randomized linear algebra [13, 33], graph problems [29], linear programming [34], and mathematics [37, 51].

There are many non-private algorithms for computing spectral sparsification of graphs [2, 7, 35, 36, 50, 52]. However, to the best of our knowledge, there is no prior work on differentially private graph sparsification. This paper initiates this study by formally stating the goal of private graph sparsification and presents the first algorithm for efficiently computing a graph sparsification. The main contributions of this paper are as follows.

1. We show that differential privacy is not achievable under the traditional notion of spectral sparsification of graphs since the output itself may reveal information about the edges. Furthermore, we put forth an alternate but a well-posed formulation of differentially private graph sparsification problem.
2. We give an efficient algorithm that outputs a private sparse graph with $O(n/\varepsilon^2)$ edges. Since our output is a graph and preserves the spectrum of the input graph, we can solve many graph related combinatorial problems efficiently while preserving differential privacy.

The works most closely related to that of ours are that of Blocki et al. [9], Dwork et al. [19], and Gupta et al. [23]. Blocki et al. [9] and Dwork et al. [19] give an algorithm that returns a symmetric matrix that may not correspond to a graph but can be used to answer cut queries accurately while Gupta et al. [23] output a private graph that approximates cut functions, but cannot approximate spectral properties. Our algorithm for computing private sparse graph not only solves the two problems simultaneously but also improves upon both of these works.

Spectral sparsification of graphs finds many applications including, but not limited to, spectral clustering, heat kernel computations, separators, etc. Since differential privacy is preserved under post-processing, our result can be used in these applications to get a private algorithm for these tasks (see Table 1 in Section 4 for more details). On top of these improvements, we can leverage the fact that our output is a graph that approximates the spectrum of the input graph to efficiently compute *Laplacian eigenmap*, a useful technique used in manifold learning (see Section 4 for more details).

2 Preliminaries and Notations

The central object of interest in this paper is the Laplacian of a graph.

Definition 1 (Laplacian of graph)**.** Let $\mathcal{G}$ be an undirected graph with n vertices, m edges, and edge weights w_e. Consider an arbitrary orientation of edges of $\mathcal{G}$, and let $E_{\mathcal{G}}$ be the signed edge adjacency matrix of $\mathcal{G}$ given by

$$(E_{\mathcal{G}})_{e,v} := \begin{cases} +\sqrt{w_e} & \text{if } v \text{ is } e\text{'s head} \\ -\sqrt{w_e} & \text{if } v \text{ is } e\text{'s tail} \\ 0 & \text{otherwise} \end{cases}.$$

The *Laplacian* of $\mathcal{G}$ is defined as $L_{\mathcal{G}} = E_{\mathcal{G}}^\top E_{\mathcal{G}}$. Equivalently, $L_{\mathcal{G}} = D_{\mathcal{G}} - A_{\mathcal{G}}$, where $A_{\mathcal{G}}$ is the adjacency matrix and $D_{\mathcal{G}}$ is the degree matrix. One can verify that $L_{\mathcal{G}}\mathbf{1} = \mathbf{1}^\top L_{\mathcal{G}} = \mathbf{0}$, where $\mathbf{1}$ denotes all 1 vector and $\mathbf{0}$ denotes all 0 vector.

For a vertex $u \in V$, let $\delta_u \in \{0,1\}^n$ denote the row vector with 1 only at the u-th coordinate. Lets represent the edges of the graph with vectors $b_1, \ldots, b_m \in \{-1, 0, 1\}^n$ such that $b_e = \delta_u - \delta_v$ where the edge e connects $u, v \in V$. Then, $L_{\mathcal{G}} = \sum_{e \in E_{\mathcal{G}}} b_e^\top b_e$. The spectral sparsification of a

graph can be casted as the following linear algebraic problem: given a sparsity parameter s and row vectors $b_1, \ldots, b_m$, find a set of scalars $\tau_1, \ldots, \tau_m$ such that $|\{\tau_i : \tau_i \neq 0\}| \leq s$ and $(1-\varepsilon)L_{\mathcal{G}} \preceq \sum_{e \in E_{\mathcal{G}}} \tau_e b_e^\top b_e \preceq (1+\varepsilon)L_{\mathcal{G}}$. For any graph $\mathcal{G}$ with edge-adjacency matrix $E_{\mathcal{G}}$, the *effective resistance* (also known as leverage score) of the edge $e_i \in E_{\mathcal{G}}$ is defined as $\widetilde{\tau}_i := e_i^\top (E_{\mathcal{G}}^\top E_{\mathcal{G}})^\dagger e_i = e_i^\top L_{\mathcal{G}}^\dagger e_i$. It is well known that by sampling the edges (rows of $E_{\mathcal{G}}$) of $\mathcal{G}$ according to its leverage score, we obtain a graph $\widetilde{\mathcal{G}}$ such that $(1-\varepsilon)L_{\mathcal{G}} \preceq L_{\widetilde{\mathcal{G}}} \preceq (1+\varepsilon)L_{\mathcal{G}}$ (for example, see [50]).

Notations. We use the notation $(A \quad | \quad B)$ to denote the matrix formed by appending the columns of matrix B to that of matrix A. We denote by $A^\dagger$ the Moore-Penrose pseudoinverse of A and by $\|A\|_2$ its spectral norm. We use caligraphic letters to denote graphs, $V_{\mathcal{G}}$ to denote the vertices of $\mathcal{G}$ and $E_{\mathcal{G}}$ to denote the edges of $\mathcal{G}$. We drop the subscript when it is clear from the context. We use the symbol K_n to denote an n vertex complete graph and L_n to denote its Laplacian. For any $S, T \subseteq V_{\mathcal{G}}$, the size of *cut* between S and T, denoted by $\Phi_{\mathcal{G}}(S,T)$, is the sum of weight of the edges that are present between S and T. When $T = V \backslash S$, we denote the size of cut between S and $V \backslash S$ by $\Phi_{\mathcal{G}}(S)$. For a set $S \subseteq [n]$, we use the notation $\mathbf{1}_S = \sum_{i \in S} e_i$, where $\{e_1, \cdots, e_n\}$ denote the standard basis.

3 Differentially Private Graph Sparsification

We begin by noting that differential privacy is incompatible with the requirements of spectral sparsification in equation (1), because if we output a sparse subgraph of the input graph, then it will leak information about $\widetilde{O}(n)$ edges present in the graph. This motivates us to consider the following "relaxation" of the spectral sparsification problem.

Definition 2 ($(\varepsilon, \alpha, \beta, n)$-Private Spectral Sparsification)**.** Let $\mathfrak{G}$ be the set of all n vertex positive weighted graphs. We are interested in designing an efficient algorithm $\mathcal{M} : \mathfrak{G} \to \mathfrak{G}$ such that

1. (**Privacy**) For all graphs $\mathcal{G}, \mathcal{G}' \in \mathfrak{G}$ that differ in only one edge by weight 1, and all possible measurable $S \subseteq \mathfrak{G}$, $\mathsf{Pr}[\mathcal{M}(\mathcal{G}) \in S] \leq e^\alpha \mathsf{Pr}[\mathcal{M}(\mathcal{G}') \in S] + \beta$.

2. (**Sparsity**) $\mathcal{M}(\mathcal{G})$ has at most $\widetilde{O}(n)$ edges, and

3. (**Spectral approximation**) $\widetilde{\mathcal{G}} \leftarrow \mathcal{M}(\mathcal{G})$ satisfies $(\kappa L_{\mathcal{G}} - \zeta L_n) \preceq L_{\widetilde{\mathcal{G}}} \preceq \eta L_{\mathcal{G}} + \xi L_n$ where functions η, ξ, κ and ζ dependent on input parameters $n, \varepsilon, \alpha, \beta$.

The function ζ and ξ can be seen as the distortion we are willing to accept to preserve privacy. That is, we would like ζ and ξ to be as small as possible. Informally, we view adding L_n as introducing plausible deniability pertaining to the presence of an edge in the output; it could be coming from either $\mathcal{G}$ or the complete graph.

The choice of L_n is arbitrary and for simplicity. Our choice of L_n is motivated by the fact that an n vertex unweighted complete graph is the same irrespective of the input graph once the number of vertices is fixed. We can instead state item 3 by replacing L_n by Laplacian of any graph whose edges are independent of any function of $\mathcal{G}$. For example, we can use a d-regular expander instead of K_n; however, this would not change our result as one can prove that a d-regular expander are spectral sparsification of K_n. In the definition, we consider *edge level privacy*, a setting studied in many recent works [9, 19, 25, 54][2] .

We first enumerate why previous work do not suffice. The two works most related work to ours is by Blocki et al. [9] and Upadhyay [54] used random projection on the edge-adjacency matrix of graph $\mathcal{G}$ to output a matrix, R. One can show that the spectrum of $R^\top R$ approximates the spectrum of $L_{\mathcal{G}}$ if the dimension of random projection is high enough. It is claimed in Blocki et al. [9] that their output is a sanitized graph. However, even though their output is a Laplacian, one major drawback of random projection is that it does not preserve the structure of the matrix, which in our case is a edge-incidence matrix of a graph (we refer the readers to [13, 55] for more discussion on benefits and pitfalls of random projections). Likewise, the output of Dwork et al. [19] is a full rank matrix (and

[2]Note that the number of non-zero singular values of n vertex graph can be at most $n-1$ [20], where n is the number of nodes in the graphs. That is, if we consider node-level privacy, then the Laplacian of a graph $\mathcal{G}$ has at most n singular values and that for neighboring graph $\mathcal{G}'$ is at most $n-1$ singular values. Thus outputting any spectral sparsification would not preserve privacy.

hence not a Laplacian) that is not positive semi-definite matrix. Consequently, their result cannot be used for spectral sparsification and other applications considered in this paper. On the other hand, Gupta et al. [23] only approximates cut functions and not the spectrum.

The existing techniques for graph sparsification are either deterministic [7][3] or use importance sampling that depends on the graph itself [50, 52]. A popular approach (and the one we use) involves sampling each edge with probability proportional to its *effective resistance*. We show that effective resistance based sampling can be done privately, albeit not trivially.

In order to comprehend the issues with private sampling based on the effective resistance, consider two neighboring graphs, $\mathcal{G}$ and $\mathcal{G}'$, such that $\mathcal{G}$ has two connected components and $\mathcal{G}'$ has an edge e between the two connected components of $\mathcal{G}$. No sparsifier for $\mathcal{G}$ will have the edge e; however, every sparsifier for $\mathcal{G}'$ has to contain the edge e. This allows one to easily differentiate the two cases. Furthermore, we show that the effective resistance is not Lipschitz smooth, so we cannot hope to privately compute effective resistance through output perturbation. One could argue that we can instead use (a variant of) *smooth sensitivity framework* [40], but it is not clear which function is a smooth bound on the sensitivity of effective resistance.

3.1 A High-level Overview of Our Algorithm

As we noted earlier, spectral sparsification of a graph can be posed as a linear algebra problem and computing effective resistance suffices for private sparsification of graphs. We propose an algorithm to sample using privately computed effective resistance of edges. Our algorithm is based on the following key ideas:

1. Private sketch of the left and right singular vectors of the edge-adjacency matrix is enough to compute all the effective resistances privately and a (possibly dense) private graph, $\mathcal{G}_{\mathsf{int}}$, that approximates the spectrum of the input graph.
2. We then sparsify $\mathcal{G}_{\mathsf{int}}$ to output a graph with $O(n/\varepsilon^2)$ edges with a small depreciation in the spectral approximation using any known non-private spectral sparsifier.

Computing effective resistance privately. We use the input perturbation technique (and its variants) first introduced by Blocki et al. [9] to compute effective resistances privately. The scalars $\{\tau_1, \ldots, \tau_m\}$ and $\mathcal{G}_{\mathsf{int}}$ are then computed using these effective resistances. Blocki et al. [9] first overlay a weighted complete graph K_n on the input graph $\mathcal{G}$ to get a graph $\widehat{\mathcal{G}}$ and then multiply its edge-adjacency matrix $E_{\widehat{\mathcal{G}}}$ with a Gaussian matrix, say N. We view this algorithm as a private sketching algorithm, where the sketch is $R := NE_{\widehat{\mathcal{G}}}$. We prove in the supplementary material that if N is of appropriate dimension as chosen in Step 2 of Algorithm 1, then $(1-\varepsilon)L_{\mathcal{G}} \preceq E_{\widehat{\mathcal{G}}}^{\top} N^{\top} N E_{\widehat{\mathcal{G}}} \preceq L_{\mathcal{G}}$. However, R does not contains enough information to estimate the effective resistances of $E_{\widehat{\mathcal{G}}}$. For this, we sketch the left singular space as $L := \left(E_{\widehat{\mathcal{G}}} \quad | \quad w\mathbb{I}\right) M$ for a Gaussian matrix M. We use different methods to sketch the right and left singular space so as to preserve differential privacy. Combined together L and R has enough statistics to privately estimate effective resistance, $\widetilde{\tau}_e$, of each edge e using a simple linear algebra computation (Step 2 of Algorithm 1).

Constructing $\mathcal{G}_{\mathsf{int}}$. We overlay a weighted K_n in order to privately sketch the left singular space leading to $\binom{n}{2}$ effective resistances. We define a set of scalars $\widetilde{\tau}_1, \ldots, \widetilde{\tau}_{\binom{n}{2}}$ using the computed effective resistance as in Step 2 of Algorithm 1. Let $\widetilde{\mathcal{G}}'$ be the graph formed by overlaying a weighted complete graph with edge weights sampled i.i.d. from a Gaussian distribution on $\widehat{\mathcal{G}}$. We cannot use existing non-private algorithms for spectral sparsification on graph $\widetilde{\mathcal{G}}'$ as it may have negative weight edges. To get a sanitized graph on which we can perform sparsification, we solve the following:

$$\text{SDP-1} : \min \left\{ \lambda : L_{\bar{\mathcal{G}}} - L_{\widetilde{\mathcal{G}}'} \preceq \lambda L_n,\ L_{\widetilde{\mathcal{G}}'} - L_{\bar{\mathcal{G}}} \preceq \lambda L_n,\ L_{\bar{\mathcal{G}}} \in \mathcal{C} \right\}.$$

Here $\mathcal{C}$ is the convex cone of the Laplacian of graphs with positive weights. Note that, SDP-1 has a solution with $\lambda = O(\frac{w}{n})$. This is because the original graph $\mathcal{G}$ achieves this value. Since, the semidefinite program requires to output a graph $\bar{\mathcal{G}}$ with Laplacian $L_{\bar{\mathcal{G}}}$, we are guaranteeed that its

[3] Deterministic algorithms cannot be differentially private. On the other hand, if not done carefully, the sampling probability that depends on the graph can itself leak privacy.

Algorithm 1 PRIVATE-SPARSIFY$(\mathcal{G}, \varepsilon, (\alpha, \beta))$

Input: An n vertex graph $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$, privacy parameters: (α, β), sparsification parameter: ε.
Output: A graph $\widetilde{\mathcal{G}}$.

1: **Initialization.** Set $m = \binom{n}{2}, p = m + n$. Sample random Gaussian matrices $M \in \mathbb{R}^{p \times n}, N \in \mathbb{R}^{n/\varepsilon^2 \times m}$, $Q \in \mathbb{R}^{p \times p/\varepsilon^2}$, such that $M_{ij} \sim \mathcal{N}(0, \frac{1}{n}), N_{ij} \sim \mathcal{N}(0, \frac{\varepsilon^2}{n})$, and $Q_{ij} \sim \mathcal{N}(0, \frac{\varepsilon^2}{p})$.
2: **Set** $w = O(\frac{1}{\alpha\varepsilon}\sqrt{n \log n \log(1/\beta)} \log(1/\beta))$.
3: **Compute** $L := \left(E_{\widehat{\mathcal{G}}} \quad | \quad w\mathbb{I}\right) M, R := NE_{\widehat{\mathcal{G}}}$, where $E_{\widehat{\mathcal{G}}} = \sqrt{\left(1 - \frac{w}{n}\right)} E_{\mathcal{G}} + \sqrt{\frac{w}{n}} E_{K_n}$.
4: **Define** $\widetilde{\tau}_i = L_i \left(R^\top R\right)^\dagger L_i^\top$ for every $i \in [m]$ and $p_i = \min\left\{c\widetilde{\tau}_i \varepsilon^{-2} \log(n/\delta), 1\right\}$.
5: **Construct** a diagonal matrix $D \in \mathbb{R}^{\binom{n}{2} \times \binom{n}{2}}$ whose non-zero diagonal entries are $D_{ii} := p_i^{-1}$ with probability p_i.
6: **Construct** a complete graph $\mathcal{H}$ with edge weights i.i.d. sampled from $\mathcal{N}(0, \frac{12 \log(1/\beta)}{\alpha^2})$.
7: **Construct** $\widetilde{\mathcal{G}}'$ such that $L_{\widetilde{\mathcal{G}}'} = L_{\widehat{\mathcal{G}}} + L_{\mathcal{H}}$. Solve the SDP-1 to get a graph $\bar{\mathcal{G}}$.
8: **Output** $\widetilde{\mathcal{G}}$ formed by running the algorithm of Lee and Sun [35] on $E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}}$.

output, $\bar{\mathcal{G}}$, will have $\lambda = O(\frac{w}{n})$. The Laplacian of the graph $\mathcal{G}_{\mathsf{int}}$ is then constructed by computing $\sum_e \widetilde{\tau}_e u_e^\top u_e$, where u_e are the edges of $\bar{\mathcal{G}}$. The graph $\mathcal{G}_{\mathsf{int}}$ can be a dense graph, but we show that it approximates the spectrum of the input graph up to $\widetilde{O}(\frac{w}{n})$ additive error. To reduce the number of edges, we then run any existing non-private spectral sparsification algorithm to get the final output.

3.2 Main Result

We now state our result that achieves both the guarantees of Definition 2.

Theorem 3 (Private spectral sparsification)**.** *Given privacy parameter (α, β), accuracy parameter ε, an n vertex graph $\mathcal{G}$, let L_n denote the Laplacian of an unweighted complete graph, K_n, and $w = O\left(\frac{\log(1/\beta)}{\alpha\varepsilon}\sqrt{n \log n \log(1/\beta)}\right)$. Then we have the following:*

1. PRIVATE-SPARSIFY *is a polynomial time (α, β)-differentially private algorithm.*
2. $\widetilde{\mathcal{G}} \leftarrow$ PRIVATE-SPARSIFY$(\mathcal{G}, \varepsilon, (\alpha, \beta))$ *has $O(n/\varepsilon^2)$ edges, such that with probability at least $9/10$, we have*

$$(1 - \varepsilon)\left(\left(1 - \frac{w}{n}\right) L_{\mathcal{G}} + \frac{w}{n} L_n\right) \preceq L_{\widetilde{\mathcal{G}}} \preceq (1 + \varepsilon)\left(\left(1 - \frac{w}{n}\right) L_{\mathcal{G}} + \frac{w}{n} L_n\right).$$

Proof Sketch of Theorem 3. Our algorithm requires performing matrix computations and solving a semi-definite program. All matrix computation requires at most $O(n^3)$ time. Solving a semi-definite program takes $\mathrm{poly}(n)$ time, where the exact polynomial depends on whether we use interior point, ellipsoid method, or primal-dual approach. To prove privacy, we need to argue that computing L and R is $(\alpha/3, \beta/3)$-differentially private. Blocki et al. [9] showed that computing R is $(\alpha/3, \beta/3)$-differentially private while computing $\mathcal{G}'$ is private due to Gaussian mechanism. The privacy guarantee on L follows using arguments similar to [9].

For the accuracy proof, recall that we approximate the left singular space by L and the right singular space by R by using random Gaussian matrix, and then use $(L(R^\top R)^\dagger L^\top)_{ii}$ to approximate effective resistance $\widetilde{\tau}_i$ (see Step 2) – this follows from the concentration property of random Gaussian matrices. It is straightforward, then, to argue that sampling using the effective resistance thus computed would result in a graph. Proving that the estimated effective resistances provide the spectral approximation is a bit more involved and relies on matrix Bernstein inequality [53].

Our proof requires the spectral relation between $\bar{\mathcal{G}}$ and $\widehat{\mathcal{G}}$ since X_i is defined with respect to the edges of $\bar{\mathcal{G}}$ and $\widetilde{\tau}_i$ is defined with respect to $\widehat{\mathcal{G}}$. Since the solution of SDP-1 is $\lambda = c\sqrt{\frac{\log n \log(1/\delta)}{n\alpha^2}}$, we have

$$L_{\widetilde{\mathcal{G}}'} - c\sqrt{\frac{\log n \log(1/\delta)}{n\alpha^2}} L_n \preceq L_{\bar{\mathcal{G}}} \preceq L_{\widetilde{\mathcal{G}}'} + c\sqrt{\frac{\log n \log(1/\delta)}{n\alpha^2}} L_n.$$

Another application of matrix Bernstein inequality gives us for some small constant $\rho > 0$,

$$\mathsf{Pr}\left[L_{\widehat{\mathcal{G}}} - \rho\sqrt{\frac{\log n \log(1/\delta)}{n\alpha^2}}L_n \preceq L_{\widetilde{\mathcal{G}}'} \preceq L_{\widehat{\mathcal{G}}} + \rho\sqrt{\frac{\log n \log(1/\delta)}{n\alpha^2}}L_n\right] \geq 1 - \delta.$$

Let e_i be the edges in $\bar{\mathcal{G}}$ defined in Algorithm 1. Define random variables X_i as follow:

$$X_i := \begin{cases} \frac{e_i e_i^\top}{p_i} & \text{with probability } p_i \\ 0 & \text{with probability } 1 - p_i \end{cases}.$$

Let $Y = \sum_i X_i$. Then we show the following:

$$\mathsf{E}[Y] = \sum_i \mathsf{E}[X_i] = L_{\bar{\mathcal{G}}} \quad \text{and} \quad X_i \preceq O\left(\frac{\varepsilon^2}{c\log(n/\delta)}\right) L_{\bar{\mathcal{G}}}.$$

Applying matrix Bernstein inequality [53] gives

$$\mathsf{Pr}\left[(1-\varepsilon)L_{\bar{\mathcal{G}}} \preceq Y \preceq (1+\varepsilon)L_{\bar{\mathcal{G}}}\right] \geq 1 - ne^{-c\varepsilon^{-2}\log(n/\delta)/3}$$

for large enough c. Now $E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}} = Y$ implies $(1-\varepsilon)L_{\bar{\mathcal{G}}} \preceq E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}} \preceq (1+\varepsilon)L_{\bar{\mathcal{G}}}$ with probability at least $99/100$. Now $E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}}$ can be the Laplacian of a dense graph. Using the result of Lee and Sun [35], we have $(1-\varepsilon)L_{\widetilde{\mathcal{G}}} \preceq L_{\bar{\mathcal{G}}} \preceq (1+\varepsilon)L_{\widetilde{\mathcal{G}}}$ and that $L_{\widetilde{\mathcal{G}}}$ has $O(n/\varepsilon^2)$ edges. Combining all these partial orderings gives us the accuracy bound.

Extension to Weighted Graphs We can use a standard technique to extend the result in Theorem 3 to weighted graphs. We assume that the weights on the graph are integers in the range $[1, \operatorname{poly} n]$. We consider different levels $(1+\varepsilon)^i$ for $i \in [c\log n]$ for some constant c. Then, we consider input graphs of the following form, $L_{\mathcal{G}} = \sum_{i=1}^{c\log n} L_{\mathcal{G},i}$, where $L_{\mathcal{G},i}$ has edges with weights $\left\{0, (1+\varepsilon)^i\right\}$. In other words, we use the $(1+\varepsilon)$-ary representation of weights on the edges and partition the graph accordingly and run Algorithm 1 on each $L_{\mathcal{G},i}$. Again, since there are at most $\operatorname{poly}\log n$ levels, the number of edges in $\widetilde{\mathcal{G}}$ is $\widetilde{O}(\frac{n}{\varepsilon^2})$. Since $\widetilde{\mathcal{G}}$ is $(\alpha \operatorname{poly}\log n, \beta \operatorname{poly}\log n)$-differentially private, we can run another instance of [35] to get a graph $\widehat{\mathcal{G}}$ with $O(\frac{n}{\varepsilon^2})$ edges.

Private Estimation of Effective Resistances. Drineas and Mahoney showed a close relation between effective resistance and statistical leverage scores [16]. Their result imply that effective resistance are important statistical property of a graph. As a by-product of Algorithm 1, we can privately estimate the effective resitances of the edges. As mentioned earlier, this is not possible through previous approaches. For example, both Blocki et al. [9] and Dwork et al. [19] only approximates the right singular space, which is not sufficient to capture enough statistics to estimate the effective resistances. While effective resistance have been less explored, statistical leverage scores have been widely used by statisticians. Consequently over the period of time, researchers have explored some other statistical applications includes random feature learning [47] and quadrature [6] of effective resistances.

4 Applications of Theorem 3

Since differential privacy is preserved under any post-processing, the output of Algorithm 1 can be used in many graph problems. It is well known that the spectrum of a graph defines many structural and combinatorial properties of the graph [20]. Since Theorem 3 guarantees a sparse graph, it allows us to significantly improve the run-time for many graph algorithms, for example, min-cuts and separators [49], heat kernel computations [41], approximate Lipschitz learning on graphs [32], spectral clustering, linear programming [34], solving Laplacian systems [33], approximation algorithms [29, 42], and matrix polynomials in the graph Laplacian [12].

For example, Theorem 3 with Goemans and Williamson [21] (and Arora et al. [5]) allows us to output a partition of nodes that approximates MAX-CUT (SPARSEST-CUT, respectively). We can also answer all possible cut queries with the same accuracy as previous best in $O(|S|)$ time instead of $O(n|S|)$ time. This is a significant improvement for practical graphs. Lastly, we exhibit the versatility of our result by showing its application in extracting low-dimensional representations when data arise from sampling a probability distribution on a manifold, a typical task in representation learning.

Problem	Additive Error	Previous Best Error	Theorem
$(S, V\backslash S)$ queries	$\widetilde{O}\left(\frac{\lvert S\rvert\sqrt{n}}{\alpha}\right)$	$\widetilde{O}\left(\frac{\lvert S\rvert\sqrt{n}}{\alpha}\right)$ [9, 19]	Theorem 4
(S, T) queries	$\widetilde{O}\left(\frac{\lvert S\rvert\lvert T\rvert}{\alpha\sqrt{n}}\right)$	$\widetilde{O}\left(\frac{\sqrt{n\lvert S\rvert\lvert T\rvert}}{\alpha}\right)$ [23]	Theorem 4
MAX-CUT	$\widetilde{O}\left(\frac{\lvert S_m\rvert\sqrt{n}}{\alpha}\right)$	$\widetilde{O}\left(\frac{n^{3/2}}{\alpha}\right)$ [23]	Theorem 5
SPARSEST-CUT	$\widetilde{O}(\frac{1}{\sqrt{n\alpha^2}})$	$\widetilde{O}(\frac{\sqrt{n}}{\lvert S_s\rvert\alpha})$ [23]	Theorem 5
Eigenmap	$\widetilde{O}\left(\frac{\sqrt{n}}{\alpha}\right)$	$\widetilde{O}\left(\frac{\sqrt{n}}{\alpha}\right)$ [19]	Theorem 6

Table 1: Applications of Our Results for $\beta = \Theta(n^{-\log n})$ ($\alpha > 0$ is an arbitrary constant, S_m is vertex set inducing MAX-CUT, S_c is vertex set inducing SPARSEST-CUT, Q is vertex set for cut query).

In the rest of this section, we discuss these applications in more detail (see Table 1 for comparison). For this section, let $w = O(\frac{\log(1/\beta)}{\alpha\varepsilon}\sqrt{n\log n\log(1/\beta)})$.

Answering (S, T) cut queries efficiently. Cut queries is one of the most widely studied problem in private graph analysis [9, 23, 54] and has wide applications [43, 46]. Given a graph $\mathcal{G}$, an (S, T)-cut query requires us to estimate the sum of weights of edges between the vertex sets in S and T. Since our output is a graph, we can use it to answer (S, T) cut queries privately and more efficiently.

Theorem 4 (Cut queries)**.** *Given a graph with n vertices, there is an efficient (α, β)-differentially private algorithm that outputs a graph $\widetilde{\mathcal{G}}$, which can be used to answer all possible (S, T)-cut queries with additive error $O(\frac{\lvert S\rvert\lvert T\rvert}{\alpha\varepsilon}\sqrt{\frac{\log n\log^3(1/\beta)}{n}})$. In particular, when $T = V\backslash S$, then our algorihtm incur an additive error $O(\frac{\min\{\lvert S\rvert,\lvert V\backslash S\rvert\}}{\alpha\varepsilon}\log^3(1/\beta)\sqrt{n\log n\log(1/\beta)})$.*

This improves the result of Gupta et al. [23] who incur an $O(\frac{1}{\alpha}\sqrt{n\lvert S\rvert\lvert T\rvert})$ additive error. Note that Blocki et al. [9] and Dwork et al. [19] can be used to only answer cut queries when $T = V\backslash S$. For $(S, V\backslash S)$-cut queries, recall that Dwork et al. output $C = E_{\mathcal{G}}^{\top}E_{\mathcal{G}} + N$, where N is a Gaussian matrix with appropriate noise to preserve differential privacy. For a set of q cut queries of form $(S, V\backslash S)$ with $\lvert S\rvert \leq \lvert V\backslash S\rvert$, standard concentration result of Gaussian distribution implies that Dwork et al. [19] incur $\lvert\mathbf{1}_S^{\top}N\mathbf{1}_S\rvert = O(\lvert S\rvert\sqrt{\log(1/\delta)\log(q)}/\alpha)$ additive error, where $\mathbf{1}_S \in \{0, 1\}^n$ has 1 only in coordinate corresponding to $S \subseteq [n]$. In particular, if we wish to answer all possible cut queries, it leads to an additive error $O(\lvert S\rvert\sqrt{n\log(1/\beta)}/\alpha)$. We thus match these bounds [9, 19, 54] while answering an $(S, V\backslash S)$ query in $O(\lvert S\rvert/\varepsilon^2)$ amortized time instead of $O(\lvert S\rvert^2)$ amortized time.

Optimization problems. Given a graph $\mathcal{G} = (V, E)$ on a vertex set V, the goal of MAX-CUT to output a set of vertices $S \subseteq V$ that maximizes the value of $\Phi_S(\mathcal{G})$. It is well known that solving MAX-CUT exactly and even with $(0.87856 + \rho)$-approximation for $\rho > 0$ is NP-hard [31]. However, Goemans and Williamson [21] gave an elegant polynomial time algorithm for computing $0.87856 - \eta$ approximation to MAX-CUT for some $\eta > 0$, thereby giving an approximation algorithm that is optimal with respect to the multiplicative approximation. Another problem that is considered in graph theory is the problem of finding SPARSEST-CUT. Here, given a graph $\mathcal{G} = (V, E)$ on a vertex set V, the goal is to output a set of vertices $S \subseteq V$ that minimizes the value $\frac{\Phi_S(\mathcal{G})}{\lvert S\rvert(n-\lvert S\rvert)}$. The proposed algorithms for these problems first computes a private sparse graph as in Algorithm 1 followed by a run of the non-private algorithm ([21] in the case of MAX-CUT and [5] in the case of SPARSEST-CUT) to obtain a set of vertices S. We show the following guarantee.

Theorem 5 (Optimization problems)**.** *There is a polynomial-time algorithm that, for an n-vertex graph $\mathcal{G} := (V, E)$, is (α, β)-differentially private with respect to the edge level privacy and produces a partition of nodes $(S, V\backslash S)$ satisfying*

$$\text{MAX-CUT:}\quad \Phi_S(\mathcal{G}) \geq (0.87856 - \eta)\left(\frac{1-\varepsilon}{1+\varepsilon}\right)\max_{S\subseteq V}\Phi_S(\mathcal{G}) - O\left(\frac{w\lvert S\rvert}{\alpha}\right).$$

There is a polynomial-time algorithm that, for an n-vertex graph $\mathcal{G} := (V, E)$, is (α, β)-differentially private with respect to the edge level privacy and produces a partition of nodes $(S, V\backslash S)$ satisfying

$$\text{SPARSEST-CUT:}\quad \frac{\Phi_S(\mathcal{G})}{\lvert S\rvert(n-\lvert S\rvert)} \leq O(\sqrt{\log n})\left(\frac{1+\varepsilon}{1-\varepsilon}\right)\min_{S\subseteq V}\left(\frac{\Phi_S(\mathcal{G})}{\lvert S\rvert(n-\lvert S\rvert)}\right) + O\left(\frac{w\log^2 n}{n}\right).$$

The above theorem states that we can approximately and privately compute MAX-CUT and SPARSEST-CUT of an arbitrary graph in polynomial time. Further, the price of privacy in the form of the additive error scales sublinearly with n. On the other hand, if we use the privatized graph of [23], it would incur an error of $O(\frac{n^{3/2}}{\alpha})$ for MAX-CUT and $O(\frac{\sqrt{n}}{\varepsilon|S|})$ for SPARSEST-CUT.

One may argue that we can use the output of Dwork et al. [19] or Blocki et al. [9] to solve theses optimization problem. Unfortunately, it is not the case. To see this, let us recall their output. For a given graph $\mathcal{G}$, Blocki et al. [9] output $R^\top R$, where R is as computed in Step 4 of Algorithm 1. On the other hand, Dwork et al. [19] computes $L_\mathcal{G} + N$, where N is a symmetric Gaussian matrix with entries sampled i.i.d. with variance required to preserve privacy.

Both these approach output a symmetric matrix; however, the output of Dwork et al. [19] is neither a Laplacian nor a positive semi-definite matrix, a requirement in all the existing algorithms. On the other hand, even though the output of Blocki et al. [9] is a positive semi-definite matrix, it can have positive off-diagonal entries. As such, we cannot use them in the existing algorithms for MAX-CUT or SPARSEST-CUT since (analysis of) existing techniques for these optimization problems requires graph to be positively weighted (see the note by Har-Peled [24])[4]. Even if we can use their output, our algorithm allows a faster computation since we significantly reduce the number of constraints in the corresponding semi-definite programs for these optimization problems.

Learning laplacian eigenmap. A basic challenge in machine learning is to learn a low-dimensional representation of data drawn from a probability distribution on a manifold. This is also referred to as the problem of *manifold learning*. In recent years, many approaches have been proposed for manifold learning, including that of ISOMAP, local linear embedding, and Laplacian eigenmap.

In particular, the state-of-the-art Laplacian eigenmaps are relatively insensitive to outliers and noise due to locality-preserving character. In this approach, given n data samples $\{x_1, \ldots, x_n\} \in \mathbb{R}^d$, we construct a weighted graph $\mathcal{G}$ with n nodes and a set of edges connecting neighboring points. The embedding map is now provided by computing the top k eigenvectors of the graph Laplacian. There are multiple ways in which we assign an edge $e = (u, v)$ and edge weight between nodes u and v. We consider the following neighborhood graph: we have an edge $e = (u, v)$ if $\|x_u - x_v\|_2 \leq \rho$ for some parameter ρ. If there is an edge, then that edge is given a weight as per the *heat kernel*, $e^{-\|x_u - x_v\|_2^2/t}$ for some parameter $t \in \mathbb{R}$. The goal here is to find the embedding map, i.e., an orthonormal projection matrix, $UU^\top$, close to the optimal projection matrix, $U_k U_k^\top$, where the columns of U_k are the top-k eigenvectors of $L_\mathcal{G}$. Using our framework, we can guarantee the following for privately learning the Laplacian eigenmap matching the bound achieved by previous results [19].

Theorem 6 (Laplacian eigenmap)**.** *Let $\mathcal{G}$ be the neighborhood graph formed by the data samples $\{x_1, \cdots, x_n\} \in \mathbb{R}^d$ as described above. Let $L_{\mathcal{G},k} = U_k U_k^\top L_\mathcal{G}$, where the columns of U_k are the top-k eigenvectors of $L_\mathcal{G}$. Then, there is an efficient (α, β)-differentially private learning algorithm that outputs a rank-k orthonormal matrix $U \in \mathbb{R}^{n \times k}$ such that*

$$\left\|(\mathbb{I} - UU^\top)L_\mathcal{G}\right\|_2 \leq (1+\varepsilon)\left\|L_\mathcal{G} - L_{\mathcal{G},k}\right\|_2 + O\left(w\right).$$

5 Differentially Private Cut Sparsification

Benzur and Karger [8] introduced the notion of cut sparsification. In this section, we present an algorithm that outputs a cut sparsifier while preserving $(\alpha, 0)$-differential privacy. We use this algorithm to answer cut queries, approximately computing MAX-CUT, SPARSEST-CUT, and EDGE-EXPANSION, with $(\alpha, 0)$-differential privacy. We show the following:

Theorem 7. *Let $\mathcal{G} = (V, E)$ be an unweighted graph. Given an approximation parameter $0 < \varepsilon < 1$ and privacy parameter α,* PRIVATE-CUT-SPARSIFY*, described in Algorithm 2, is $(\alpha, 0)$-differentially private. Further,* PRIVATE-CUT-SPARSIFY *outputs an n vertices $\widetilde{O}(n/\varepsilon^2)$ edges graph, $\widetilde{\mathcal{G}}$, such that, with probability at least $99/100$, we have $\forall S \subseteq V$,*

$$(1-\varepsilon)\mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}}\mathbf{1}_S - O\left(\sqrt{n\mathbf{1}_S^\top L_n \mathbf{1}_S}\right) \leq \alpha \cdot \mathbf{1}_S^\top L_\mathcal{G}\mathbf{1}_S \leq (1+\varepsilon)\mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}}\mathbf{1}_S + O\left(\sqrt{n\mathbf{1}_S^\top L_n \mathbf{1}_S}\right).$$

[4] Alon et al. [3] gave an algorithm for solving MAX-CUT for real-weight graphs using quadratic program (instead of semi-definite program based approach [21]), but that leads to an $O(\log n)$ multiplicative approximation.

Algorithm 2 PRIVATE-CUT-SPARSIFY $(\mathcal{G}, \varepsilon, \alpha)$

Input: An n vertex graph $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$, privacy parameters (α, β), approximation parameter ε.
Output: A Laplacian $L_{\widetilde{\mathcal{G}}}$.

1: **Privatize.** Construct a complete graph $\widehat{G}$ with weights w_e defined as follows:

$$w_e := \begin{cases} +1 & \text{with probability } (1+\alpha)/2 \quad \text{if } e \in E_{\mathcal{G}} \\ -1 & \text{with probability } (1-\alpha)/2 \quad \text{if } e \in E_{\mathcal{G}} \\ +1 & \text{with probability } 1/2 \quad \text{if } e \notin E_{\mathcal{G}} \\ -1 & \text{with probability } 1/2 \quad \text{if } e \notin E_{\mathcal{G}} \end{cases}.$$

2: **Compute** $\bar{\mathcal{G}}$ using the linear program of Gupta et al. [23] on $\widehat{\mathcal{G}}$.
3: **Output** $\widetilde{\mathcal{G}}$ by running the algorithm of Benzur and Karger [8] on $\bar{\mathcal{G}}$.

The above theorem says that any cut can be approximated by our cut sparsifier up to $\widetilde{O}(\sqrt{n(n-s)s})$ error, while preserving $(\alpha, 0)$-differential privacy. This is an instance based bound and matches the accuracy achieved by our graph sparsification result (and the best possible [9, 54]) when $s = O(n)$.

6 Discussion

In this paper, we introduced private spectral sparsification of graphs. We gave efficient algorithms to compute one such sparsification. Our algorithm outputs a graph with $O(n/\varepsilon^2)$ edges, while preserving differential privacy. Our algorithmic framework allows us to compute an approximation to MAX-CUT and SPARSEST-CUT with better accuracy than previously known, and a first algorithm for learning differentially private Laplacian eigenmaps.

Our algorithm uses both importance sampling and random sketching. At a high level, sketching allows us to ensure privacy and importance sampling allows us to produce spectral sparsification. To the best of our knowledge, this is the first instance of using importance sampling in the context of privacy. Since important sampling is an important tool in non-private low-space algorithms [55], this we believe can lead to development of other private algorithms.

Our work differs from previous works that use random projections or graph sparsification [9, 54]. The only work that we are aware of which uses spectral sparsification in the context of differential privacy is that of Upadhyay [54] with an aim to improve the run-time efficiency of Blocki et al. [9]; however, their algorithm does not output a graph let alone a sparse graph. Hence, their approach cannot be used to approximate MAX-CUT or SPARSE-CUT – the only method to solve these problems would be to run all possible cut queries leading to an error of $O(n^{3/2}/\alpha)$. This follows because the error per cut query would be $O(\frac{\sqrt{n}|S|}{\varepsilon})$, and since there are 2^n possible cuts, an application of Chernoff bound results in worst case error $O(\frac{n^{3/2}}{\alpha})$, matching the guarantee of Gupta et al. [23].

On the other hand, Gupta et al. [23] give an algorithm to output a graph that preserves cut functions on a graph. However, their output does not preserve the spectral properties of the graph and so cannot be used in spectral applications of graphs, such as Laplacian eigenmap or learning Lipschitz functions on graphs. Moreover, their algorithm incurs large additive error. We also give a tighter analysis for a variant of their algorithm to achieve an instance based additive error.

Acknowledgements. This research was supported, in part, by NSF BIGDATA grants IIS-1546482 and IIS-1838139, and DARPA award W911NF1820267. This work was done when Jalaj Upadhyay was working as a postdoctoral researcher with Raman Arora at the Johns Hopkins University. Authors would like to thank Adam Smith, Lorenzo Orecchia, Cynthia Steinhardt, and Sarvagya Upadhyay for insightful discussions during the early stages of the project.

References

[1] Dimitris Achlioptas and Frank McSherry. Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM)*, 54(2):9, 2007.

[2] Zeyuan Allen-Zhu, Zhenyu Liao, and Lorenzo Orecchia. Spectral sparsification and regret minimization beyond matrix multiplicative updates. In *STOC*, pages 237–245. ACM, 2015.

[3] Noga Alon, Konstantin Makarychev, Yury Makarychev, and Assaf Naor. Quadratic forms on graphs,. *Inventiones Mathematicae*, 163(3):499–522, 2006.

[4] Raman Arora, Vladimir Braverman, and Jalaj Upadhyay. Differentially private robust low-rank approximation. In *Advances in Neural Information Processing Systems*, pages 4137–4145, 2018.

[5] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):5, 2009.

[6] Francis Bach. On the equivalence between kernel quadrature rules and random feature expansions. *The Journal of Machine Learning Research*, 18(1):714–751, 2017.

[7] Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.

[8] András A Benczúr and David R Karger. Approximating st minimum cuts in õ (n 2) time. In *STOC*, pages 47–55. ACM, 1996.

[9] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The Johnson-Lindenstrauss Transform Itself Preserves Differential Privacy. In *FOCS*, pages 410–419, 2012.

[10] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *ITCS*, pages 87–96, 2013.

[11] Shixi Chen and Shuigeng Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 653–664. ACM, 2013.

[12] Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, and Shang-Hua Teng. Efficient sampling for gaussian graphical models via spectral sparsification. In *COLT*, pages 364–390, 2015.

[13] Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *ITCS*, pages 181–190. ACM, 2015.

[14] Michael B Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1758–1777. SIAM, 2017.

[15] James C Costello, Mehmet M Dalkilic, Scott M Beason, Jeff R Gehlhausen, Rupali Patwardhan, Sumit Middha, Brian D Eads, and Justen R Andrews. Gene networks in drosophila melanogaster: integrating experimental data to predict gene function. *Genome biology*, 10(9):R97, 2009.

[16] Petros Drineas and Michael W Mahoney. Effective resistances, statistical leverage, and applications to linear equation solving. *arXiv preprint arXiv:1005.3097*, 2010.

[17] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[18] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and Differential Privacy. In *FOCS*, pages 51–60, 2010.

[19] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze Gauss: Optimal Bounds for Privacy-Preserving Principal Component Analysis. In *STOC*, pages 11–20, 2014.

[20] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[21] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[22] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *SODA*, pages 1106–1125. SIAM, 2010.

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

[24] Sariel Har-Peled. Approximate max-cut. 2005.

[25] Moritz Hardt and Aaron Roth. Beating randomized response on incoherent matrices. In Howard J. Karloff and Toniann Pitassi, editors, *STOC*, pages 1255–1268. ACM, 2012.

[26] Minoru Kanehisa and Susumu Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, 2000.

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

[28] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography*, pages 457–476. Springer, 2013.

[29] Jonathan A Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *SODA*, pages 217–226. SIAM, 2014.

[30] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775. ACM, 2002.

[31] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007.

[32] Rasmus Kyng, Anup Rao, Sushant Sachdeva, and Daniel A Spielman. Algorithms for lipschitz learning on graphs. In *Conference on Learning Theory*, pages 1190–1223, 2015.

[33] Yin Tat Lee, Richard Peng, and Daniel A Spielman. Sparsified cholesky solvers for sdd linear systems. *arXiv preprint arXiv:1506.08204*, 2015.

[34] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in o (vrank) iterations and faster algorithms for maximum flow. In *FOCS*, pages 424–433. IEEE, 2014.

[35] Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. In *FOCS*, pages 250–269. IEEE, 2015.

[36] Yin Tat Lee and He Sun. An sdp-based algorithm for linear-sized spectral sparsification. In *STOC*, pages 678–687. ACM, 2017.

[37] Adam Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families ii: Mixed characteristic polynomials and the kadison-singer problem. *arXiv preprint arXiv:1306.3969*, 2013.

[38] Carl D Meyer, Jr. Generalized inversion of modified matrices. *SIAM Journal on Applied Mathematics*, 24(3):315–323, 1973.

[39] Darakhshan Mir and Rebecca N Wright. A differentially private estimator for the stochastic kronecker graph model. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 167–176. ACM, 2012.

[40] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, 2007.

[41] Lorenzo Orecchia, Sushant Sachdeva, and Nisheeth K Vishnoi. Approximating the exponential, the lanczos method and an o (m)-time spectral algorithm for balanced separator. In *STOC*, pages 1141–1160. ACM, 2012.

[42] Richard Peng. Approximate undirected maximum flows in o(m polylog (n)) time. In *SODA*, pages 1862–1867. Society for Industrial and Applied Mathematics, 2016.

[43] J Scott Provan and Michael O Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.

[44] Sofya Raskhodnikova and Adam D. Smith. Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism. In *FOCS*, pages 495–504, 2016.

[45] Vibhor Rastogi, Michael Hay, Gerome Miklau, and Dan Suciu. Relationship privacy: output perturbation for queries with joins. In *Symp. Principles of Database Systems (PODS)*, pages 107–116, 2009.

[46] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[47] Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, pages 3215–3225, 2017.

[48] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pages 143–152. IEEE, 2006.

[49] Jonah Sherman. Nearly maximum flows in nearly linear time. In *FOCS*, pages 263–269. IEEE, 2013.

[50] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[51] Daniel A Spielman and Nikhil Srivastava. An elementary proof of the restricted invertibility theorem. *Israel Journal of Mathematics*, 190(1):83–91, 2012.

[52] Daniel A Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

[53] Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.

[54] Jalaj Upadhyay. Random Projections, Graph Sparsification, and Differential Privacy. In *ASIACRYPT (1)*, pages 276–295, 2013.

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

Supplementary Material to "On Differentially Private Graph Sparsification and Applications"

A Notation and Preliminaries

We let $\mathbb{N}$ to denote the set of natural numbers. We use capital letters to denote matrices and small letters to denote vectors. We denote by $\mathbf{0}^{m\times n}$ the all-zero $m \times n$ matrix and by $\mathbb{I}$ be the identity matrix. For a matrix A, we let $\mathsf{rk}(A)$ denote the rank, $\Delta(A)$ to denote its *determinant*, and $\mathsf{Tr}(A)$ denote the trace norm of the matrix A. The *singular-value decomposition* (SVD) of an $m \times n$ rank-r matrix A is a decomposition of A as a product of three matrices, $A = U\Sigma V^\top$ such that $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ have orthonormal columns and $\Sigma \in \mathbb{R}^{r\times r}$ is a diagonal matrix with singular values of A on its diagonal. We let $\mathbf{1}_S \in \{0,1\}^n$ denote the characteristic vector for a subset $S \subseteq [n]$. For a symmetric matrix, we write $A \succeq 0$ to denote a matrix whose singular values are greater than 0 and $A \succeq B$ if $A - B \succeq 0$. The *Moore-Penrose pseudo-inverse* of a matrix $A = U\Sigma V^\top$ is denoted by $A^\dagger$ and has a SVD $A^\dagger = V\Sigma^\dagger U^\top$, where $\Sigma^\dagger$ consists of inverses of only non-zero singular values of A. We will use the following fact in our proof.

Fact 8. *For any two symmetric matrices A and B, if $A \succeq B$, then $A^\dagger \preceq B^\dagger$.*

A matrix A has a *left-inverse* (*right-inverse*, respectively) if and only if it has full column rank (full row rank, respectively). If A has a left-inverse, then $A^\dagger = (A^\top A)^{-1}A^\top$ and if A has right-inverse, then $A^\dagger = A^\top(AA^\top)^{-1}$.

We need the following results for the privacy proof.

Theorem 9 (Lidskii Theorem)**.** *Let A, B be $n \times n$ Hermittian matrices. Then for any choice of indices $1 \le i_1 \le \cdots \le i_k \le n$,*

$$\sum_{j=1}^{k} \lambda_{i_j}(A+B) \le \sum_{j=1}^{k} \lambda_{i_j}(A) + \sum_{j=1}^{k} \lambda_{i_j}(B),$$

where $\{\lambda_i(A)\}_{i=1}^n$ are the eigen-values of A in decreasing order.

A.1 Gaussian Distribution

Given a random variable x, we denote by $\mathcal{N}(\mu, \rho^2)$ the fact that x has a normal Gaussian distribution with mean μ and variance ρ^2. The Gaussian distribution is invariant under affine transformation, i.e., if $x \sim \mathcal{N}(\mu_1, \sigma_1)$ and $y \sim \mathcal{N}(\mu_2, \sigma_2)$, then $z = ax+by$ has the distribution $z \sim \mathcal{N}(a\mu_1+b\mu_2, a\sigma_1^2+b\sigma_2^2)$. By simple computation, one can verify that the tail of a standard Gaussian variable decays exponentially. More specifically, for a random variable $x \sim \mathcal{N}(0,1)$, we have $\mathsf{Pr}\left[|x| > t\right] \le 2e^{-t^2/2}$.

The multivariate Gaussian distribution is a generalization of univariate Gaussian distribution. Given an m-dimensional multivariate random variable, $X \sim \mathcal{N}(\mu, \Sigma)$ with mean $\mu \in \mathbb{R}^m$ and covariance matrix $\Sigma = \mathsf{E}[(X-\mu)(X-\mu)^T]$, the probability density function of a multivariate Gaussian is given by $\frac{1}{\sqrt{2\pi\tilde{\Delta}(\Sigma)}} \exp\left(-\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x}\right)$. It is easy to see from the description of the PDF that, in order to define the PDF corresponding to a multivariate Gaussian distribution, Σ has to have full rank. If Σ has a non-trivial kernel space, then the PDF is undefined. However, in this paper, we only need to compare the probability distribution of two random variables which are defined over the same subspace. Therefore, in those scenarios, we would restrict our attention to the subspace orthogonal to the kernel space of Σ.

Multivariate Gaussian distribution maintains many key properties of univariate Gaussian distribution. For example, any (non-empty) subset of multivariate normals is multivariate normal. Another key property that is important in our analysis is that linearly independent linear functions of multivariate normal random variables are multivariate normal random variables, i.e., if $Y = AX + b$, where A is an $n \times n$ non-singular matrix and b is a (column) n-vector of constants, then $Y \sim \mathcal{N}(A\mu+b, A\Sigma A^T)$.

A.2 Graph Theory

We reserve the symbol $\mathcal{G}, \widehat{\mathcal{G}}$ and $\widetilde{\mathcal{G}}$ to denote a graph. We denote by $\mathcal{G}'$ the graph formed by adding an edge to the graph $\mathcal{G}$. For any $S \subseteq V(\mathcal{G})$, the *cut* of the set of vertices S, denoted it by $\Phi_{\mathcal{G}}(S)$, is the weight of the edges that are present between S and $V \backslash S$. Since the vertex set is constant throughout this paper, we just write V to denote $V(\mathcal{G})$.

Definition 10 (Laplacian of graph)**.** *For an undirected graph $\mathcal{G}$, its edge adjacency matrix $E_{\mathcal{G}} \in \mathbb{R}^{\binom{n}{2} \times n}$ is defined as follow: the row corresponding to edge $e = (u, v)$ with weight w_e contains w_e in column u, $(-w_e)$ in column v, and 0 elsewhere. The Laplacian is then defined as $L_{\mathcal{G}} = E_{\mathcal{G}}^{\top} E_{\mathcal{G}}$.*

For any graph $\mathcal{G}$ with edge-adjacency matrix $E_{\mathcal{G}}$, consider its singular value decomposition $U \Sigma V^{\top}$. Then the effective resistance (also known as leverage score) of the edge e_i is defined as $\tau_i := e_i^{\top}(E_{\mathcal{G}}^{\top} E_{\mathcal{G}})^{\dagger} e_i = e_i^{\top} L_{\mathcal{G}}^{\dagger} e_i$. It is well known that by sampling the edges (rows of $E_{\mathcal{G}}$) of $\mathcal{G}$ according to its leverage score, it is possible to obtain a graph $\widetilde{\mathcal{G}}$ such that $(1-\varepsilon) L_{\widetilde{\mathcal{G}}} \preceq L_{\mathcal{G}} \preceq (1+\varepsilon) L_{\widetilde{\mathcal{G}}}$. Our analysis critically uses the fact that sampling by leverage scores is robust against overestimate of leverage scores.

Many interesting properties of the Laplacian of a graph follows from this representation. For example, Laplacian of a graph is positive semi-definite, i.e., all the eigenvalues are non-negative. For a set S of vertices, its cut-set is $\Phi_S(\mathcal{G}) = \mathbf{1}_S^T L_{\mathcal{G}} \mathbf{1}_S$.

We let $\lambda_i(\mathcal{G})$ denote the eigenvalues of $L_{\mathcal{G}}$ for $1 \leq i \leq n$. Next we present few lemmata that are useful in our analysis. We analyze multivariate Gaussian distributions that are linear combination of the Laplacian of two graphs. In order to analyze the two distributions, the corresponding covariance matrices must span the same subspace. The first lemma allows us to work on the same subspace, that is, the subspace orthogonal to $\mathsf{Span}\{\mathbf{1}\}$.

Lemma 11. *Let $0 = \lambda_1(\mathcal{G}) \leq \lambda_2(\mathcal{G}) \cdots \leq \lambda_n(\mathcal{G})$ be the n eigenvalues of $L_{\mathcal{G}}$. Then $\mathcal{G}$ is connected iff $\lambda_2 > 0$ and the kernel space of a connected graph is $\mathsf{Span}\{\mathbf{1}\}$. More generally, if a graph has k components, then the multiplicity of the eigenvalue 0 is k.*

The following two lemmata are useful in giving the upper bound while proving the DP of our sanitizer.

Lemma 12. *Let $\mathcal{G}$ and $\mathcal{G}'$ be two graphs, where $\mathcal{G}'$ is obtained from $\mathcal{G}$ by adding one edge joining two distinct vertices of $\mathcal{G}$. Then*

$$\lambda_2(\mathcal{G}) \leq \lambda_2(\mathcal{G}') \leq \lambda_2(\mathcal{G}) + 2.$$

Lemma 13. *Let $\mathcal{G}'$ be formed by adding an edge (u, v) to $\mathcal{G}$. For any vector $\mathbf{x} \in \mathbb{R}^n$, we have $\mathsf{Tr}(L_{\mathcal{G}'}) \leq \mathsf{Tr}(L_{\mathcal{G}}) + 2$.*

The following lemma is particularly useful in arguing that the lowest non-zero eigenvalues of all the graphs is bounded from below by a constant (which is the second smallest eigenvalue of an expander).

Lemma 14 (Eigenvalue Interlacing)**.** *Let $\mathcal{G}$ and $\mathcal{G}'$ be two graphs, where $\mathcal{G}'$ is obtained from $\mathcal{G}$ by adding one edge joining two distinct vertices of $\mathcal{G}$. Then*

$$\lambda_i(\mathcal{G}) \leq \lambda_i(\mathcal{G}') \leq \lambda_{i+1}(\mathcal{G}).$$

In particular, if $\mathcal{H}$ be a subgraph of $\mathcal{G}$, then $\lambda_i(\mathcal{H}) \leq \lambda_i(\mathcal{G}) \forall 1 \leq i \leq n$.

A.3 Differential Privacy

It is important to make a note that, unlike the traditional notions of privacy in cryptography, differential privacy is a measure of privacy loss. This allows one to quantify and compare various techniques and answer questions like, which is the most accurate mechanism for a given privacy loss, or which mechanism provides most privacy for a given error in the accuracy.

There are two settings in which a privacy preserving mechanism can be designed: non-interactive and interactive setting. In the *non-interactive* setting the curator publishes some sanitized database, and the database is not used further. The precise answers released by the curator might be affected by the sanitized database. As the database is never used again, the curator may as well destroy the database once it has published the sanitized form.

On the other hand, in the *interactive* setting, the curator sits between the analyst and the database, and queries are posed by the analyst in an adaptive manner. The curator must respond to these queries in a fashion so as to protect the privacy of database.

In this paper, all our algorithms is *non-interactive*. Differential privacy satisfies many required properties of any privacy guarantee. For example, differential privacy is immune to auxiliary information whether the auxiliary information is current or any side information an adversary can access in the future. We call this property the *robustness* of differential privacy. Another key property that differentially private mechanisms "composes" really well. This allows us to construct complicated mechanism from basic and simpler private mechanisms. We next cover both these properties in more details.

Robustness of Differential Privacy. One of the key features of differential privacy is that it is preserved under arbitrary post-processing, i.e., an analyst, without additional information about the private database, cannot compute a function that makes an output less differentially private. In other words,

Lemma 15 (Post-processing [17])**.** *Let $\mathfrak{M}(\mathbf{D})$ be an (α, β)-differential private mechanism for a database $\mathbf{D}$, and let h be any function, then any mechanism $\mathfrak{M}' := h(\mathfrak{M}(\mathbf{D}))$ is also (α, β)-differentially private for the same set of tasks.*

Proof. Let $\mathfrak{M}$ be a differentially private mechanism. Let $\mathsf{Range}(\mathfrak{M})$ denote the range the of $\mathfrak{M}$.Let R be the range of the function $h(\cdot)$. Without loss of generality, we assume that $h(\cdot) : \mathsf{Range}(\mathfrak{M}) \to \mathcal{R}$ is a deterministic function. This is because any randomized function can be decomposed into a convex combination of deterministic function, and a convex combination of differentially private mechanisms is differentially private. Fix any pair of neighbouring data-sets $\mathbf{DB}$ and $\widetilde{\mathbf{DB}}$ and an event $S \subseteq \mathcal{R}$. Let $T = \{y \in \mathsf{Range}(\mathfrak{M}) : f(r) \in S\}$. Then

$$\begin{aligned}\mathsf{Pr}[f(\mathfrak{M}(\mathbf{DB})) \in S] &= \mathsf{Pr}[\mathfrak{M}(\mathbf{DB}) \in T] \\ &\leq \exp(\alpha)\mathsf{Pr}[\mathfrak{M}(\widetilde{\mathbf{DB}}) \in T] + \beta \\ &= \exp(\alpha)\mathsf{Pr}[f(\mathfrak{M}(\widetilde{\mathbf{DB}})) \in S] + \beta\end{aligned}$$

as required. □

Composition. Before we begin, we discuss what does it mean by the term "composition" of differentially private mechanism. The composition that we consider covers the following two cases:

1. Repeated use of differentially private mechanism on the same database.
2. Repeated use of differentially private mechanism on different database that might contain information relating to a particular individual.

The first case covers the case when we wish to use the same mechanism multiple times while the second case covers the case of cumulative loss of privacy of a single individual whose data might be spread across many databases.

It is easy to see that the composition of pure differentially private mechanisms yields another pure differentially private mechanism, i.e., composition of an $(\alpha_1, 0)$-differentially private and an $(\alpha_2, 0)$-differentially private mechanism results in an $(\alpha_1 + \alpha_2, 0)$-differentially private mechanism. In other words, the privacy guarantee depreciates linearly with the number of compositions. In the case of approximate differential privacy, we can improve on the degradation of α parameter at the cost of slight depreciation of the β factor. We use this strengthening in our proofs. In our proofs of differential privacy, we prove that each row of the published matrix preserves (α_0, β_0)-differential privacy for some appropriate α_0, β_0, and then invoke a composition theorem by [17] to prove that the published matrix preserves (α, β)-differential privacy. The following theorem is the composition theorem that we use.

Theorem 16 (Composition theorem [17])**.** *Let $\alpha_0, \beta_0 \in (0, 1)$, and $\beta' > 0$. If $\mathfrak{M}_1, \cdots, \mathfrak{M}_\ell$ are each (α, β)-differential private mechanism, then the mechanism $\mathfrak{M}(\mathbf{D}) := (\mathfrak{M}_1(\mathbf{D}), \cdots, \mathfrak{M}_\ell(\mathbf{D}))$ releasing the concatenation of each algorithm is $(\alpha', \ell\beta + 0 + \beta')$-differentially private for $\alpha' < \sqrt{2\ell \ln(1/\beta')}\alpha_0 + 2\ell\alpha_0^2$.*

A proof of this theorem could be found in [17, Chapter 3].

B Missing Proofs of Section 3 and Section 5

Before we give a detail proof of our theorem, we make few important remarks regarding the result. Our bound (Theorem 3) states that the Laplacian of the graph is perturbed by an additive term, $\frac{w}{n}L_n$, due to privacy considerations. Given the choice of w, the scale of this additive perturbation is $\widetilde{O}(1/\sqrt{n})$. To better appreciate the bound, let us consider the following simple procedure where we obfuscate the edges through randomization and then run a non-private sparsification algorithm. This procedure would incur an approximation of the form $(1-\varepsilon)(L_n + L_{\mathcal{G}}) \preceq L_{\widetilde{\mathcal{G}}} \preceq (1+\varepsilon)(L_n + L_{\mathcal{G}})$, i.e., the scale of the additive perturbation matrix (which is still L_n) is $O(1)$. In Appendix E, we show that a solution based on the exponential mechanism will suffer from a similar problem. In light of this, our result yields a significant improvement over those alternatives. Finally, we note that returning a complete graph is not an option because the spectrum of $(1-\varepsilon)L_n$ might not be a lower bound on the spectrum of $\mathcal{G}$.

Theorem 3 does not put any restriction on $\mathcal{G}$. In particular, $\mathcal{G}$ can be a weighted graph with possibly very large spectrum compared to $L_n/\sqrt{n}$. Further, larger the weight, larger would be the value of MAX-CUT or $\Phi_{\mathcal{G}}(S)$. In fact, many naturally occurring graphs have large weights. One such graph arises in the discovery of metabolic pathways in genomic networks [26], where weights indicate strength between genes [15].

B.1 Utility Proof

We now prove the utility proof. We first prove that $\widetilde{\mathcal{G}}'$ is a graph that spectrally approximates $\bar{\mathcal{G}}$. The complication in the proof arise from the fact that D is not formed by the effective resistances of $\bar{\mathcal{G}}$, but rather the approximation of effective resistances of $\widehat{\mathcal{G}}$.

We construct matrix random variables corresponding to the graph $\bar{\mathcal{G}}$ defined in Algorithm 1. Let e_i be the edges in $\bar{\mathcal{G}}$ defined in Algorithm 1. Recall that $p_i = c\widetilde{\tau}_i\varepsilon^{-2}\log(n/\delta)$. We define the following random variables corresponding to each edge,

$$X_i := \begin{cases} \frac{e_i e_i^\top}{p_i} & \text{with probability } p_i \\ 0 & \text{with probability } 1-p_i \end{cases}.$$

Then $Y = \sum_i X_i = E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}}$. It is easy to see that $\mathsf{E}[Y] = L_{\bar{\mathcal{G}}}$, where expectation is over X_i.

Then, we can finish the proof by appealing to the matrix Bernstein inequality if we can bound $X_i \preceq R \cdot \mathsf{E}[Y]$. When $p_i = 1$, $X_i = e_i e_i^\top$ with probability 1, which is the same as selecting and summing $c\varepsilon^{-2}\log(n/\delta)$ random variables each equal to $\frac{e_i e_i^\top}{c\varepsilon^{-2}\log(n/\delta)}$, and hence $X_i \preceq \frac{L_{\bar{\mathcal{G}}}}{c\varepsilon^{-2}\log(n/\delta)}$. When $p_i < 1$, we can bound X_i as follows.

Proposition 17. *Let X_i, $\bar{\mathcal{G}}$ be as defined above. Then with probability $1-\delta$, $X_i \preceq O\left(\frac{\varepsilon^2}{c\log(n/\delta)}\right) L_{\bar{\mathcal{G}}}$.*

Proof. Since $\bar{\mathcal{G}}$ is a connected graph, the all ones vector, denoted $\mathbf{1}$, spans the kernel of the column space of $L_{\bar{\mathcal{G}}}$ [20]. Then, for an $x \in \mathbb{R}^n$, write $x = x_\parallel + x_\perp$, where $x_\parallel$ is in the span of column space of $L_{\widehat{\mathcal{G}}}$ and $x_\perp$ is in the span of $\mathbf{1}$. In other words, $\langle x_\parallel, \mathbf{1}\rangle = 0$. Let $U\Sigma^2 U^\top$ be the singular value decomposition of $L_{\bar{\mathcal{G}}}$. So,

$$\begin{aligned} x^\top L_{\bar{\mathcal{G}}} x &= (x_\parallel + x_\perp)^\top L_{\bar{\mathcal{G}}}(x_\parallel + x_\perp) = x_\parallel^\top L_{\bar{\mathcal{G}}} x_\parallel \\ &= y^\top U\Sigma^{-1}U^\top L_{\bar{\mathcal{G}}} U\Sigma^{-1}U^\top y \\ &= y^\top U\Sigma^{-1}U^\top U\Sigma^2 U^\top U\Sigma^{-1}U^\top y = \|y\|_2^2 \end{aligned}$$

for some y in the column space of $L_{\bar{\mathcal{G}}}$. Now with probability at least $1-\delta$,

$$\begin{aligned}
x^\top e_i e_i^\top x &= (x_\parallel + x_\perp)^\top e_i e_i^\top (x_\parallel + x_\perp) \\
&= x_\parallel^\perp e_i e_i^\top x_\parallel \\
&= y^\top U\Sigma^{-1}U^\top e_i e_i^\top U\Sigma^{-1}U^\top y \\
&\le \mathsf{Tr}(U\Sigma^{-1}U^\top e_i e_i^\top U\Sigma^{-1}U^\top)\|y\|_2^2 \\
&= e_i^\top L_{\widehat{\mathcal{G}}}^\dagger e_i \|y\|_2^2 \\
&\le C e_i^\top L_{\widehat{\mathcal{G}}}^\dagger e_i \|y\|_2^2 \\
&= C\widetilde{\tau}_i x^\top L_{\bar{\mathcal{G}}} x
\end{aligned}$$

for $C > 2$. Here the inequality follows from the subadditivity of trace norm, Fact 8, equation (2), and the final equality follows from the definition of $\widetilde{\tau}_i$. Since $p_i = c\widetilde{\tau}_i\varepsilon^{-2}\log(n/\delta)$, we have

$$X_i \preceq \frac{e_i e_i^\top}{p_i} \preceq \frac{e_i e_i^\top}{c\widetilde{\tau}_i\varepsilon^{-2}\log(n/\delta)} \preceq \frac{L_{\bar{\mathcal{G}}}}{c\varepsilon^{-2}\log(n/\delta)}$$

as required. □

Using matrix Bernstein inequality [53], we have

$$\mathsf{Pr}\left[(1-\varepsilon)Y \preceq L_{\bar{\mathcal{G}}} \preceq (1+\varepsilon)Y\right] \ge 1 - ne^{-c\varepsilon^{-2}\log(n/\delta)/3}$$

for large enough c. Now $E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}} = Y$ implies

$$(1-\varepsilon)E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}} \preceq L_{\bar{\mathcal{G}}} \preceq (1+\varepsilon)E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}}.$$

with probability $1-\delta$. That is, $E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}}$ is a private spectral approximation of $\bar{\mathcal{G}}$ with Laplacian $\left(\frac{w}{n}L_n + \left(1-\frac{w}{n}\right)L_{\mathcal{G}}\right)$. $E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}}$ can be the Laplacian of a dense graph. Using the result of [35], we have $(1-\varepsilon)L_{\widetilde{\mathcal{G}}} \preceq E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}} \preceq (1+\varepsilon)L_{\widetilde{\mathcal{G}}}$ and that $L_{\widetilde{\mathcal{G}}}$ has $O(n/\varepsilon^2)$ edges. In total, this implies that

$$\mathsf{Pr}\left[(1-\varepsilon)^2 L_{\widetilde{\mathcal{G}}} \preceq L_{\bar{\mathcal{G}}} \preceq (1+\varepsilon)^2 L_{\widetilde{\mathcal{G}}}\right] \ge 1 - 2\delta.$$

Since the solution of SDP-1 is $\lambda = c\sqrt{\frac{\log n\log(1/\delta)}{n\alpha^2}}$, we have

$$L_{\widetilde{\mathcal{G}}'} - c\sqrt{\frac{\log n\log(1/\delta)}{n\alpha^2}}L_n \preceq L_{\bar{\mathcal{G}}} \preceq L_{\widetilde{\mathcal{G}}'} + c\sqrt{\frac{\log n\log(1/\delta)}{n\alpha^2}}L_n.$$

Another application of matrix Bernstein inequality gives us

$$\mathsf{Pr}\left[L_{\widehat{\mathcal{G}}} - \rho\sqrt{\frac{\log n\log(1/\delta)}{n\alpha^2}}L_n \preceq L_{\widetilde{\mathcal{G}}'} \preceq L_{\widehat{\mathcal{G}}} + \rho\sqrt{\frac{\log n\log(1/\delta)}{n\alpha^2}}L_n\right] \ge 1-\delta. \tag{2}$$

for some small constant $\rho > 0$. Now by construction,

$$L_{\widehat{\mathcal{G}}} := \frac{w}{n}L_n + \left(1-\frac{w}{n}\right)L_{\mathcal{G}}$$

for $w = O(\sqrt{\frac{\log n\log(1/\delta)}{n\alpha^2}})$. Combining all these partial orderings gives us the accuracy bound.

B.2 Privacy Proof of Algorithm 1

In this section, we give a detail proof of the privacy guarantee of Algorithm 1. First note that computing $L_{\widetilde{\mathcal{G}}'}$ and R is $(\alpha/3, \beta/3)$-differentially private due to Gaussian mechanism [17] and Johnson-Lindenstrauss mechanism [9].

Lemma 18. *Let $M \in \mathbb{R}^{(m+n)\times t}$ be a random Gaussian matrix with every entries sampled i.i.d. from $\mathcal{N}(0, 1/t)$. Let $E_{\widehat{\mathcal{G}}}$ be a graph formed by overlaying the input graph with a complete graph with weights w/n, where $w = \frac{\sqrt{t\log(1/\beta)}}{\alpha}\log(1/\beta)$. Then*

$$B := \begin{pmatrix} E_{\widehat{\mathcal{G}}} & w\mathbb{I}\end{pmatrix} M$$

is (α, β)-differentially private.

Proof. We spot a subtle mistake in the previous analysis of privacy by [9, 54]. As in their proof, we also prove that each row of the published matrix preserves (α_0, β_0)-differential privacy for some appropriate α_0, β_0, and then invoke Theorem 16 to prove that the published matrix preserves (α, β)-differential privacy. Denote by $\widehat{A} = \begin{pmatrix} E_{\widehat{\mathcal{G}}} & w\mathbb{I}_m \end{pmatrix}$ and by $\widehat{A}' = \begin{pmatrix} E'_{\widehat{\mathcal{G}}} & w\mathbb{I}_m \end{pmatrix}$, where $\widehat{\mathcal{G}}$ and $\widehat{\mathcal{G}}'$ differ in one edge of weight 1. Note that both $\widehat{A}$ and $\widehat{A}'$ are full rank matrix because of the construction. This implies that the affine transformation of the multi-variate Gaussian is well-defined (both the covariance $(\widehat{A}\widehat{A}^\top)^{-1}$ has a full rank and the $\Delta(\widehat{A}\widehat{A}^\top)$ is non-zero). That is, the PDF of the distributions of the columns, corresponding to $\widehat{A}$ and $\widehat{A}'$, is just a linear transformation of $\mathcal{N}(\mathbf{0}, \mathbb{I})$. Let $y \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$ be a column vector. Let x is sampled either from $\widehat{A}y$ or $\widehat{A}'y$. Then the corresponding PDFs are as follows:

$$\mathsf{PDF}_{\widehat{A}y}(x) = \frac{e^{(-\frac{1}{2}x(\widehat{A}\widehat{A}^\top)^{-1}x^\top)}}{\sqrt{(2\pi)^d \Delta(\widehat{A}\widehat{A}^\top)}}, \quad \mathsf{PDF}_{\widehat{A}'y}(x) = \frac{e^{(-\frac{1}{2}x(\widehat{A}'\widehat{A}'^\top)^{-1}x^\top)}}{\sqrt{(2\pi)^d \Delta(\widehat{A}'\widehat{A}'^\top)}}$$

Let

$$\alpha_0 = \frac{\alpha}{\sqrt{4t \ln(1/\beta)\log(1/\beta)}}, \quad \beta_0 = \frac{\beta}{2t},$$

where $t = n$ in the case of PRIVATE-SPARSIFY. We prove that every row of the published matrix is (α_0, β_0)-differentially private; the theorem follows from Theorem 16. Let x be sampled either from $\mathcal{N}(0, \widehat{A}\widehat{A}^\top)$ or $\mathcal{N}(0, \widehat{A}'\widehat{A}'^\top)$. It is straightforward to see that the combination of Claim 19 and Claim 20 below proves differential privacy for a row of published matrix. The lemma then follows by an application of Theorem 16 and our choice of α_0 and β_0. We prove a stronger guarantee than what is required, i.e., we prove privacy when $\widehat{A}' - \widehat{A} = uv^\top$ for unit vectors u, v. Privacy for various theorems in this paper can be realized by setting u to be a standard basis vector.

Claim 19. *Let $\widehat{A}$, $\widehat{A}'$ and α_0 be as defined above. Then*

$$e^{-\alpha_0} \leq \sqrt{\frac{\Delta(\widehat{A}\widehat{A}^\top)}{\Delta(\widehat{A}'\widehat{A}'^\top)}} \leq e^{\alpha_0}.$$

Proof. The claim follows simply as in [9]. More concretely, we have $\Delta(\widehat{A}\widehat{A}^\top) = \prod_i \sigma_i^2$, where $\sigma_1 \geq \cdots \geq \sigma_m \geq \sigma_{\mathsf{min}}(\widehat{A})$ are the singular values of $\widehat{A}$. Let $\widetilde{\sigma}_1 \geq \cdots \geq \widetilde{\sigma}_m \geq \sigma_{\mathsf{min}}(\widehat{A}')$ be its singular value for $\widehat{A}'$. The matrix $E = \widehat{A}' - \widehat{A}$ has only one singular value.

Since the singular values of $\widehat{A} - \widehat{A}'$ and $\widehat{A}' - \widehat{A}$ are the same, Lidskii's theorem (Theorem 9) gives $\sum_i (\sigma_i - \widetilde{\sigma}_i) \leq 1$. Therefore, with probability $1 - \beta_0$,

$$\sqrt{\prod_{i:\widetilde{\sigma}_i \geq \sigma_i} \frac{\widetilde{\sigma}_i^2}{\sigma_i^2}} = \prod_{i:\widetilde{\sigma}_i \geq \sigma_i} \left(1 + \frac{\widetilde{\sigma}_i - \sigma_i}{\sigma_i}\right) \leq \exp\left(\frac{\varepsilon}{32\sqrt{t\log(2/\beta)}\log(t/\beta)} \sum_{i:\widetilde{\sigma}_i \geq \sigma_i} (\widetilde{\sigma}_i - \sigma_i)\right) \leq e^{\alpha_0/2},$$

where the first inequality follows from the fact that $1 + x \leq e^x$ for $x \geq 0$ and the fact that $\sigma_i \geq \sigma_{\mathsf{min}}$.

The case for all $i \in [m]$ when $\widetilde{\sigma}_i \leq \sigma_i$ follows similarly as the singular values of E and $-E$ are the same. This completes the proof of Claim 19. □

Claim 20. *Let $\widehat{A}, \alpha_0$, and β_0 be as defined earlier. Let $y \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$. If x is sampled either from $\widehat{A}y$ or $\widehat{A}'y$, then we have*

$$\mathsf{Pr}\left[\left|x^\top(\widehat{A}\widehat{A}^\top)^{-1}x - x^\top(\widehat{A}'\widehat{A}'^\top)^{-1}x\right| \leq \alpha_0\right] \geq 1 - \beta_0.$$

Proof. Without any loss of generality, we can assume $x = \widehat{A}y$. The case for $x = \widehat{A}'y$ is analogous. Let $\widehat{A}' - \widehat{A} = uv^\top$. Then

$$\begin{aligned} x^\top(\widehat{A}\widehat{A}^\top)^{-1}x - x^\top(\widehat{A}'\widehat{A}'^\top)^{-1}x &= x^\top(\widehat{A}\widehat{A}^\top)^{-1}(\widehat{A}'\widehat{A}'^\top)(\widehat{A}'\widehat{A}'^\top)^{-1}x - x^\top(\widehat{A}'\widehat{A}'^\top)^{-1}x \\ &= x^\top(\widehat{A}\widehat{A}^\top)^{-1}(\widehat{A} + uv^\top)(\widehat{A}^\top + vu^\top)(\widehat{A}'\widehat{A}')^{-1}x - x^\top(\widehat{A}'\widehat{A}')^{-1}x \\ &= x^\top\left[(\widehat{A}\widehat{A}^\top)^{-1}(\widehat{A}uv^\top + vu^\top\widehat{A}'^\top + uu^\top)(\widehat{A}'\widehat{A}'^\top)^{-1}\right]x. \end{aligned}$$

Previous analysis failed to bound the last term due to $uu^\top$. Using the singular value decomposition of $\widehat{A} = U_A \Sigma_A V_A^\top$ and $\widehat{A}' = \widetilde{U}_A \widetilde{\Sigma}_A \widetilde{V}_A^\top$, we have

$$\left(x^\top (U_A \Sigma_A^{-1} V_A^\top) u\right) \left(v^\top (\widetilde{U}_A \widetilde{\Sigma}_A^{-2} \widetilde{U}_A^\top) x\right) + \left(x^\top (U_A \Sigma_A^{-2} U_A^\top) v\right) \left(u^\top (\widetilde{V}_A \widetilde{\Sigma}_A^{-1} \widetilde{U}_A^\top) x\right) + \left(x^\top (U_A \Sigma_A^{-2} U_A^\top) u\right) \left(u^\top (\widetilde{U}_A \widetilde{\Sigma}_A^{-2} \widetilde{U}_A^\top) x\right)$$

Since $x \sim \widehat{A}y$, where $y \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$, we can write the above expression as $\tau_1\tau_2 + \tau_3\tau_4$, where

$$\tau_1 = \left(y^\top \widehat{A}^\top (U_A \Sigma_A^{-1} V_A^\top) u\right), \quad \tau_2 = \left(v^\top (\widetilde{U}_A \widetilde{\Sigma}_A^{-2} \widetilde{U}_A^\top) \widehat{A} y\right), \quad \tau_3 = \left(y^\top \widehat{A}^\top (U_A \Sigma_A^{-2} U_A^\top) v\right)$$
$$\tau_4 = \left(u^\top (\widetilde{V}_A \widetilde{\Sigma}_A^{-1} \widetilde{U}_A^\top) \widehat{A} y\right), \quad \tau_5 = \left(y^\top \widehat{A}^\top (U_A \Sigma_A^{-2} U_A^\top) u\right), \quad \tau_6 = \left(u^\top (\widetilde{U}_A \widetilde{\Sigma}_A^{-2} \widetilde{U}_A^\top) \widehat{A} y\right).$$

From the construction, we have $\|\widetilde{\Sigma}_A\|_2, \|\Sigma_A\|_2 \geq w$. Using the SVD of $\widehat{A}$ and $\widehat{A} - \widehat{A}' = vu^\top$, and that every term τ_i in the above expression is a linear combination of a Gaussian, i.e., each term is distributed as per $\mathcal{N}(0, \|\tau_i\|^2)$, we calculate $\|\tau_i\|$ as below.

$$\begin{aligned}
\|\tau_1\|_2 &= \|(V_A \Sigma_A U_A^\top)(U_A \Sigma_A^{-1} V_A^\top) u\|_2 \leq \|u\|_2 \leq 1, \\
\|\tau_2\|_2 &= \|v^\top (\widetilde{U}_A \widetilde{\Sigma}_A^{-2} \widetilde{U}_A^\top)(\widetilde{U}_A \widetilde{\Sigma}_A \widetilde{V}_A^\top - vu^\top)\|_2 \\
&\leq \|v^\top (\widetilde{U}_A \widetilde{\Sigma}_A^{-2} \widetilde{U}_A^\top) \widetilde{U}_A \widetilde{\Sigma}_A \widetilde{U}_A^\top\|_2 + \|v^\top (\widetilde{U}_A \widetilde{\Sigma}_A^{-2} \widetilde{U}_A^\top) vu^\top\|_2 \leq \frac{1}{w} + \frac{1}{w^2}, \\
\|\tau_3\|_2 &= \|(V_A \Sigma_A U_A^\top)(U_A \Sigma_A^{-2} U_A^\top) v\|_2 \leq \|\Sigma_A^{-1}\|_2 \leq \frac{1}{w}, \\
\|\tau_4\|_2 &= \|u^\top (\widetilde{V}_A \widetilde{\Sigma}_A^{-1} \widetilde{U}_A^\top)(\widetilde{U}_A \widetilde{\Sigma}_A \widetilde{V}_A^\top - vu^\top)\|_2 \\
&\leq \|u^\top (\widetilde{V}_A \widetilde{\Sigma}_A^{-1} \widetilde{U}_A^\top)(\widetilde{U}_A \widetilde{\Sigma}_A \widetilde{V}_A^\top\|_2 + \|u^\top (\widetilde{V}_A \widetilde{\Sigma}_A^{-1} \widetilde{U}_A^\top) v\|_2 \leq 1 + \frac{1}{w} \\
\|\tau_5\|_2 &= \|(V_A \Sigma_A U_A^\top)(U_A \Sigma_A^{-2} U_A^\top) u\|_2 \leq \|\Sigma_A^{-1}\|_2 \leq \frac{1}{w}, \\
\|\tau_6\|_2 &= \|u^\top (\widetilde{U}_A \widetilde{\Sigma}_A^{-2} \widetilde{U}_A^\top)(\widetilde{U}_A \widetilde{\Sigma}_A \widetilde{V}_A^\top - vu^\top)\|_2 \\
&\leq \|u^\top (\widetilde{U}_A \widetilde{\Sigma}_A^{-2} \widetilde{U}_A^\top) \widetilde{U}_A \widetilde{\Sigma}_A \widetilde{U}_A^\top\|_2 + \|u^\top (\widetilde{U}_A \widetilde{\Sigma}_A^{-2} \widetilde{U}_A^\top) vu^\top\|_2 \leq \frac{1}{w} + \frac{1}{w^2}.
\end{aligned}$$

Using the concentration bound on the Gaussian distribution, each term, $\tau_1, \tau_2, \tau_3, \tau_4, \tau_5$, and τ_6, is less than $\|\tau_i\| \ln(4/\beta_0)$ with probability $1 - \beta_0/2$. The second claim follows from the following inequality:

$$\mathsf{Pr}\left[\left|x^\top (\widehat{A}\widehat{A}^\top)^{-1} x - x^\top (\widehat{A}'^\top \widehat{A}')^{-1} x\right| \leq 2\left(\frac{1}{w} + \frac{1}{w^2} + \frac{1}{w^3}\right) \ln(4/\beta_0) \leq \alpha_0\right] \geq 1 - \beta_0,$$

where the second inequality follows from the choice of w. □

Combining the two claims proves the lemma. □

B.3 Finishing the Proof of Theorem 3

In Lemma 21, we prove the algorithm is (α, β)-differentially private. We prove that the effective resistance computed in PRIVATE-SPARSIFY is an overestimate in Lemma 22.

Lemma 21. *$L_{\widetilde{\mathcal{G}}}$ outputted by* PRIVATE-SPARSIFY *is (α, β)-differentially private with respect to the edge-level privacy.*

Proof. We first prove the privacy guarantee. There are two times when the graph is used in the algorithm PRIVATE-SPARSIFY in Step 3. We produce a sketch of the form $B = AM$, where M is a random Gaussian matrix and $A = \begin{pmatrix} E_{\widehat{\mathcal{G}}} & w\mathbb{I} \end{pmatrix}$. B is $(\alpha/3, \beta/3)$-differentially private for the choice of w due to Lemma 18. The second place where we use the graph is $NE_{\widehat{\mathcal{G}}}$. Again, here by the choice of w, we have $(\alpha/3, \beta/3)$-differentially private due to [9, 54]. Finally, we use the graph to compute $\widetilde{\mathcal{G}}'$. This is private due to Gaussian mechanism [17]. Using the composition theorem [18] and post-processing property (Lemma 15) of differential privacy, the result follows. □

We next prove that $\widetilde{\tau}_i$ is an overestimate of the original τ_i corresponding to the edges in $\widehat{\mathcal{G}}$.

Lemma 22. *Let τ_i be the true leverage score of the i-th row of $E_{\bar{\mathcal{G}}}$. Then with probability $1-\delta$, $\widetilde{\tau}_i \geq \tau_i$, where $\widetilde{\tau}_i$ is as computed in* PRIVATE-SPARSIFY. *Furthermore, $\sum_{i=1}^{\binom{n}{2}} \widetilde{\tau}_i = O(n^2)$.*

Proof. We first prove the second part of the lemma. Let $M = \begin{pmatrix} M_1 \\ M_2 \end{pmatrix}$, where $M_1 \in \mathbb{R}^{n\times n}$ and $M_2 \in \mathbb{R}^{m\times n}$. Then

$$\begin{aligned}
\sum_i \widetilde{\tau}_i &= \mathsf{Tr}\left(\begin{pmatrix} E_{\widehat{\mathcal{G}}} & w\mathbb{I} \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \end{pmatrix} L_G^{\dagger} \begin{pmatrix} M_1^\top & M_2^\top \end{pmatrix} \begin{pmatrix} E_{\widehat{\mathcal{G}}}^\top \\ w\mathbb{I} \end{pmatrix} \right) \\
&\leq \mathsf{Tr}\left(\begin{pmatrix} L_{\widehat{\mathcal{G}}} & 0 \\ 0 & w^2\mathbb{I} \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \end{pmatrix} L_{\widehat{\mathcal{G}}}^{\dagger} \begin{pmatrix} M_1^\top & M_2^\top \end{pmatrix} \right) \\
&= \mathsf{Tr}\left(\begin{pmatrix} L_{\widehat{\mathcal{G}}} & 0 \\ 0 & w^2\mathbb{I} \end{pmatrix} \begin{pmatrix} M_1 L_{\widehat{\mathcal{G}}}^{\dagger} M_1^\top & 0 \\ 0 & M_2 L_G^{\dagger} M_2^\top \end{pmatrix} \right) \\
&= \mathsf{Tr}\left(\begin{pmatrix} L_{\widehat{\mathcal{G}}} M_1 L_{\widehat{\mathcal{G}}}^{\dagger} M_1^\top & 0 \\ 0 & w^2 M_2 L_G^{\dagger} M_2^\top \end{pmatrix} \right) \\
&= \mathsf{Tr}\left(M_1^\top L_{\widehat{\mathcal{G}}} M_1 L_{\widehat{\mathcal{G}}}^{\dagger} \right) + w^2 \mathsf{Tr}\left(M_2^\top M_2 L_{\widehat{\mathcal{G}}}^{\dagger} \right) \\
&= O(n^2).
\end{aligned}$$

The last equality holds because $\mathsf{Tr}(M_1^\top L_{\widehat{\mathcal{G}}} M_1 L_{\widehat{\mathcal{G}}}^{\dagger})$ can be written as $\mathsf{Tr}(U_{\widehat{\mathcal{G}}} M_1^\top U_{\widehat{\mathcal{G}}}^\top \Sigma_{\widehat{\mathcal{G}}} U_{\widehat{\mathcal{G}}} M_1 U_{\widehat{\mathcal{G}}}^\top \Sigma_{\widehat{\mathcal{G}}}^{\dagger})$, where $L_{\widehat{\mathcal{G}}} = U_{\widehat{\mathcal{G}}}^\top \Sigma_{\widehat{\mathcal{G}}} U_{\widehat{\mathcal{G}}}$. Using the symmetric property of Gaussian distribution, this is identically distributed as $\mathsf{Tr}(M_1^\top \Sigma_{\widehat{\mathcal{G}}} M_1 \Sigma_{\widehat{\mathcal{G}}}^{\dagger})$. Using the concentration of Gaussian vector, we have that with probability at least $1-\beta$, $\mathsf{Tr}(M_1^\top \Sigma_{\widehat{\mathcal{G}}} M_1 \Sigma_{\widehat{\mathcal{G}}}^{\dagger}) = O(n)$. Similarly, for the second term, we know that $L_{\widehat{\mathcal{G}}} \succeq w^2\mathbb{I}$, which in turn implies that $L_{\widehat{\mathcal{G}}}^{\dagger} \preceq w^{-2}\mathbb{I}$. Let $R \sim \mathcal{N}(0,1)^{\binom{n}{2}\times n}$. We

$$w^2 \mathsf{Tr}\left(M_2^\top M_2 L_{\widehat{\mathcal{G}}}^{\dagger} \right) = w^2 \mathsf{Tr}(\frac{1}{n} R^\top R L_{\widehat{\mathcal{G}}}^{\dagger}) \leq w^2 \frac{1}{n} \times \binom{n}{2} \times n \times \frac{1}{w^2} = \binom{n}{2}.$$

Note that this defines the number of edges in $E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}}$; however, crucially, as we show later, we achieve multiplicative spectral approximation while preserving differential privacy. Both these points are crucial in achieving our final bounds as we use the fact that differential privacy is preserved under arbitrary post-processing by running the spectral sparsification algorithm on the private Laplacian $E_{\bar{\mathcal{G}}}^\top D E_{\bar{\mathcal{G}}}$ to reduce the number of edges to $O(n/\varepsilon^2)$ as claimed.

Let e_i be the i-th row of $E_{\bar{\mathcal{G}}}$. Then

$$\begin{aligned}
\widetilde{\tau}_i &= B_i {L_{\widehat{\mathcal{G}}}}^{\dagger} B_i^\top \\
&= e_i M_1 {L_{\widehat{\mathcal{G}}}}^{\dagger} M_1^\top e_i + w^2 M_2 {L_{\widehat{\mathcal{G}}}}^{\dagger} M_2^\top \\
&\geq \frac{1}{c} e_i {L_{\bar{\mathcal{G}}}}^{\dagger} e_i = \frac{1}{c}\tau_i,
\end{aligned}$$

where the last inequality follows using the same argument as above. This completes the proof. □

B.4 Proof of Theorem 7

Proof. We first prove cut sparsification. Fix a set of nodes $S \subseteq [n]$. We have

$$\mathsf{E}[\mathbf{1}_S^\top L_{\widehat{\mathcal{G}}} \mathbf{1}_S] = \alpha \mathbf{1}_S^\top L_{\mathcal{G}} \mathbf{1}, \qquad \mathsf{Var}(\mathbf{1}_S^\top L_{\widehat{\mathcal{G}}} \mathbf{1}_S) = \Theta(n(n-s)),$$

where $s = |S|$. Using Chernoff bound,

$$\mathsf{Pr}\left[\left| \mathbf{1}_S^\top L_{\widehat{\mathcal{G}}} \mathbf{1}_S - \alpha \cdot \mathbf{1}_S^\top L_{\mathcal{G}} \mathbf{1}_S \right| \leq \sqrt{s(n-s)\log(2^n/\delta)} \right] \geq 1 - \delta/2^n.$$

Using union bound over all 2^n possible choices of S, we have with probability $1-\delta$,

$$\mathbf{1}_S^\top L_{\widehat{\mathcal{G}}}\mathbf{1}_S - \sqrt{n\mathbf{1}_S^\top L_n \mathbf{1}_S \log(1/\delta)} \leq \alpha \cdot \mathbf{1}_S^\top L_{\mathcal{G}}\mathbf{1}_S \leq \mathbf{1}_S^\top L_{\widehat{\mathcal{G}}}\mathbf{1}_S + \sqrt{n\mathbf{1}_S^\top L_n \mathbf{1}_S \log(1/\delta)}. \quad (3)$$

Using [35], we have $(1-\varepsilon)L_{\widetilde{\mathcal{G}}} \preceq L_{\widehat{\mathcal{G}}} \preceq (1+\varepsilon)L_{\widetilde{\mathcal{G}}}$ and $\widetilde{\mathcal{G}}$ has $O(n/\varepsilon^2)$ edges. Combining this with equation (3), cut sparsification follows. The privacy proof follows by simple case analysis. This completes the proof of Theorem 7. □

C Applications of Our Results

We presented differentially private algorithms for cut sparsifier and spectral sparsifier. In practice, cut sparsifier and spectral sparsifier are used as black-box to improve run-time efficiency by solving various graph related algorithms on sparse graphs instead of the original dense graph. For example, cut sparsifier can be used to compute approximations for maximum cut, sparsest cut, and maximum flow, etc. Since spectral sparsification also preserve spectral properties of the Laplacian of the graph, on top of cut related problems, they can be used to approximately solve linear systems over the Laplacian of G, and to approximate spectral clusterings, random walk properties, etc.

Our results can be applied to all these settings. This is because differential privacy is preserved under post-processing. Therefore, once we publish a differentially private representation of a graph, it can be used as black-box in any graph algorithm.

C.1 Applications of Private Spectral Sparsification

Spectral sparsification is used as a first step for solving many graph related problems when the graph is dense and the complexity of the graph problems depends on the number of edges, such as cut queries. In many cases, it also allows one to approximately and efficiently infer many structural information about the graph that could be inferred using the spectral properties of the graph, for example, connectivity.

In what follows, we enumerate the implications of our results for answering all possible cut-queries, privately computing the set of vertices contributing in the approximate solution to MAX-CUT and SPARSE-CUT. All these results can also be achieved using cut-sparsification albeit with larger accuracy loss (see Appendix 5). We show the strength of private spectral sparsification over private cut sparsification by also privately computing Laplacian eigenmap, a popular approach used in manifold learning.

In addition to constructing private algorithms for many problems that have no prior such algorithms, our approach also leads to an efficient construction. This is because most of these applications runs in time dependent on the number of edges.

Cut Queries. Spectral sparsifier also preserves all cuts in the graph. Consequently, the first application of spectral sparsifier is that we can compute all possible cut queries with error that scales linearly with n.

Theorem 23. *Given an n vertices m edges graph $\mathcal{G}$, there is an efficient (α, β)-differentially private algorithm that preprocess the graph to output another graph $\widetilde{\mathcal{G}}$. Given a cut query in the form of set of vertices S, one can use $\widetilde{\mathcal{G}}$ to output the size of the cut with additive error $\tau \leq \widetilde{O}\left(\sqrt{\frac{s(n-s)}{\alpha^2}}\right)$ with probability at least $9/10$. Furthermore, answering a cut query, S, takes $O(|S|)$ amortized time.*

Proof. We first present the algorithm and then prove that it achieves the bound claimed in the theorem. The algorithm is as follows:

Algorithm. Given a graph $\mathcal{G}$, compute a sparse graph $\widetilde{\mathcal{G}}$ using Theorem 39. Then given a vector $\mathbf{1}_S$, compute $\frac{1}{1-\frac{w}{n}}\left(\mathbf{1}_S^\top L_{\widetilde{G}}\mathbf{1}_S - \frac{ws(n-s)}{n}\right)$.

We now argue the correctness of our algorithm. Using Theorem 39, we have

$$\begin{aligned}\mathbf{1}_S^\top L_{\widetilde{G}} \mathbf{1}_S &\leq (1+\varepsilon)\left(\left(1-\frac{w}{n}\right)\Phi_S(\mathcal{G}) + \frac{w}{n}\mathbf{1}_S^\top L_n \mathbf{1}_S\right) \\ &= (1+\varepsilon)\left(1-\frac{w}{n}\right)\Phi_S(\mathcal{G}) + (1+\varepsilon)\frac{ws(n-s)}{n}\end{aligned}$$

Therefore,

$$\frac{1}{1-\frac{w}{n}}\left(\mathbf{1}_S^\top L_{\widetilde{G}} \mathbf{1}_S - \frac{ws(n-s)}{n}\right) \leq (1+\varepsilon)\Phi_S(\mathcal{G}) + \frac{\varepsilon ws}{1-\frac{w}{n}}$$

This completes the proof of Theorem 4. □

Our algorithm answers a cut query, S, in $O(|S|)$ amortized time. This is a vast improvement over the previous best algorithm for answering cut queries [9, 54], which takes $O(n|S|)$ amortized cost. This improvement is especially significant when the graph is large, a more common phenomenon in practice. For example, social graphs like Facebook can have as many as 10^9 vertices.

MAX-CUT. Another problem one can solve using our spectral sparsifier is that of MAX-CUT. Here, given a graph $\mathcal{G} = (V, E)$ on a vertex set V, the goal is to output a set of vertices $S \subseteq V$ that maximizes the value of $\Phi_S(\mathcal{G})$. It is well known that MAX-CUT is NP-hard. However, Goemans and Williamson gave an elegant polynomial time algorithm for computing $\zeta_{GW} - \eta$ approximation to MAX-CUT for an arbitrary constant $\eta > 0$, where

$$\zeta_{GW} := \min_{0<\theta<\pi} \frac{\theta/\pi}{(1-\cos\theta)/2} \approx 0.87856. \tag{4}$$

It is known that even approximation within a factor of $\zeta_{GW} + \rho$, for all $\rho > 0$ is NP-hard [31] under the unique games conjecture. [21] showed the following:

Theorem 24 ([21])**.** *Let $\eta > 0$ be an arbitrary small constant. For an n-vertex graph $\mathcal{G}$, there is a polynomial-time algorithm that produces a set of nodes S satisfying $\Phi_S(\mathcal{G}) = (\zeta_{GW} - \eta)\mathsf{OPT}_{\mathsf{max}}$, where $\mathsf{OPT}_{\mathsf{max}}$ is the optimal max-cut.*

We show the following:

Theorem 25. *For an n-vertex graph $\mathcal{G} = (V, E)$, there is a polynomial-time algorithm that is (α, β)-differentially private with respect to the edge level privacy and produces a set of nodes $S \subseteq V$ satisfying*

$$\Phi_S(\mathcal{G}) \geq (\zeta_{GW} - \eta)\left(\frac{1-\varepsilon}{1+\varepsilon}\right)\mathsf{OPT}_{\mathsf{max}} - O\left(\frac{|S|\sqrt{n}}{\alpha}\right)$$

with probability at least $9/10$. Here ζ_{GW} is as defined in equation (4), $\mathsf{OPT}_{\mathsf{max}}$ is the optimal value of MAX-CUT *and $\Phi_S(\mathcal{G})$ is the size of cut for vertex set in $S \subseteq V$.*

Proof. We first present the algorithm and then prove that it achieves the bound claimed in the theorem. The algorithm is as follows:

Algorithm. The algorithm first computes a private sparse graph using Algorithm 1, then runs the SDP based algorithm of [21] on the private sparse graph, and output the set of vertices S outputted by that algorithm.

We next argue the correctness of our algorithm. Let $\widetilde{S}_{\mathsf{OPT}}$ be the solution of MAX-CUT on graph $\widetilde{\mathcal{G}}$ and S_{OPT} be the solution of MAX-CUT on the graph $\mathcal{G}$. In other words,

$$\widetilde{S}_{\mathsf{OPT}} = \operatorname*{argmax}_{S\subseteq V} \mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}} \mathbf{1}_S \quad \text{and} \quad S_{\mathsf{OPT}} = \operatorname*{argmax}_{S\subseteq V} \mathbf{1}_S^\top L_{\mathcal{G}} \mathbf{1}_S.$$

From Theorem 24, we know that

$$\Phi_S(\widetilde{\mathcal{G}}) \geq (0.87856 - \eta) \max_{S\subseteq V} \mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}} \mathbf{1}_S.$$

Further, since $\widetilde{\mathcal{G}}$ is a spectral sparsification, Theorem 3 implies that

$$(1-\varepsilon)\, x^\top \left(\frac{1}{\sqrt{n}} L_n + \left(1-\frac{1}{\sqrt{n}}\right) L_{\mathcal{G}}\right) x \le x^\top L_{\widetilde{\mathcal{G}}} x \le (1+\varepsilon)\, x^\top \left(\frac{1}{\sqrt{n}} L_n + \left(1-\frac{1}{\sqrt{n}}\right) L_{\mathcal{G}}\right) x.$$

holds simultaneously for $\forall x \in \mathbb{R}^n$ with probability $9/10$. Applying Theorem 3 twice, once with $\widetilde{S}_{\mathsf{OPT}}$ and S_{OPT}, we have the desired claim. □

SPARSEST-CUT. Another problem that is considered in graph theory is the problem of sparsest cut, SPARSE-CUT. Here, given a graph $\mathcal{G} = (V, E)$ on a vertex set V, the goal is to output a set of vertices S that minimizes the value $\frac{\Phi_S(\mathcal{G})}{|S|(n-|S|)}$. For sparsest cut, [5] showed an $O(\sqrt{\log n})$ approximation.

Theorem 26 ([5])**.** *For an n-vertex graph $\mathcal{G}$, there is a polynomial-time algorithm that produces a set of nodes S satisfying $\Phi_S(\mathcal{G}) = O(\sqrt{\log n})\mathsf{OPT}_{\mathsf{sparsest}}$, where $\mathsf{OPT}_{\mathsf{sparsest}}$ is the optimal sparsest cut.*

Combining Theorem 26 result with Theorem 3, we immediately get the following:

Theorem 27. *For an n-vertex graph $\mathcal{G} = (V, E)$, there is a polynomial-time algorithm that is (α, β)-differentially private with respect to the edge level privacy and produces a set of nodes $S \subseteq V$ satisfying*

$$\Phi_S(\mathcal{G}) \le O(\sqrt{\log n}) \left(\frac{1+\varepsilon}{1-\varepsilon}\right) \mathsf{OPT}_{\mathsf{sparsest}} + O\left(\frac{\log^2 n}{\varepsilon\sqrt{n}}\right),$$

with probability at least $9/10$. Here $\mathsf{OPT}_{\mathsf{sparsest}}$ is the optimal value of SPARSE-CUT *and $\Phi_S(\mathcal{G})$ is the size of cut for vertex set in $S \subseteq V$.*

Proof. The algorithm is as follows:

Algorithm. The algorithm first computes a private sparse graph using Algorithm 1, then runs the SDP based algorithm of [5] on the private sparse graph, and output the set of vertices S outputted by that algorithm.

We next argue the correctness of our algorithm. Let $\widetilde{S}_{\mathsf{OPT}}$ be the solution of SPARSEST-CUT on graph $\widetilde{\mathcal{G}}$ and S_{OPT} be the solution of SPARSEST-CUT on the graph $\mathcal{G}$. In other words,

$$\widetilde{S}_{\mathsf{OPT}} = \operatorname*{argmin}_{S \subseteq V} \left(\frac{\mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}} \mathbf{1}_S}{|S|(n-|S|)}\right) \quad \text{and} \quad S_{\mathsf{OPT}} = \operatorname*{argmin}_{S \subseteq V} \left(\frac{\mathbf{1}_S^\top L_{\mathcal{G}} \mathbf{1}_S}{|S|(n-|S|)}\right).$$

From Theorem 26, we know that

$$\Phi_S(\widetilde{\mathcal{G}}) \le O(\sqrt{\log n}) \min_{S \subseteq V} \left(\frac{\mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}} \mathbf{1}_S}{|S|(n-|S|)}\right).$$

Since $\widetilde{\mathcal{G}}$ is a spectral sparsification, Theorem 3 implies that

$$(1-\varepsilon)\, x^\top \left(\frac{1}{\sqrt{n}} L_n + \left(1-\frac{1}{\sqrt{n}}\right) L_{\mathcal{G}}\right) x \le x^\top L_{\widetilde{\mathcal{G}}} x \le (1+\varepsilon)\, x^\top \left(\frac{1}{\sqrt{n}} L_n + \left(1-\frac{1}{\sqrt{n}}\right) L_{\mathcal{G}}\right) x.$$

holds simultaneously for $\forall x \in \mathbb{R}^n$. Applying Theorem 3 twice, once with $\widetilde{S}_{\mathsf{OPT}}$ and S_{OPT}, we have the desired claim. □

Edge Expansion. The EDGE-EXPANSION ratio of a cut is the ratio of the weight of edges across the cut to the total weighted degree of edges incident to the side that is smaller with respect to total weighted degree. The edge expansion ratio of a graph is the minimum edge expansion ratio of any cut. [5] showed the following approximation result for edge expansion.

Lemma 28 ([5])**.** *For any constant $c > 0$ there is a polynomial-time algorithm that, given any regular weighted graph and a number $\gamma > 0$ behaves as follows. If the graphs has a c-balanced cut of edge expansion ratio less than γ then the algorithm outputs a $c/2$-balanced cut of edge expansion ratio $2\sqrt{\gamma}$. If the graph does not have such a cut, the algorithm finds a set of at least $(1 - c/2)n$ nodes such that the induced subgraph (with deleted edges replaced by self-loops) on them has edge expansion ratio at least 2γ.*

Using the fact that, up to a factor 2, computing the sparsest cut is the same as computing the *edge expansion* of the graph, we have the following corollary.

Theorem 29. *For an n-vertex graph $\mathcal{G} = (V, E)$, there is a polynomial-time algorithm that is (α, β)-differentially private with respect to the edge level privacy and produces a set of nodes $S \subseteq V$ satisfying*

$$\mathsf{OUT} \leq O(\sqrt{\log n}) \left(\frac{1+\varepsilon}{1-\varepsilon}\right) \mathsf{OPT}_{\mathsf{edge}} + O\left(\frac{\log^2 n}{\varepsilon\sqrt{n}}\right),$$

with probability at least $9/10$. Here $\mathsf{OPT}_{\mathsf{edge}}$ is the optimal value of EDGE-EXPANSION.

C.2 Laplacian Eigenmap

Recently, there has been renewed interest in the problem of developing low-dimensional representations when data arise from sampling a probability distribution on a manifold. This problem is referred to as manifold learning. There have been many approaches to manifold learning, such as *Isomap embedding*, *LLE embedding*, and *Laplacian eigenmap*. The locality-preserving character of the Laplacian eigenmap makes it relatively insensitive to outliers and noise, and hence the focus of this paper [5].

In the approach of Laplacian eigenmap, given n data samples $\{x_1, \cdots, x_n\} \in \mathbb{R}^d$, we construct a weighted graph $\mathcal{G}$ with n nodes and set of edges connecting neighboring points. The embedding map is now provided by computing the top k eigenvectors of the graph Laplacian. There are multiple ways in which we assign an edge $e = (u, v)$ and edge weight between nodes u and v. In this section, we consider an edge $e = (u, v)$ if $\|x_u - x_v\|_2 \leq \rho$ for some parameter ρ. If there is an edge, then that edge is given a weight, known as *heat kernel*, $e^{-\|x_u - x_v\|_2^2/t}$ for some parameter $t \in \mathbb{R}$. The goal here is to find an embedding mapping, i.e., an orthonormal projection matrix, $UU^\top$, which is close to the optimal projection matrix $U_k U_k^\top$ (here columns of U_k is formed by the top-k eigenvectors of $L_{\mathcal{G}}$).

Theorem 30. *There is an efficient learning algorithm that (α, β)-differentially privately output a rank-k subspace $\widehat{P}$ such that*

$$\left\|(\mathbb{I} - \widehat{P})L_{\mathcal{G}}\right\|_2 \leq \|L_{\mathcal{G}} - L_{\mathcal{G},k}\|_2 + O\left(\sqrt{\frac{n\log(1/\beta)}{\alpha^2\varepsilon^2}}\log\left(\frac{1}{\beta}\right)\right).$$

Here $L_{\mathcal{G},k} = \operatorname{argmin}_X \|L_{\mathcal{G}} - X\|_2$ is the optimal subspace of $L_{\mathcal{G}}$.

Proof. The algorithm is as follows:

Algorithm. Run the algorithm PRIVATE-SPARSIFY to output a Laplacian L_{priv}. Compute the top-k singular vectors of L_{priv} and output it.

We next argue the correctness of our algorithm to compute Laplacian eigenmap. We first fix some notations. Let $\widehat{P}$ be the top-k singular vectors of L_{priv} and $\widetilde{P}$ be the top-k singular vectors of $L_{\widehat{\mathcal{G}}}$. We use the notation $\sigma(A)$ to denote the spectral norm of the matrix A. Denote by $\widehat{Q} = \mathbb{I} - \widehat{P}, \widetilde{Q} = \mathbb{I} - \widetilde{P}$. Let $\widetilde{x}$ be the witness vector for $\widetilde{P}$ and $\widehat{x}$ be a witness vector for $\widehat{P}$. Further, let

$$\widetilde{x}_{L_{\mathsf{priv}}} = \operatorname*{argmax}_{x, \|x\|_2 = 1} x^\top \widetilde{Q}^\top L_{\mathsf{priv}} \widetilde{Q} x, \quad \widehat{x}_{L_{\widehat{\mathcal{G}}}} = \operatorname*{argmax}_{x, \|x\|_2 = 1} x^\top \widehat{Q}^\top L_{\widehat{\mathcal{G}}} \widehat{Q} x_2.$$

Using the definition of $\widehat{x}$, $\widetilde{x}_{L_{\mathsf{priv}}}$, and $\widetilde{Q}$, we have

$$\widehat{x}^\top \widehat{Q}^\top L_{\mathsf{priv}} \widehat{Q}\widehat{x} = \sigma\left(\widehat{Q} L_{\mathsf{priv}} \widehat{Q}\right) \leq \sigma\left(\widetilde{Q} L_{\mathsf{priv}} \widetilde{Q}\right) = \widetilde{x}_{L_{\mathsf{priv}}}^\top \widetilde{Q}^\top L_{\mathsf{priv}} \widetilde{Q} \widetilde{x}_{L_{\mathsf{priv}}}, \tag{5}$$

where the inequality is due to the fact that $\widehat{Q}$ mininimizes the quantity on the left.

[5] A related problem is that of privately learning a robust subspace which was recently studied by [4].

Then we have the following:

$$\begin{aligned}
\sigma\left(\widetilde{Q}L_{\widehat{\mathcal{G}}}\widetilde{Q}\right) &= \widetilde{x}^\top\widetilde{Q}^\top L_{\widehat{\mathcal{G}}}\widetilde{Q}\widetilde{x} \\
&\geq \widetilde{x}_{L_{\mathsf{priv}}}^\top\widetilde{Q}^\top L_{\widehat{\mathcal{G}}}\widetilde{Q}\widetilde{x}_{L_{\mathsf{priv}}} && \text{(Maximality)} \\
&\geq (1-\varepsilon)\widetilde{x}_{L_{\mathsf{priv}}}^\top\widetilde{Q}^\top L_{\mathsf{priv}}\widetilde{Q}\widetilde{x}_{L_{\mathsf{priv}}} && \text{(Spectral guarantee)} \\
&\geq (1-\varepsilon)\widehat{x}^\top\widehat{Q}^\top L_{\mathsf{priv}}\widehat{Q}\widehat{x} && \text{(equation (5))} \\
&= (1-\varepsilon)\sigma\left(\widehat{Q}L_{\mathsf{priv}}\widehat{Q}\right).
\end{aligned}$$

Similarly, we have the following:

$$\begin{aligned}
\sigma\left(\widehat{Q}L_{\mathsf{priv}}\widehat{Q}\right) &= \widehat{x}^\top\widehat{Q}^\top L_{\mathsf{priv}}\widehat{Q}\widehat{x} \\
&\geq \widehat{x}_{L_{\widehat{\mathcal{G}}}}^\top\widehat{Q}^\top L_{\mathsf{priv}}\widehat{Q}\widehat{x}_{L_{\widehat{\mathcal{G}}}} && \text{(Maximality)} \\
&\geq \frac{1}{(1+\varepsilon)}\widehat{x}_{L_{\widehat{\mathcal{G}}}}^\top\widehat{Q}^\top L_{\widehat{\mathcal{G}}}\widehat{Q}\widehat{x}_{L_{\widehat{\mathcal{G}}}} && \text{(Spectral guarantee)} \\
&= \frac{1}{(1+\varepsilon)}\sigma\left(\widehat{Q}L_{\widehat{\mathcal{G}}}\widehat{Q}\right). && \text{(By definition of } \widehat{x}_{L_{\widehat{\mathcal{G}}}})
\end{aligned}$$

In other words,

$$\sigma\left(\widehat{Q}L_{\widehat{\mathcal{G}}}\widehat{Q}\right) \leq \frac{1+\varepsilon}{1-\varepsilon}\sigma\left(\widetilde{Q}L_{\widehat{\mathcal{G}}}\widetilde{Q}\right)$$

Since $L_{\widehat{\mathcal{G}}} = \left(1-\frac{w}{n}\right)L_{\mathcal{G}} + \frac{w}{n}L_n$, subadditivity of spectral norms implies

$$\sigma\left(\widehat{Q}L_{\widehat{\mathcal{G}}}\widehat{Q}\right) \geq \sigma(\widehat{Q}L_{\mathcal{G}}\widehat{Q}) - \sigma\left(\frac{w}{n}L_n\right) = \sigma(\widehat{Q}L_{\mathcal{G}}\widehat{Q}) - w \quad \text{and}$$

$$\sigma\left(\widetilde{Q}L_{\widehat{\mathcal{G}}}\widetilde{Q}\right) = \sigma\left(L_{\widehat{\mathcal{G}}} - L_{\widehat{\mathcal{G}},k}\right) \leq \sigma\left(L_{\mathcal{G}} - L_{\widehat{\mathcal{G}},k}\right) + \sigma\left(\frac{w}{n}L_n\right) = \sigma\left(L_{\mathcal{G}} - L_{\widehat{\mathcal{G}},k}\right) + w.$$

Using Aclioptas and McSherry [1], we know that

$$\left\|L_{\mathcal{G}} - L_{\widehat{\mathcal{G}},k}\right\|_2 \leq \|L_{\mathcal{G}} - L_{\mathcal{G},k}\|_2 + O(\sqrt{n}).$$

This implies that

$$\left\|(\mathbb{I}-\widehat{P})L_{\mathcal{G}}\right\|_2 \leq \frac{1+\varepsilon}{1-\varepsilon}\|L_{\mathcal{G}} - L_{\mathcal{G},k}\|_2 + O(w)$$

The theorem follows after rescaling the value of ε. □

C.3 Applications of Cut Sparsification: Achieving Differential Privacy

Cut sparsifiers were introduced by [8] to compute approximations for maximum cut, sparsest cut, and maximum flow, etc. Our results can be applied to all these settings. The main benefit of using cut-sparsifier is that we get $(\alpha, 0)$-differential privacy with a comparatively more efficient algorithm.

Cut Queries. The first application of cut sparsifier is that we can efficiently compute all possible cut queries with error that scales linearly with n.

Theorem 31. *Given an n vertices m edges graph $\mathcal{G}$, there is an efficient α-differentially private algorithm that preprocess the graph in polynomial time to output another graph $\widetilde{\mathcal{G}}$. Further, given a cut query in the form of set of vertices S, one can output the cut with additive error $\tau \leq \widetilde{O}\left(\sqrt{\frac{sn(n-s)}{\alpha^2}}\right)$ with probability at least $99/100$. Further, computing a cut query takes $O(s)$ time.*

Proof. The algorithm is as follows:

Algorithm. Given a graph $\mathcal{G}$, compute a sparse graph $\widetilde{\mathcal{G}}$ using Theorem 7. Then given a vector $\mathbf{1}_S$, compute $\alpha^{-1}\mathbf{1}_S^\top L_{\widetilde{G}}\mathbf{1}_S$.

We next argue the correctness of the algorithm. Using Theorem 7, we have

$$\begin{aligned}\frac{1}{\alpha}\mathbf{1}_S^\top L_{\widetilde{G}}\mathbf{1}_S &\leq (1+\varepsilon)\Phi_S(\mathcal{G}) + \frac{\sqrt{n\mathbf{1}_S^\top L_n \mathbf{1}_S}}{\alpha} \\ &= (1+\varepsilon)\Phi_S(\mathcal{G}) + O\left(\sqrt{\frac{sn(n-s)}{\alpha^2}}\right)\end{aligned}$$

with probability at least $99/100$. Similarly, for the lower bound, we have

$$\begin{aligned}\frac{1}{\alpha}\mathbf{1}_S^\top L_{\widetilde{G}}\mathbf{1}_S &\geq (1-\varepsilon)\Phi_S(\mathcal{G}) - \frac{\sqrt{n\mathbf{1}_S^\top L_n \mathbf{1}_S}}{\alpha} \\ &= (1-\varepsilon)\Phi_S(\mathcal{G}) - O\left(\sqrt{\frac{sn(n-s)}{\alpha^2}}\right)\end{aligned}$$

This completes the proof of Theorem 31. □

MAX-CUT. Another problem one can solve using our cut sparsifier is that of MAX-CUT. Here, given a graph $\mathcal{G} = (V, E)$ on vertex set V, the goal is to output a set of vertices $S \subseteq V$ that maximizes the value of $\Phi_S(\mathcal{G})$. It is well known that MAX-CUT is NP-hard. However, Goemans and Williamson gave an elegant polynomial time algorithm for computing $\zeta_{GW} - \eta$ approximation to MAX-CUT—it is known that even approximation within a factor of $\zeta_{GW} + \rho$, for all $\rho > 0$ is NP-hard [31] under the unique games conjecture [30], where ζ_{GW} is as defined in equation (4).

Theorem 32. *For an n-vertex graph $\mathcal{G} := (V, E)$, there is a polynomial-time algorithm that is $(\alpha, 0)$-differentially private with respect to the edge level privacy and produces a set of nodes $S \subseteq V$ satisfying*

$$\Phi_S(\mathcal{G}) \geq (\zeta_{GW} - \eta)\left(\frac{1-\varepsilon}{1+\varepsilon}\right)\mathsf{OPT}_{\mathsf{max}} - O\left(\frac{sn}{\alpha}\right),$$

with probability at least $9/10$. Here ζ_{GW} is as defined in equation (4), $\mathsf{OPT}_{\mathsf{max}}$ is the optimal value of MAX-CUT *and $\Phi_S(\mathcal{G})$ is the size of cut for vertex set in $S \subseteq V$.*

Proof. The idea is to first compute a private sparse graph using Algorithm 2, then run the SDP based algorithm of [21] on the private sparse graph, and output the set of vertices S outputted by that algorithm. Let $\widetilde{S}_{\mathsf{OPT}}$ be the solution of MAX-CUT on graph $\widetilde{\mathcal{G}}$ and S_{OPT} be the solution of MAX-CUT on the graph $\mathcal{G}$. In other words,

$$\widetilde{S}_{\mathsf{OPT}} = \operatorname*{argmax}_{S\subseteq V} \mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}}\mathbf{1}_S \quad \text{and} \quad S_{\mathsf{OPT}} = \operatorname*{argmax}_{S\subseteq V} \mathbf{1}_S^\top L_{\mathcal{G}}\mathbf{1}_S.$$

From Theorem 24, we know that

$$\Phi_S(\widetilde{\mathcal{G}}) \geq (\zeta_{GW} - \eta)\max_{S\subseteq V} \mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}}\mathbf{1}_S$$

with probability at least $99/100$. Further, since $\widetilde{\mathcal{G}}$ is a cut-sparsification, Theorem 7 implies that

$$(1-\varepsilon)\mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}}\mathbf{1}_S - O\left(\sqrt{n\mathbf{1}_S^\top L_n \mathbf{1}_S}\right) \leq \alpha\mathbf{1}_S^\top L_{\mathcal{G}}\mathbf{1}_S \leq (1+\varepsilon)\mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}}\mathbf{1}_S + O\left(\sqrt{n\mathbf{1}_S^\top L_n \mathbf{1}_S}\right)$$

holds simultaneously for $\forall S \subseteq [n]$ with probability $99/100$. Applying Theorem 7 twice, once with $\widetilde{S}_{\mathsf{OPT}}$ and S_{OPT}, we have the desired claim. □

SPARSEST-CUT. Another problem that is considered in graph theory is the problem of sparsest cut, SPARSE-CUT. Here, given a graph $\mathcal{G} = (V, E)$ on vertex set V, the goal is to output a set of vertices S that minimizes the value $\frac{\Phi_S(\mathcal{G})}{|S|(n-|S|)}$.

Combining Theorem 26 result with Theorem 7, we immediately get the following:

Theorem 33. *For an n-vertex graph $\mathcal{G} := (V, E)$, there is a polynomial-time algorithm that is $(\alpha, 0)$-differentially private with respect to the edge level privacy and produces a set of nodes S satisfying*

$$\Phi_S(\mathcal{G}) \leq O(\sqrt{\log n}) \left(\frac{1+\varepsilon}{1-\varepsilon}\right) \mathsf{OPT}_{\mathsf{sparsest}} + O\left(\frac{\log^2 n}{\varepsilon\sqrt{|S|}}\right),$$

with probability at least $9/10$. Here $\mathsf{OPT}_{\mathsf{sparsest}}$ is the optimal value of SPARSE-CUT *and $\Phi_S(\mathcal{G})$ is the size of cut for vertex set in $S \subseteq V$.*

Proof. The idea is to first compute a private sparse graph using Algorithm 2, then run the SDP based algorithm of [5] on the private sparse graph, and output the set of vertices S outputted by that algorithm. Let $\widetilde{S}_{\mathsf{OPT}}$ be the solution of SPARSEST-CUT on graph $\widetilde{\mathcal{G}}$ and S_{OPT} be the solution of SPARSEST-CUT on the graph $\mathcal{G}$. In other words,

$$\widetilde{S}_{\mathsf{OPT}} = \operatorname*{argmin}_{S \subseteq V} \left(\frac{\mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}} \mathbf{1}_S}{|S|(n-|S|)}\right) \quad \text{and} \quad S_{\mathsf{OPT}} = \operatorname*{argmin}_{S \subseteq V} \left(\frac{\mathbf{1}_S^\top L_{\mathcal{G}} \mathbf{1}_S}{|S|(n-|S|)}\right).$$

From Theorem 26, we know that

$$\Phi_S(\widetilde{\mathcal{G}}) \leq O(\sqrt{\log n}) \min_{S \subseteq V} \left(\frac{\mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}} \mathbf{1}_S}{|S|(n-|S|)}\right)$$

with probability at least $99/100$. Further, since $\widetilde{\mathcal{G}}$ is a cut-sparsification, Theorem 7 implies that

$$(1-\varepsilon)\mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}} \mathbf{1}_S - O\left(\sqrt{n\mathbf{1}_S^\top L_n \mathbf{1}_S}\right) \leq \alpha \mathbf{1}_S^\top L_{\mathcal{G}} \mathbf{1}_S \leq (1+\varepsilon)\mathbf{1}_S^\top L_{\widetilde{\mathcal{G}}} \mathbf{1}_S + O\left(\sqrt{n\mathbf{1}_S^\top L_n \mathbf{1}_S}\right)$$

holds simultaneously for $\forall S \subseteq V$ with probability at least $99/100$. Applying Theorem 7 twice, once with $\widetilde{S}_{\mathsf{OPT}}$ and S_{OPT}, we have the desired claim. □

EDGE-EXPANSION. Using the fact that, up to a factor 2, computing the sparsest cut is the same as computing the *edge expansion* of the graph, we have the following corollary.

Theorem 34. *For an n-vertex graph $\mathcal{G} = (V, E)$, there is a polynomial-time algorithm that is (α, β)-differentially private with respect to the edge level privacy and produces a set of nodes $S \subseteq V$ satisfying*

$$\mathsf{OUT} \leq O(\sqrt{\log n}) \left(\frac{1+\varepsilon}{1-\varepsilon}\right) \mathsf{OPT}_{\mathsf{edge}} + O\left(\frac{\log^2 n}{\varepsilon\sqrt{|S|}}\right),$$

with probability at least $9/10$. Here $\mathsf{OPT}_{\mathsf{edge}}$ is the optimal value of EDGE-EXPANSION.

D Flexibility of Our Approach

In this section, we illustrate the flexibility of our approach. For the ease of presentation, we first assume that the graph is unweighted. Later, in Section D.1, we show how to remove this assumption. We prove Theorem 35 through a series of lemmata. We first show in Lemma 36 that we can privately compute an overestimate of the leverage scores and still the vector of leverage scores has a small ℓ_0 norm. This implies that the number of edges in our sparsified graph is of order $\widetilde{O}(n/\varepsilon^2)$.

Theorem 35. *Let $\mathcal{G}$ be a graph on n vertices. Given an approximation parameter $0 < \varepsilon < 1$, confidence parameter δ, and privacy parameters α, β, let $w = O\left(\sqrt{\frac{n\log(1/\beta)}{\alpha^2\varepsilon^2}}\log(1/\beta)\right)$. Then* PRIVATE-SPARSIFY-ADD-MULT*, described in Algorithm 3, is (α, β)-differentially private algorithm,*

Algorithm 3 PRIVATE-SPARSIFY-ADD-MULT $(\mathcal{G}, \varepsilon, (\alpha, \beta))$

Input: An n vertex graph $\mathcal{G} = (V, E)$, privacy parameters (α, β), approximation parameter ε.
Output: A Laplacian $L_{\widetilde{\mathcal{G}}}$.

1: **Initialization.** Sample a random Gaussian matrix $M \in \mathbb{R}^{n/\varepsilon^2 \times m}$ such that $M_{ij} \sim \mathcal{N}(0, \varepsilon^2/n)$.
2: **Privatize.** Compute a graph $\widehat{\mathcal{G}}$ with Laplacian $L_{\widehat{\mathcal{G}}} = \left(1 - \frac{w}{n}\right) L_{\mathcal{G}} + \frac{w}{n} L_n$, where $w = O\left(\sqrt{\frac{n \log(1/\beta)}{\alpha^2 \varepsilon^2}} \log\left(\frac{1}{\beta}\right)\right)$. Let $E_{\widehat{\mathcal{G}}}$ be the corresponding edge-adjacency matrix and E_{K_n} be the edge-adjacency matrix of weighted complete graph, $\sqrt{w/n} K_n$. Compute $H = M E_{\widehat{\mathcal{G}}}$.
3: **Compute effective resistance.** $\widetilde{\tau}_i = e_i (H^\top H)^\dagger e_i^\top$, where e_i is i-th the row of E_{K_n}.
4: **Construct** diagonal matrix $D \in \mathbb{R}^{m \times m}$ whose diagonal entries are $D_{ii} := p_i^{-1}$ with probability p_i, where $p_i = \min\left\{c \widetilde{\tau}_i \varepsilon^{-2} \log(n/\delta), 1\right\}$, and 0 otherwise.
5: **Output.** $L_{\widetilde{\mathcal{G}}} := E_{K_n}^\top D E_{K_n}$.

and outputs a Laplacian $L_{\widetilde{\mathcal{G}}}$ of $\widetilde{O}(n/\varepsilon^2)$ edges graph, such that, with probability at least $99/100$, we have

$$\widetilde{O}\left(\frac{1-\varepsilon}{\sqrt{n}}\right) L_n - \varepsilon\left(1 - \frac{w}{n}\right) L_{\mathcal{G}} \preceq L_{\widetilde{\mathcal{G}}} \preceq \varepsilon\left(1 - \frac{w}{n}\right) L_{\mathcal{G}} + \widetilde{O}\left(\frac{1+\varepsilon}{\sqrt{n}}\right) L_n,$$

where $L_{\mathcal{G}}$ is the Laplacian of the input graph $\mathcal{G}$ and L_n is the Laplacian of complete graph K_n.

We first give a proof sketch. The privacy proof follows from [9, 54]. To complete the proof, we prove the following: $\tau_i \leq \widetilde{\tau}_i$, where τ_i are the true effective resistance, $\sum_i \widetilde{\tau}_i = O(n)$, and the spectral guarantee. Since M is a random Gaussian matrix, we can show that $(1-\varepsilon) L_{\widehat{\mathcal{G}}} \preceq E_{\widehat{\mathcal{G}}}^\top M^\top M E_{\widehat{\mathcal{G}}} \preceq (1+\varepsilon) L_{\widehat{\mathcal{G}}}$. Intuitively, our sampling is equivalent to sampling by the standard effective resistance of $L_{\widehat{\mathcal{G}}}$. We use the matrix Bernstein to show that sampling by these effective resistance will yield $L_{\widetilde{\mathcal{G}}}$ satisfying $(1-\varepsilon) L_{\widehat{\mathcal{G}}} \preceq L_{\widetilde{\mathcal{G}}} \preceq (1+\varepsilon) L_{\widehat{\mathcal{G}}}$. However, we actually sample the edges of complete graph K_n. Subtracting off the effect of $\left(1 - \frac{w}{n}\right) L_{\mathcal{G}}$ yields the mixed additive-multiplicative bound. We now give a detailed proof.

Lemma 36. *Let M be an $O\left(\frac{n \log(1/\delta)}{\varepsilon^2}\right) \times \binom{n}{2}$ random Gaussian matrix with entries sampled iid from $\mathcal{N}(0, 2\varepsilon/n \log(1/\delta))$. Let $E_{\widehat{\mathcal{G}}}$ be the edge-adjacency matrix of the graph $\widehat{\mathcal{G}}$ formed by overlaying a weighted complete graph $\frac{w}{n} L_n$ on top of the input graph, i.e., $L_{\widehat{\mathcal{G}}} = \left(1 - \frac{w}{n}\right) L_{\mathcal{G}} + \frac{w}{n} K_n$. Let e_i be the i-th row for edge-adjacency matrix of complete graph K_n. Define a diagonal matrix $\widetilde{\tau} \in \mathbb{R}^{\binom{n}{2} \times \binom{n}{2}}$*

$$\widetilde{\tau}_i = e_i \left(E_{\widehat{\mathcal{G}}}^\top M^\top M E_{\widehat{\mathcal{G}}}\right)^\dagger e_i^\top.$$

Then $\|\widetilde{\tau}\|_0 \leq \widetilde{O}(n(1+\varepsilon)/\varepsilon^2)$.

Proof. Let $E_{\widehat{\mathcal{G}}}^\top E_{\widehat{\mathcal{G}}} = U \Lambda^2 U^\top$. Since $L_{\widehat{\mathcal{G}}}$ has rank $n-1$ and M is a random Gaussian matrix of dimensions $O\left(\frac{n \log(1/\delta)}{\varepsilon^2}\right) \times \binom{n}{2}$, [48] gives us that with probability $1 - \delta$,

$$(1-\varepsilon) L_{\widehat{\mathcal{G}}} \preceq E_{\widehat{\mathcal{G}}}^\top M^\top M E_{\widehat{\mathcal{G}}} \preceq (1+\varepsilon) L_{\widehat{\mathcal{G}}}$$

In other words, $(1-\varepsilon) L_{\widehat{\mathcal{G}}}^\dagger \preceq (E_{\widehat{\mathcal{G}}}^\top M^\top M E_{\widehat{\mathcal{G}}})^\dagger \preceq (1+\varepsilon) L_{\widehat{\mathcal{G}}}^\dagger$. This in particular implies that $\widetilde{\tau}_i \leq (1+\varepsilon)\tau_i$, where τ_i is the true effective resistance for i-th edge.

Set $p_i = \min\left\{\widetilde{\tau}_{i,i} \frac{c \log(n/\delta)}{\varepsilon^2}, 1\right\}$ for $1 \leq i \leq n$. Let D is a diagonal matrix whose diagonal entries are p_i^{-1} with probability p_i. The expected number of edges in the resulting graph is $\sum_i \widetilde{\tau}_i$. So Chernoff

bound will give that with probability $1-\delta$, the number of edges in the graph $\widetilde{\mathcal{G}}$ is

$$\begin{aligned}
\sum_i p_i &\le \frac{c\log(n/\delta)}{\varepsilon^2}\sum_i \widetilde{\tau}_i \\
&\le \frac{c(1+\varepsilon)\log(n/\delta)}{\varepsilon^2}\mathsf{Tr}\left(E_{K_n}L_{\widehat{\mathcal{G}}}^{\dagger}E_n^{\top}\right) \\
&= \frac{c(1+\varepsilon)\log(n/\delta)}{\varepsilon^2}\mathsf{Tr}\left(E_n^{\top}E_{K_n}L_{\widehat{\mathcal{G}}}^{\dagger}\right) \\
&\le \frac{c(1+\varepsilon)\log(n/\delta)}{\varepsilon^2}\left(\mathsf{Tr}\left(E_n^{\top}E_{K_n}L_{\widehat{\mathcal{G}}}^{\dagger}\right)+\mathsf{Tr}\left(E_{\mathcal{G}}^{\top}E_{\mathcal{G}}L_G^{\dagger}\right)\right) \\
&= \frac{c(1+\varepsilon)\log(n/\delta)}{\varepsilon^2}\mathsf{Tr}\left(E_{\widehat{\mathcal{G}}}L_{\widehat{\mathcal{G}}}^{\dagger}E_{\widehat{\mathcal{G}}}^{\top}\right) \\
&\le \frac{cn(1+\varepsilon)\log(n/\delta)}{\varepsilon^2}.
\end{aligned}$$

This completes the proof of Lemma 36. □

Equipped with Lemma 36, we can now prove the spectral sparsification guarantee of our output.

Lemma 37. *With probability at least $1-\delta$, we have*

$$\widetilde{O}\left(\frac{1-\varepsilon}{\sqrt{n}}\right)L_n-\varepsilon\left(1-\frac{w}{n}\right)L_{\mathcal{G}}\preceq\varepsilon\left(1-\frac{w}{n}\right)L_{\mathcal{G}}+L_{\widetilde{\mathcal{G}}}\preceq\widetilde{O}\left(\frac{1+\varepsilon}{\sqrt{n}}\right)L_n$$

Proof. We consider the graph $\mathcal{G}$ overlaid with a weighted complete graph with weights w/n, i.e.,

$$L_{\widehat{\mathcal{G}}}=\frac{w}{n}L_n+\left(1-\frac{w}{n}\right)L_{\mathcal{G}}$$

In other words, we have $E_{K_n}^{\top}DE_{K_n}$. More precisely, $\tau_i=\frac{w}{n}e_iL_{\widehat{\mathcal{G}}}^{\dagger}e_i^{\top}=\frac{w}{n}e_iU_{\widehat{G}}\Sigma_{\widehat{G}}^{-2}U_{\widehat{G}}^{\top}e_i^{\top}$. We borrow the idea from [14]. Let define $Q_i=\sqrt{\frac{w}{n}}e_iU_{\widehat{G}}\Sigma_{\widehat{G}}^{-1}$. Next define the following matrix valued random variable:

$$X_i:=\begin{cases}\left(\frac{1}{p_i}-1\right)Q_i^{\top}Q_i & \text{with probability } p_i\\ -Q_i^{\top}Q_i & \text{with probability } 1-p_i\end{cases}.$$

Now $Y=\sum_i X_i$. Then we have

$$\mathsf{E}[Y]=\sum_{i=1}^{n}\left(p_i\left(\frac{1}{p_i}-1\right)Q_i^{\top}Q_i-(1-p_i)Q_i^{\top}Q_i\right)=0.$$

By the definition of p_i, $\|X_i\|_2=1$. We have to bound $\sigma^2=\left\|\mathsf{E}\left[Y^2\right]\right\|_2$. For this note that

$$\begin{aligned}
\mathsf{E}[Y^2] &= \sum_i\left[p_i\left(\frac{1}{p_i}-1\right)^2+(1-p_i)\right]\frac{w^2}{n^2}(e_iU_{\widehat{G}}\Sigma_{\widehat{G}}^{-1})^{\top}e_iU_{\widehat{G}}\Sigma_{\widehat{G}}^{-2}U_{\widehat{G}}^{\top}e_i^{\top}e_iU_{\widehat{G}}\Sigma_{\widehat{G}}^{-1} \\
&\preceq \sum_i O\left(\frac{\tau_i}{p_i}\right)\frac{w}{n}\Sigma_{\widehat{G}}^{-1}U_{\widehat{G}}^{\top}e_i^{\top}e_iU_{\widehat{G}}\Sigma_{\widehat{G}}^{-1} \\
&\preceq O\left(\frac{\varepsilon^2}{\log(n/\delta)}\right)\Sigma_{\widehat{G}}^{-1}U_{\widehat{G}}^{\top}\left(\frac{w}{n}L_n\right)U_{\widehat{G}}\Sigma_{\widehat{G}}^{-1} \\
&\preceq O\left(\frac{\varepsilon^2}{\log(n/\delta)}\right)\mathbb{I}.
\end{aligned}$$

This is because $\frac{w}{n}L_n\preceq U_{\widehat{G}}\Sigma_{\widehat{G}}^2U_{\widehat{G}}^{\top}$. In other words, $\sigma^2\le O\left(\frac{\varepsilon^2}{\log(n/\delta)}\right)$. Further, we can compute the intrinsic dimension as

$$\frac{\mathsf{Tr}(\mathbb{I})}{\|\mathbb{I}\|_2}=n-1.$$

Using Matrix Bernstein inequality for intrinsic dimension [53] now gives us the following:

$$\mathsf{Pr}\left[\|Y\|_2 \geq \varepsilon\right] \leq 4ne^{-c\ln(n/\delta)/2} \leq \delta/2$$

for large enough c. Noting that $L_{\widetilde{\mathcal{G}}} = U_{\widehat{G}}\Sigma_{\widehat{G}}Y\Sigma_{\widehat{G}}U_{\widehat{\mathcal{G}}}^{\top} + \frac{w}{n}L_n$ gives us that

$$\widetilde{O}\left(\frac{1-\varepsilon}{\sqrt{n}}\right)L_n - \varepsilon\left(1-\frac{w}{n}\right)L_{\mathcal{G}} \preceq L_{\widetilde{\mathcal{G}}} \preceq \varepsilon\left(1-\frac{w}{n}\right)L_{\mathcal{G}} + \widetilde{O}\left(\frac{1+\varepsilon}{\sqrt{n}}\right)L_n$$

This completes the proof of Lemma 37. □

Lemma 38. *$L_{\widetilde{\mathcal{G}}}$ outputted by* PRIVATE-SPARSIFY-ADD-MULT *is (α,β)-differentially private with respect to the edge-level privacy.*

Proof. We first prove the privacy guarantee. The only time the graph is used in the algorithm PRIVATE-SPARSIFY-ADD-MULT is when we compute $H := ME_{\widehat{\mathcal{G}}}$. This is differentially private due to [9, 54] by our choice of w. Using the post-processing property (Lemma 15) of differential privacy, the result follows. □

D.1 Extension to Weighted Graph

We can use a standard technique to extend our result for Theorem 35 to weighted graphs in which an edge's weight is specified. We assume that the weights on the graph are integers in the range $[1, \operatorname{poly} n]$. We consider different levels $(1+\varepsilon)^i$ for $i \in [c\log n]$ for some constant c. Then we consider input graphs being partitioned in form

$$L_{\mathcal{G}} = \sum_{i=1}^{c\log n} L_{\mathcal{G},i},$$

where $L_{\mathcal{G},i}$ has edges with weights $\left\{0, (1+\varepsilon)^i\right\}$. In other words, we use the $(1+\varepsilon)$-ary representation of weights on the edges and partition the graph accordingly. Again since there are at most $\operatorname{poly}\log n$ levels, the number of edges in $\widetilde{\mathcal{G}}$ is $\widetilde{O}(n/\varepsilon^2)$. Since $\widetilde{\mathcal{G}}$ is (α,β)-differentially private, we can run another instance of [35] to get $O(n/\varepsilon^2)$ edge graph $\widehat{\mathcal{G}}$. Using Lemma 22, we therefore have

Theorem 39. *Given the privacy parameter (α,β), the accuracy parameter ε and confidence parameter δ, let $w = \widetilde{O}\left(\sqrt{\frac{n\log(1/\beta)}{\alpha^2\varepsilon^2}}\log\left(\frac{1}{\beta}\right)\right)$. Given a weighted graph $\mathcal{G}$,* PRIVATE-SPARSIFY-ADD-MULT *outputs a Laplacian of a graph $L_{\widetilde{\mathcal{G}}}$ with the following guarantees:*

1. *$L_{\widetilde{\mathcal{G}}}$ is (α,β)-differentially private with respect to the edge-level privacy.*
2. *With probability at least $1-\delta$, we have*
$$O\left(\frac{w(1-\varepsilon)}{n}\log n\right)L_n - \varepsilon L_{\mathcal{G}} \preceq L_{\widetilde{\mathcal{G}}} \preceq O\left(\frac{w(1+\varepsilon)}{n}\log n\right)L_n + \varepsilon L_{\mathcal{G}}.$$
3. *The number of edges in $L_{\widetilde{\mathcal{G}}}$ is $O(n/\varepsilon^2)$.*

E Why Traditional Approaches Do Not Work?

We argued briefly in the introduction that traditional approaches for spectral sparsification do not work. We also argued that traditional privacy mechanisms also fails to either give good spectral sparsification guarantees. In this section, we give the technical reasons why all these approaches do not work for privacy.

E.1 Using Known Privacy Techniques

Construction. Compute the leverage score and then add noise scaled to its sensitivity. This would incur an error proportional to $O(\ell_{\mathsf{Lip}}n^2)$ if effective resistance has Lipschitz constant ℓ_{Lip} as the vector of effective resistance has dimension $O(n^2)$. Unfortunately, we show that effective resistance is not Lipschitz continuous. In other words,

Lemma 40. *Effective resistance of edges is not a Lipschitz smooth function.*

Proof. Let e_i be an edge in the graph $\mathcal{G}$ with weight 1. Let $\mathcal{G}'$ be a neighboring graph with all edges same as in $\mathcal{G}$ except for the edge i that has weight 2. We can consider this action as a diagonal matrix C acting on $E_{\mathcal{G}}$ where $C_{ii} = 2$ for $i \in [\binom{n}{2}]$ and $C_{jj} = 1$ for all $j \neq i$. In this notation $E_{\mathcal{G}'} = CE_{\mathcal{G}}$. Let denote by τ_i the effective resistance for an edge $i = (u, v)$ between nodes u and v, i.e., $\tau_i := (E_{\mathcal{G}})_i (L_{\mathcal{G}})^\dagger (E_{\mathcal{G}})_i^\top$. Using [38, Corollary 3], we have

$$\begin{aligned}
\tau_i' := \tau_i'(E_{\mathcal{G}'}) &= 2(E_{\mathcal{G}})_i \left(L_{\mathcal{G}} + 2(E_{\mathcal{G}})_i (E_{\mathcal{G}})_i^\top\right)^\dagger (E_{\mathcal{G}})_i^\top \\
&= 2(E_{\mathcal{G}})_i \left(L_{\mathcal{G}}^\dagger - 2\frac{L_{\mathcal{G}}^\dagger (E_{\mathcal{G}})_i^\top (E_{\mathcal{G}})_i L_{\mathcal{G}}^\dagger}{1 + 2(E_{\mathcal{G}})_i L_{\mathcal{G}}^\dagger (E_{\mathcal{G}})_i^\top}\right) (E_{\mathcal{G}})_i^\top \\
&= 2\left(\tau_i - 2 \times \frac{\tau_i^2}{1 + 2\tau_i}\right) \\
&= \frac{2\tau_i}{1 + 2\tau_i} \geq \frac{2}{3}\tau_i
\end{aligned}$$

On the other hand, using [38, Corollary 3] again, we have for $j \neq i$,

$$\begin{aligned}
\tau_j' := \tau_j'(E_{\mathcal{G}'}) &= (E_{\mathcal{G}})_j \left(L_{\mathcal{G}} + 2(E_{\mathcal{G}})_j (E_{\mathcal{G}})_j^\top\right)^\dagger (E_{\mathcal{G}})_j^\top \\
&= (E_{\mathcal{G}})_j \left(L_{\mathcal{G}}^\dagger - 2 \times \frac{L_{\mathcal{G}}^\dagger (E_{\mathcal{G}})_j^\top (E_{\mathcal{G}})_j L_{\mathcal{G}}^\dagger}{1 + 2(E_{\mathcal{G}})_j L_{\mathcal{G}}^\dagger (E_{\mathcal{G}})_j^\top}\right) (E_{\mathcal{G}})_j^\top \\
&= \tau_j - \frac{2\tau_j^2}{1 + 2\tau_j} \\
&= \tau_j
\end{aligned}$$

In other words, effective resistances are not Lipschitz smooth. □

Objective Perturbation and Resampling. Another option to preserve privacy is performing output perturbation. If we compute a sparse graph and then perform output perturbation, we need to perturb every possible edges. We can still sparsify this graph since differential privacy is preserved under post-processing; however, this procedure leads to an error term that scales proportional to $O(n^2)$. So any hope of using output perturbation seems to hit a road block.

Recursive Sampling. Another approach that one can try is to use recursive sampling, i.e., iterate the following few number of times: form a coarse sparsifier, add noise, and then sparsify again. This approach has seen success in the context of k-rank approximation where the general technique used is Krylov subspace iteration: compute a QR decomposition, then add noise, and then compute the QR decomposition again. Unfortunately, while in the case of low-rank approximation, the noise only scales proportional to k, in this case, the added noise would aggregate leading to an error term that scales proportional to n and the number of iterations.

Exponential Mechanism. Note that the set of all sparsifiers is bounded above by $\exp(O(n \log(n)))$. 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 $\eta < 1/2$ and think of a set of inputs $\mathcal{G}$ where each graph has $n/2$ vertices with degree $n\eta$ and $n/2$ vertices with degree $n^{2\eta}$. Preserving all cuts of size 1 up to $(1 \pm \varepsilon)$ requires our output to have vertices of degree at least $(1 - \varepsilon)n^{2\eta}$ and vertices of degree less than $(1 + \varepsilon)n\eta$. 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/α.