# Differentially Private Analysis on Graph Streams

**Raman Arora**
Johns Hopkins University
Baltimore MD 21201
arora@cs.jhu.edu

**Jalaj Upadhyay**
Apple[1]
Cupertino CA 95050
jalaj@apple.com

**Sarvagya Upadhyay**
Fujitsu Laboratories of America
Sunnyvale CA 94085
supadhyay@fujitsu.com

## Abstract

In this paper, we focus on answering queries, in a differentially private manner, on graph streams. We adopt the sliding window model of privacy, where we wish to perform analysis on the last $W$ updates and ensure that privacy is preserved for the entire stream. We show that in this model, the price of ensuring differential privacy is minimal. Furthermore, since differential privacy is preserved under post-processing, our results can be used as a subroutine in many tasks, most notably solving cut functions and spectral clustering.

## 1 Introduction

Graph-based analysis has attracted considerable attention in many areas, including computer science, data analysis, physics, biology, and social sciences. A recurring feature in graph analysis is partitioning the vertices into distinct groups. For instance, partitioning a graph into distinct groups exhibiting a strong community structure (Girvan and Newman, 2002; Newman, 2004) has found applications in marketing analysis, supply chain modeling, and telecommunication networks, to name a few. The seminal work of Newman (2004) and subsequently (Frank et al., 2019) showed that learning community structure and several related statistical and learning tasks on graphs such as Lipschitz learning on graphs, can be accomplished using spectral analysis of graphs. Spectral analysis is also useful in randomized linear algebra (Cohen et al., 2015), linear programming (Lee and Sidford, 2014), and spectral clustering (Peng and Spielman, 2014).

With such a powerful analytical tool at our disposal, it is imperative to come up with mechanisms that allow us to do analysis privately. This is especially important in the wake of recent studies that showed that even innocuous statistical analysis on graphs, if not done properly, can lead to privacy leaks (Karwa et al., 2011; Liu et al., 2008). Motivated by this, algorithms have been proposed for estimating the spectrum of graphs while preserving *differential privacy* (Arora and Upadhyay, 2019; Blocki et al., 2012).

Differential privacy as a privacy tool has gained significant traction in the last decade. From a theoretical viewpoint, it offers an intuitive, meaningful, and formally rigorous notion of privacy that quantifies the loss of privacy in an easy to understand measurable fashion. The notion is adaptable to a large class of algorithms where simpler differentially private subroutines can be composed sequentially or in parallel to yield complex algorithms without compromising too much on privacy. This is a highly desirable feature, and unsurprisingly, a plethora of techniques have been developed that focus on preserving differential privacy. They have also led to several large scale real-world deployments for various tasks (Ding et al., 2017; Erlingsson et al., 2014; Haney et al., 2017; Kenthapadi et al., 2017; Thakurta et al., 2017a,b).

However, there is a gap between theoretical research and practical implementation of differentially private algorithms for estimating graph spectrum and analysis. All the results in private graph analysis assume that the graphs are static (Blocki et al., 2012; Eliáš et al., 2020; Gupta et al., 2012; Upadhyay, 2013), while many current implementations (of private and non-private algorithms) assume that the underlying data is dynamically updated. Moreover, recent updates are given more significance for generating any prediction model or analysis (Lee and Maggioni, 2011; Tylenda et al., 2009). For instance, the differential privacy overview of Apple states, "Apple retains the collected data for a maximum of three months." Given the ubiquitous nature of graphs in statistical analysis

[1]Work done while at Johns Hopkins University.

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. 

and learning theory, the aim of this paper is to give a theoretical foundation to the heuristics and empirical observations.

To do this formally, we consider the *sliding window model of privacy* introduced by Bolot et al. (2013). This model is parameterized by window size $W$ and the objective is to perform the required task using data received in the last $W$ updates while using $o(W)$ space.

As a motivating example of this set up, consider the sceanrio where we are contact tracing a contagious disease with an incubation period of $W$. This can be modeled as a graph $\mathcal{G} = (V, E)$, where vertex set $V$ corresponds to individuals and an edge $(u, v) \in E$ is present between two nodes if individuals $u$ and $v$ have come in contact. At a given time, suppose we learn that a certain set of individuals (say $S \subseteq V$) are tested positive for the disease. A natural privacy requirement would be to hide whether two individuals came in contact (or presence/absence of an edge). A natural query would be identify the total number of individuals that have come in contact with anyone in the set $S$ during the last $W$ time period. Another natural query could be to understand the spread of the disease due to infected individual set $S$ or the set of individuals that can lead to maximum number of possibly infected individuals. All these problems can be succintly answered by studying the spectrum of the underlying graph.

Our main contribution is the first efficient differentially private algorithm in the sliding window model that outputs a graph approximating the spectrum of the positive weighted input graph using $o(W)$ space.

**Notation.** For a graph $\mathcal{G} := (V, E)$ with vertex set $V$ and edge set $E$, let $w(u, v)$ denote the weight on the edge between vertices $u$ and $v$. If $(u, v) \notin E$, then $w(u, v) = 0$. Given a graph, $\mathcal{G}$, its *Laplacian*, denoted $L_{\mathcal{G}}$, is the matrix with the following entries

$$L_{\mathcal{G}}[u, v] = \begin{cases} -w(u, v) & \text{if } u \neq v \\ \sum_{v \in V \setminus \{u\}} w(u, v) & \text{if } u = v \end{cases}.$$

We use $K_n$ to denote an $n$-vertex complete graph with all edge weights equal to 1 and $\mathcal{G} \cup \mathcal{H}$ to denote the $n$ vertex graph formed by the union of edges in $\mathcal{G}$ and $\mathcal{H}$. For symmetric positive semidefinite matrices, $A, B \in \mathbb{R}^{n \times n}$, we use $A \succeq B$ to denote the p.s.d. ordering, i.e., $A - B \succeq 0$. The notation $\|A\|_2$ denotes the spectral norm of a matrix $A$. The asymptotic notation $\widetilde{O}(f(n))$ is equal to $O(f(n) \operatorname{poly} \log(n))$, where $n$ is the number of vertices in the graph.

## 2 Problem setup and main results

The dataset considered in this paper are graphs with $n$ vertices and positive edge weights formed by a stream of (edge, weight) tuples,

$$\mathsf{S} := \left(e^{(1)}, w^{(1)}\right), \left(e^{(2)}, w^{(2)}\right), \ldots, \left(e^{t)}, w^{(t)}\right), \ldots$$

where $e^{(t)}$ is the edge to which the weight $w^{(t)} \in [0, 1]$ is added at time $t$. Our primary focus is graph analysis on the graph formed by the last $W$ updates such that the privacy of the entire stream is preserved. These are standard conditions considered by previous works in this setting (Bolot et al., 2013; Chan et al., 2012).

We first define the privacy notion considered in this paper. The definition relies crucially on what we consider *neighboring* datasets. For applications in a streaming model, it is typical to consider *event-level* privacy (Bolot et al., 2013; Dwork et al., 2010a,b), where two streams are neighboring if they differed in a single update. We adopt the same notion here for streaming graphs. We say that two streams, $\mathsf{S} := \left(e^{(t)}, w^{(t)}\right)_{t \geq 0}$ and $\bar{\mathsf{S}} := \left(\bar{e}^{(t)}, \bar{w}^{(t)}\right)_{t \geq 0}$ are neighboring if they differ in a single update, i.e., an (edge, weight) tuple.

We explore various possibilities of neighboring graphs in our neighboring relation. Let $\mathcal{G}$ be the graph formed by $\mathsf{S}$ and $\bar{\mathcal{G}}$ be the graph formed by a neighboring stream $\bar{\mathsf{S}}$. Recall these graphs are the one that would have formed if we consider the entire stream. Then our neighboring relation allows $\mathcal{G}$ to contain an edge that is not present in $\bar{\mathcal{G}}$ as long as the weight on that edge is at most 1. However, in this case, all the other edge weights are equal in both $\mathcal{G}$ and $\bar{\mathcal{G}}$. Another case that is permissible in our neighboring relation are the graphs $\mathcal{G}$ and $\bar{\mathcal{G}}$ with the same edge set, but weight on one single edge differ by at most 1. On the other hand, we do not allow that $\mathcal{G}$ and $\bar{\mathcal{G}}$ differ in two edges with the sum of the difference of their weight at most 1, a neighboring relation permissible in Sealfon (2016). We next give a formal definition.

**Definition 1.** *Let $\mathfrak{G}$ be the set of graphs over $n$ vertices and positive edge weights. A randomized algorithm $M : \mathfrak{G} \to \mathfrak{R}$ (where $\mathfrak{R}$ is some arbitrary abstract range) is $(\epsilon, \delta)$-differentially private if for all pairs of graphs $\mathcal{G}, \bar{\mathcal{G}}$ in $\mathfrak{G}$ formed by neighboring streams and*

$$\forall S \subseteq \mathfrak{R}, \Pr[M(\mathcal{G}) \in S] \leq \exp(\epsilon) \Pr[M(\bar{\mathcal{G}}) \in S] + \delta,$$

*where the probability is taken over the private coin tosses of $M$.*

Note that privacy is with respect to the entire stream while accuracy is with respect to the sliding window. There are closely related privacy definitions: *continual release model* (Dwork et al., 2010a) and *pan-privacy streaming model* (Dwork et al., 2010b) are accurate for entire stream and *privacy with expiration* (Bolot et al., 2013) classify only the current window as private. Our definition can be seen as a generalization of

continual release and privacy with expiration model. Our algorithms can be extended seamlessly to these models of privacy (see Remark 1 for more discussion).

We now briefly describe the main problem that we solve. Let $L_W$ be the Laplacian of the graph formed in the last $W$ updates. We wish to find a graph $\widetilde{\mathcal{G}}$ satisfying Definition 1 and

$$(1-\rho)x^\top(L_W - \zeta L_{K_n})x \le x^\top L_{\widetilde{\mathcal{G}}} x \le (1+\rho)x^\top(L_W + \vartheta L_{K_n})x.$$

for all $x \in \mathbb{R}^n$. Here, $\zeta$ and $\vartheta$ can be thought of distortion in spectrum that we are willing to accept to preserve privacy. Ideally, these parameters should be as small as possible. Informally, we view adding $L_{K_n}$ 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.

If we restrict $x$ to be only binary vectors, then we get cut approximation. It has been also studied previously in the literature (Eliáš et al., 2020; Gupta et al., 2012). Both of these works consider the semi-definite program of Alon et al. (2006) and give an efficient algorithm to solve it. However, their algorithm does not extend to the sliding window setting.

The choice of $L_{K_n}$ is for simplicity. It 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.

### 2.1 Main Results

Our privacy mechanism is the basic Gaussian mechanism. That is, we overlay a weighted complete graph, $\mathcal{R}$, where the weight on $\mathcal{R}$ is sampled from a Gaussian distribution with appropriate variance. We call this algorithm Priv-Graph$(\mathcal{G};\epsilon,\delta)$.

**Theorem 1.** *Let $\mathcal{G}$ be an $n$ vertex graph. Then $\widetilde{\mathcal{G}} \leftarrow$ Priv-Graph$(\mathcal{G};(\epsilon,\delta))$ satisfy $(\epsilon,\delta)$-differential privacy with respect to edge-level privacy.*

*Proof.* Let $\mathcal{G}$ and $\mathcal{G}'$ be two neighboring graphs that differs in exactly one edge $e := (i,j)$ by a weight $\gamma \in (0,1]$. Let vec$(A_{\mathcal{G}})$ and vec$(A_{\mathcal{G}'})$ be the vectorization of adjacency matrices of $\mathcal{G}$ and $\mathcal{G}'$, respectively. Then

$$\text{vec}(A_{\mathcal{G}}) - \text{vec}(A_{\mathcal{G}'}) = \gamma(\bar{e}_i \otimes \bar{e}_j + \bar{e}_j \otimes \bar{e}_i).$$

It follows from above that $\|\text{vec}(A_{\mathcal{G}}) - \text{vec}(A_{\mathcal{G}'})\|_2 = \gamma\sqrt{2}$. The proof of the theorem follows from the privacy guarantee of Gaussian mechanism with $\sigma = \frac{\gamma}{\epsilon}\sqrt{2\log(1/\delta)}$ (Dwork and Roth, 2014). □

Since the graph spectrum identifies many useful graph properties and differential privacy is preserved under post-processing, we can use Theorem 2 to perform various graph analysis. For some applications, see Arora and Upadhyay (2019).

**Algorithm 1** Priv-Graph $(\mathcal{G};(\epsilon,\delta))$

**Input:** An input graph $\mathcal{G} = (V,E)$, privacy parameter $(\epsilon,\delta)$.
**Output:** A laplacian of graph $L_{\widetilde{\mathcal{G}}}$.

1: **Set** $\sigma := \frac{4\sqrt{\log(1/\delta)}}{\epsilon}$.
2: **Sample** $g_{ij} \sim \mathcal{N}(0,\sigma^2)$ for $1 \le i < j \le n$.
3: **Define** a matrix $L_{\mathcal{R}} \in \mathbb{R}^{n\times n}$ such that

$$L_{\mathcal{R}}[i,j] := \begin{cases} g_{ij} & i<j \\ g_{ji} & j<i \\ -\sum_{k\neq i} g_{ik} & i=j. \end{cases}$$

4: **Compute** the laplacian, $L_{\widetilde{\mathcal{G}}} = L_{\mathcal{G}} + L_{\mathcal{R}}$.
5: **Output:** $\widetilde{\mathcal{G}}$.

Getting an accuracy that depends only on the window size and not the length of the stream, while preserving the privacy of the entire stream, brings a new challenge. Previous work on the sliding window model of privacy considered *decayed sum* of bits (Bolot et al., 2013), a simpler task than approximating the spectrum. So it is not clear if their approach directly extends to our case (more details are in the supplementary material). Our main contribution is a novel algorithm, Sliding-Priv-Graph (see Algorithm 2), which takes as input a stream of positive weighted edges of a graph, and maintains a data structure $\mathfrak{D}_{\mathsf{priv}}$ satisfying certain spectral properties. The main result is the following theorem.

**Theorem 2.** *Let $\epsilon > 0, \delta > 0$ be the privacy parameters and $\rho \in (0,1/2)$ be the approximation parameter. Then* Sliding-Priv-Graph *maintains an $(\epsilon,\delta)$-differentially private data structure, $\mathfrak{D}_{\mathsf{priv}}$, using space complexity of $O\left(\min\left\{W, \frac{n^3\log(W)}{\rho}\right\}\right)$, and runtime $\text{poly}(n,\log(W))$. Furthermore, at any time $T>0$, $\mathfrak{D}_{\mathsf{priv}}$ can be post-processed in $O(1)$ time to output a graph $\widetilde{\mathcal{G}}$ satisfying the following utility (spectral approximation) guarantee with high probability:*

$$(1-\rho)L_W + c_1\sqrt{\frac{\log(n)\log(W/\delta)}{n\epsilon^2}}L_{K_n} \preceq L_{\widetilde{\mathcal{G}}} \preceq (1+\rho)L_W + c_2\sqrt{\frac{\log(n)\log(W/\delta)}{n\epsilon^2}}L_{K_n},$$

*where $0 < c_1 < c_2$ are absolute constants, $L_{K_n}$ and $L_{\widetilde{\mathcal{G}}}$ are Laplacians of the graphs $K_n$ and $\widetilde{\mathcal{G}}$, and $L_W$ is the Laplacian of the graph given by edges and weights over the last $W$ updates.*

The best known non-private algorithm for spectral ap-

proximation of graph in the sliding window model requires $O\left(\frac{n^2 \log W}{\rho}\right)$ space. However, these algorithms fails with privacy since they use effective resistance based sampling, which is not Lipschitz (Arora and Upadhyay, 2019). Further, the sampling probablity can itself lead to privacy leak. We believe that $O\left(\frac{\log W}{\rho}\right)$ dependence is necessary for the sliding window model even without privacy; however, we leave it whether the dependence on $n$ is cubic (as in this paper) or quadratic (matching the non-private setting) as a question for future research.

The closest problem formulation to ours is that considered in Arora and Upadhyay (2019). Theorem 2 matches their additive error (measured in terms of the weights on $K_n$) even though they work in a static graph setting. In contrast, Blocki et al. (2012) and Upadhyay (2013) output a synthetic matrix (and not necessarily a Laplacian of positive weighted graph) with same additive error measured in terms of scaling of identity matrix. Our bound are interesting when the singular values of $L_W$ is of order $\widetilde{O}(\sqrt{n}/\epsilon)$. This happens, in particular, for very well connected graphs with large edge weights. We leave it as a problem of future research whether our accuracy bounds (as well as previous results) for spectral approximation are tight.

Our accuracy analysis depends on a non-asymptotic bound on the spectrum of graphs with Gaussian weights. Using subadditivity of spectral norm and non-asymptotic bounds on the spectrum of random matrices with unit variance Gaussian entries, we show the following result:

**Theorem 3.** *Let $\mathcal{H}$ be an $n$ vertex graph with edge weights independently and identically distributed copies of Gaussian random variable with zero mean and unit variance. Then there are absolute constants $c_1, c_2, C > 0$ such that*

$$\Pr[\|L_{\mathcal{H}}\|_2 \geq C(\sqrt{n \log(n)} + t)] \leq c_1 \exp(-c_2 t^2).$$

*Proof.* First note that we can write the Laplacian of graphs as $L_{\mathcal{G}} = G - D + S$, where $G$ be the random matrix whose entries are sampled i.i.d. from $\mathcal{N}(0,1)$, $D$ and $S$ are diagonal matrices formed in the following manner:

$$D[i,i] = G[i,i], \quad S[i,i] = \sum_{j \neq i} G[i,j].$$

Now, using subadditivity of spectral norm, we have $s_{\max}(L_{\mathcal{G}}) \leq s_{\max}(G) + s_{\max}(D) + s_{\max}(S)$. We know that $s_{\max}(G) \leq 2\sqrt{n}$ with probability $1 - e^{-c_1 n}$ for Gaussian random variable (Tao, 2012). For the second term, again using Tao (2012), we have

$$s_{\max}(D) = \max_{e_i} |e_i^\top D e_i| = \max_{e_i} |e_i^\top G e_i| \leq \max_{v \in \mathbb{S}^{n-1}} |v^\top G v| = s_{\max}(G)$$

with probability $1 - e^{-c_1 n}$, where $e_i$ is the $i$-th standard basis vector. Finally, for the last term, we note that

$$s_{\max}(S) \leq \max_i \left| \sum_{j \neq i} G[i,j] \right|.$$

The standard Chernoff-Hoefding bound for the sum of random Gaussian variable gives that with probability $1 - e^{-c_2 n}$, $s_{\max}(S) \leq C\sqrt{n \log(n)}$. The theorem follows by using union bound. □

The bound also gives us new results in representation learning and signal recovery.

## 3 Proof of Theorem 2

We first describe why proving Theorem 2 requires new techniques. Current algorithms for differentially private graph analysis under edge level privacy (Blocki et al., 2012; Eliáš et al., 2020; Gupta et al., 2012; Hay et al., 2010) utilize various techniques, such as multiplicative weight updates (Gupta et al., 2012), random projections (Blocki et al., 2012), private mirror descent (Eliáš et al., 2020), and network flow (Kasiviswanathan et al., 2013). While these techniques are provably efficient in the static setting, it is not clear how to extend them to a dynamic setting such as a sliding window model. It is also not clear if we can employ existing techniques in the non-private sliding window model – they are either limited to real-valued functions that satisfies certain smoothness property (Braverman and Ostrovsky, 2010) or tailor-made for specific functions (Braverman et al., 2020, 2018).

In the sliding window setting, there has been no prior work on problems arising in private graph analysis. To the best of our knowledge, current private algorithms are limited to either counting queries (Bolot et al., 2013) or heavy hitters (Chan et al., 2012; Upadhyay, 2019) that do not extend to spectral approximation.

One of the main technical hurdles towards spectral graph analysis in the sliding window model is that one has to work with positive semidefinite matrices instead of real-valued functions. Any real-valued function $f$ satisfies the property that if $f(x) \not\leq f(y)$ then $f(y) < f(x)$. This fact is non-trivially used in existing (non-private) algorithms in the sliding window model (Braverman and Ostrovsky, 2010; Datar and Motwani, 2007). However, $A \not\preceq B$ does not imply

$B \prec A$. The second challenge is that all existing algorithms for private graph analysis (and spectral graph analysis) use techniques for which it is unclear how to update intermediate steps to reflect the deletion of an edge that has expired (falls outside the sliding window). In what follows, we give a technical overview of how we overcome these hurdles and describe the proposed algorithm.

For ease of presentation, we cover the case when the algorithm outputs only once and defer the continual release case to the supplementary material.

To motivate the one-shot algorithm, we begin with the setting when we have no space or privacy constraints. Let $\mathcal{H}_i$ be the graph formed from the edges inserted to the graph in the last $i$ updates. Then, a simple FIFO list that stores the last $W$ tuples, $\{(\mathcal{H}_1, T), \ldots, (\mathcal{H}_W, T-W+1)\}$ suffices. However, the space required is $n^2W$ and can be prohibitive for large graphs and large window size. Our key insight is that, at any time $T$, we can efficiently find a small subset of graphs $\mathcal{G}_1, \ldots, \mathcal{G}_\ell$ such that for all $\mathcal{H}_i$ in the FIFO list, there exists a $\mathcal{G}_j$ such that $(1-\rho)L_{\mathcal{H}_i} \preceq L_{\mathcal{G}_j} \preceq (1+\rho)L_{\mathcal{H}_i}$.

The rest of the section is organized as follows. We first describe the intuition behind our algorithm that finds a small set of graphs, $\mathcal{G}_1, \ldots, \mathcal{G}_\ell$, for any time $T > 0$ when privacy is not a concern (Section 3.1). We then use Theorem 3 to provide privacy guarantee ( Section 3.2). We finally show how these two steps can be combined to give a spectral approximation (Section 3.4). We will apply Theorem 3 using $K_n$ with Gaussian weights; therefore, without any loss of generality, we assume that $L_{\mathcal{H}_1}, \ldots, L_{\mathcal{H}_W}$ have a single connected component with all non-zero spectrum bounded. For a streamed graph $\mathcal{G}$, we use the notation $\widetilde{\mathcal{G}}$ to denote the graph stored by the data structure.

### 3.1 Improving space

We consider the following data structure, $\mathfrak{D}_{\text{space}}$, which at any time $T$ maintains a small list of graphs and timestamps (that we refer to as checkpoints), $\mathfrak{D}_{\text{space}} := \{(\mathcal{G}_1, t_1), \ldots (\mathcal{G}_\ell, t_\ell)\}$. The graph $\mathcal{G}_i$ is the weighted graph given by edges and weights observed during updates between times $[t_i, T]$. The choice of $t_i$'s and size of the list are dictated by *smooth Laplacian property* (more details in Section 3.3). At a high level, the smooth Laplacian property requires that the spectrum of graphs corresponding to successive checkpoints are close enough. Let $\{\lambda_1(A), \ldots, \lambda_n(A)\}$ denote the eigenvalues of an $n \times n$ symmetric matrix $A$ and let $A \not\preceq B$ denote that the matrix $B - A$ is not positive semidefinite (i.e., there exists at least one $j \in \{1, \ldots, n\}$ such that $\lambda_j(A) > \lambda_j(B)$). Then smooth Laplacian property is defined as follows:

**Definition 2** (Smooth Laplacian property)**.** *A data structure $\mathfrak{D}$ satisfies* smooth Laplacian property *if there exists an $\ell = \text{poly}(n, \log(W))$ such that $\mathfrak{D}$ satisfies the following conditions:*

1. $\mathfrak{D}$ consists of $\ell$ timestamps $\mathsf{I} := \{t_1, \ldots, t_\ell\}$ and graphs $\mathsf{S} := \{\widetilde{\mathcal{G}}_1, \ldots, \widetilde{\mathcal{G}}_s\}$.
2. At least one of the following holds:
   (a) For $1 \le i \le \ell - 1$, if $t_{i+1} = t_i + 1$, then $(1-\rho)\, L_{\widetilde{\mathcal{G}}_i} \not\preceq L_{\widetilde{\mathcal{G}}_{i+1}}$.
   (b) For some constant $0 < \rho < 1$, both of the following properties for $1 \le i \le \ell - 2$:
      i. $\mathsf{Property}_1$: $(1-\rho) L_{\widetilde{\mathcal{G}}_i} \preceq L_{\widetilde{\mathcal{G}}_{i+1}}$.
      ii. $\mathsf{Property}_2$: $(1-\rho)\, L_{\widetilde{\mathcal{G}}_i} \not\preceq L_{\widetilde{\mathcal{G}}_{i+2}}$.
3. $t_1 \le T - W + 1 \le t_2$, where $T$ is the current time.

This is a generalization of *smooth histogram property* (Braverman and Ostrovsky, 2010) from real-valued functions to graph Laplacians. It is easy to show that Definition 2 allows us to prove a bound on the space requirement. For this, note that Definition 2 guarantees that the spectrum of graphs in the data-structure decreases exponentially: for every $i \in [\ell]$, there exists a $j \in [n]$ such that the $\lambda_j(L_{\mathcal{G}_{i+2}}) < (1-\rho)\lambda_j(L_{\mathcal{G}_i}) \le \lambda_j(L_{\mathcal{G}_{i+1}})$. Since the spectrum of the graph in any window is polynomially bounded (we justify this a little later), any eigenvalue can decrease by a $(1-\rho)$ factor at most

$$O\left(\log_{(1-\rho)} W\right) = O\left(\frac{\log(W)}{\log(1-\rho)}\right) = O\left(\frac{1}{\rho}\log(W)\right)$$

times for small $\rho$. Since there are at most $n-1$ such eigenvalues, we get the desired bound on $\ell$ and that the total space required is $O\left(\frac{n^3 \log(W)}{\rho}\right)$.

### 3.2 Incorporating privacy

The data structure $\mathfrak{D}_{\mathsf{space}}$ does not preserve privacy. Moreover, we cannot just invoke Priv-Graph for every time epoch as the resulting graph Laplacians may not be positive semi-definite; therefore, the smooth Laplacian property will not hold. We modify $\mathfrak{D}_{\text{space}}$ to get a data structure, $\mathfrak{D}_{\text{priv}}$, that is differentially private. The basic observation behind our construction is as follows. Let $\mathcal{R}$ be the graph with edge weights sampled from a zero-mean Gaussian distribution with variance $\frac{4\log(1/\delta)}{\epsilon^2}$. Then the following hold for any constant $\gamma \ge 1$:

$$\frac{\gamma \left\|L_{\mathcal{R}}\right\|_2}{n} L_{K_n} - L_{\mathcal{R}} \succeq 0, \text{ and } \frac{\gamma \left\|L_{\mathcal{R}}\right\|_2}{n} L_{K_n} + L_{\mathcal{R}} \succeq 0.$$

**Algorithm 2** SLIDING-PRIV-GRAPH($\mathfrak{D}_{\text{priv}}; (e_t, w_t)$)

**Input:** A new edge-weight pair $(e_t, w_t)$ and $\mathfrak{D}_{\text{priv}}$ containing $\ell$ tuples $\left\{(\widetilde{\mathcal{G}}_1, t_1), \ldots, (\widetilde{\mathcal{G}}_\ell, t_\ell)\right\}$.

**Output:** Updated $\mathfrak{D}_{\text{priv}}$.

1: On input an edge-weight pair $(e_t, w_t)$ at time $t$,
2: $\quad \widetilde{\mathfrak{D}}_{\text{priv}} \leftarrow$ INCLUDE-EDGE($\mathfrak{D}_{\text{priv}}; (e_t, w_t)$).
3: $\quad \mathfrak{D}_{\text{priv}} \leftarrow$ MAINTAIN($\widetilde{\mathfrak{D}}_{\text{priv}}; (e_t, w_t)$).

This follows from the fact that $L_{K_n} = n\mathbb{1}_n - \bar{e}\bar{e}^\top$ and $L_{\mathcal{R}}\bar{e} = 0$, where $\bar{e}$ denotes vector of all 1's and $\mathbb{1}_n$ is the $n \times n$ identity matrix. Equivalently, $L_{K_n}$ and $L_{\mathcal{R}}$ commute and the strictly positive eigenvalues of $\nu L_{K_n}$ dominate the singular values of $L_{\mathcal{R}}$ for $\nu \geq \frac{1}{n}\|L_{\mathcal{R}}\|_2$. Since

$$\|L_{\mathcal{R}}\|_2 \leq \frac{c}{\epsilon}\sqrt{n\log(n)\log\left(\frac{W}{\delta}\right)},$$

we have

$$L_{\mathcal{R}} + C\sqrt{\frac{\log(n)}{n\epsilon^2}\log\left(\frac{W}{\delta}\right)}L_{K_n} \succeq 0 \qquad (1)$$

for large enough $C > 3c$. Now suppose the current data structure $\mathfrak{D}_{\text{priv}}$ consists of timestamps $t_1 < t_2 < \cdots < t_\ell$ and graphs $\widetilde{\mathcal{G}}_1, \ldots, \widetilde{\mathcal{G}}_\ell$. Let $\mathcal{H}_t$ be the single edge graph formed by streamed $(e_t, w_t)$ pair at time $t$. Then $\mathfrak{D}_{\text{priv}}$ is updated by first including the timestamp, $t_{\ell+1} = t$, and a graph,

$$\widetilde{\mathcal{G}}_{\ell+1} := \mathcal{H}_t \cup \mathcal{R} \cup \left(C\sqrt{\frac{\log(n)}{n\epsilon^2}\log\left(\frac{W}{\delta}\right)}K_n\right), \qquad (2)$$

where $\mathcal{R}$ and $C$ are as in equation (1). This justifies the assumption we made earlier about the connected graph with polynomially bounded eigenvalues. We then update $\widetilde{\mathcal{G}}_i \leftarrow \widetilde{\mathcal{G}}_i \cup \mathcal{H}_t$ for $i \in [\ell]$. The privacy proof follows from Theorem 1 and that we are outputting a graph only at the very end of the stream.

### 3.3 Maintaining smooth Laplacian property

The utility guarantee of our algorithm relies crucially on the fact that $\mathfrak{D}_{\text{priv}}$ satisfies the smooth Laplacian property no matter how the graph in the current window is formed. This is maintained by an algorithm SLIDING-PRIV-GRAPH (see Algorithm 2). The algorithm can be divided in two stages: Stage 1 (INCLUDE-EDGE) and Stage 2 (MAINTAIN).

**Stage 1:** This includes the effect of the new edge in the data structure. In this stage, the algorithm checks if the second checkpoint has expired. If so, it deletes the first checkpoint in line 5. This ensures that the starting index of the window lies between the first

**Algorithm 3** INCLUDE-EDGE($\mathfrak{D}_{\text{priv}}; (e_t, w_t)$)

**Input:** A new edge-weight pair $(e_t, w_t)$ and $\mathfrak{D}_{\text{priv}}$ containing $\ell$ tuples $\left\{(\widetilde{\mathcal{G}}_1, t_1), \ldots, (\widetilde{\mathcal{G}}_\ell, t_\ell)\right\}$.

**Output:** Updated $\widetilde{\mathfrak{D}}_{\text{priv}}$.

1: $\widetilde{\mathfrak{D}}_{\text{priv}} \leftarrow \mathfrak{D}_{\text{priv}}$
2: **if** $t_2 < t - W + 1$ **then**
3: $\quad$ For $1 \leq j \leq \ell - 1$,
4: $\quad\quad$ Set $t_j = t_{j+1}, \widetilde{G}_j = \widetilde{G}_{j+1}$.
5: $\quad$ Update $\ell \leftarrow \ell - 1$.
6: **end if**
7: Construct a complete graph $\mathcal{R}_{\ell+1}$ with edge weights sampled i.i.d. from $\mathcal{N}\left(0, \frac{8\log 1/\delta}{\epsilon^2}\right)$.
8: **Privatization step.** Define $\mathcal{H}_t$ be a single-edge graph with weight $w_t$ on the edge $e_t$. Set $t_{\ell+1} = t$ and

$$\widetilde{\mathcal{G}}_{\ell+1} := \mathcal{H}_t \cup \mathcal{R}_{\ell+1} \cup \left(C\sqrt{\frac{\log(n)}{n\epsilon^2}\log\left(\frac{W}{\delta}\right)}K_n\right).$$

9: Update $\widetilde{\mathfrak{D}}_{\text{priv}} \leftarrow \widetilde{\mathfrak{D}}_{\text{priv}} \cup (\widetilde{\mathcal{G}}_{\ell+1}, t_{\ell+1})$.
10: $\ell = \ell + 1$.
11: **for** $i = 1, \ldots, \ell - 1$ **do**
12: $\quad$ Update $\widetilde{\mathcal{G}}_i \leftarrow \mathcal{H}_t \cup \widetilde{\mathcal{G}}_i$. ▷ Update the graphs with new edge.
13: **end for**
14: **Return** $\widetilde{\mathfrak{D}}_{\text{priv}}$

two checkpoints as required. Thereafter, it creates a new checkpoint with a complete weighted graph and the streamed edge in line 8 of SLIDING-PRIV-GRAPH. This checkpoint is then added to $\mathfrak{D}_{\text{priv}}$. Finally, it updates all the other checkpoints in $\mathfrak{D}_{\text{priv}}$ by adding new edge to the current graphs in lines 11 to 13 of SLIDING-PRIV-GRAPH.

**Stage 2:** Once a new set of graphs are produced, some graphs in $\mathfrak{D}_{\text{priv}}$ may violate the smooth Laplacian property. This stage ensures that these graphs are removed in MAINTAIN. In particular, MAINTAIN performs a sequential check over all the checkpoints to find if the spectrum of adjacent checkpoints becomes too close (and can be safely removed). If so, it finds all such adjacent checkpoints and remove them in lines 4 and 5 of MAINTAIN. Note that we only delete graphs up to indices $j - 1$ in line 5 of MAINTAIN. This is crucial in maintaining the smooth Laplacian property. The data structure is then updated to store rest of the graphs and corresponding timestamps in lines 6 to 12.

The proof that SLIDING-PRIV-GRAPH preserves smooth Laplacian property is intricate. We prove it by exhaustive case analysis. In particular, we consider whether a checkpoint is deleted in line 5. If it is deleted, then the result follows because line 5 of

**Algorithm 4** MAINTAIN($\mathfrak{D}_{\text{priv}}; (e_t, w_t)$)

**Input:** A data structure $\widetilde{\mathfrak{D}}_{\text{priv}}$ containing $\ell$ tuples $\left\{(\widetilde{\mathcal{G}}_1, t_1), \ldots, (\widetilde{\mathcal{G}}_\ell, t_\ell)\right\}$ and pair $(e_t, w_t)$.
**Output:** Updated $\mathfrak{D}_{\text{priv}}$.
1: Mantain PSD ordering. That is, find

$$j := \min\left\{p : L_{\mathcal{G}_p} \not\succeq L_{\mathcal{G}_\ell}\right\}$$

and delete $L_{\mathcal{G}_j}, \ldots, L_{\mathcal{G}_{\ell-1}}$.
2: Set $L_{\mathcal{G}_p} = L_{\mathcal{G}_\ell}, \ell = p$.
3: **for** $i = 1, \ldots, \ell - 2$ **do**
4: Find $\widetilde{\mathcal{G}}_{i+1}, \ldots, \widetilde{\mathcal{G}}_j$ such that

$$(1-\rho)\, L_{\widetilde{\mathcal{G}}_i} \preceq L_{\widetilde{\mathcal{G}}_j} \quad \text{and}$$
$$(1-\rho)\, L_{\widetilde{\mathcal{G}}_i} \not\preceq L_{\widetilde{\mathcal{G}}_{j+1}}.$$

5: Delete $\widetilde{\mathcal{G}}_{i+1}, \ldots, \widetilde{\mathcal{G}}_{j-1}$ and set $k = 1$.
6: **while** $j + k < \ell$ **do** ▷ Reorder the indices
7: Update $\widetilde{\mathcal{G}}_{i+k} = \widetilde{\mathcal{G}}_{j+k-1}$.
8: Update $t_{i+k} = t_{j+k-1}$.
9: Update $k := k + 1$.
10: **end while**
11: $\ell = \ell + i - j + 1$. ▷ Update #checkpoints.
12: **end for**
13: **Output:** $\mathfrak{D}_{\text{priv}} := \left\{(\widetilde{\mathcal{G}}_1, t_1), \ldots, (\widetilde{\mathcal{G}}_\ell, t_\ell)\right\}$.

SLIDING-PRIV-GRAPH maintains that we delete only up to index $j-1$. If it is not deleted, then it might be the case that an update can result in a temporary violation of the properties. To prove this cannot be the case is more involved. It uses how updates occur in line 5 and the fact that graph Laplacians are positive semidefinite. In particular, we use the following facts: (i) the properties were maintained before the update, (ii) the checkpoint became a successor at some point in the past; and (iii) adding a fixed positive semidefinite matrix to a sequence of positive semidefinite matrices preserves the partial order.

### 3.4 Smooth Laplacian property implies spectral approximation

Recall that $\widetilde{\mathcal{G}}$ is the graph stored by the data structure when the underlying streamed graph is $\mathcal{G}$. Let $L_W$ be the Laplacian of the graph streamed in the current window of size $W$ and $L_{\mathcal{G}_i}$ be the one streamed during the time interval $[t_i, T]$ for $1 \le i \le \ell$. Then the accuracy of the algorithm follows by the invariants maintained by our data structure and the relation between the tuple $\{L_{\mathcal{G}_1}, L_{\mathcal{G}_2}\}$ and their private versions $\left\{L_{\widetilde{\mathcal{G}}_1}, L_{\widetilde{\mathcal{G}}_2}\right\}$. By construction, the window is sandwiched between the first and second timestamp. This implies that $L_{\mathcal{G}_2} \preceq L_W \preceq L_{\mathcal{G}_1}$. By smooth Laplacian property, $(1-\rho)L_{\widetilde{\mathcal{G}}_1} \preceq L_{\widetilde{\mathcal{G}}_2}$, where $\widetilde{\mathcal{G}}_1$ and $\widetilde{\mathcal{G}}_2$ are the graphs maintained by SLIDING-PRIV-GRAPH in the data structure $\mathfrak{D}_{\text{priv}}$. Moreover, by the definition of $\widetilde{\mathcal{G}}_1$ and $\widetilde{\mathcal{G}}_2$ and Theorem 3, we have

$$L_{\mathcal{G}_1} + (C-c)\tau L_{K_n} \preceq L_{\widetilde{\mathcal{G}}_1} \preceq L_{\mathcal{G}_1} + (C+c)\tau L_{K_n}$$
$$L_{\mathcal{G}_2} + (C-c)\tau L_{K_n} \preceq L_{\widetilde{\mathcal{G}}_2} \preceq L_{\mathcal{G}_2} + (C+c)\tau L_{K_n}.$$

It follows that

$$(1-2\rho)L_W + c_1\tau L_{K_n} \preceq L_{\widetilde{\mathcal{G}}_1} \preceq (1+2\rho)L_W + c_2\tau L_{K_n}$$

for some $\rho \in (0, 1/2)$ and constants $c_1$ and $c_2$. Defining $\widetilde{\mathcal{G}} := \widetilde{\mathcal{G}}_1$ at the end of the stream, we have the desired accuracy bound of Theorem 2 by scaling $\rho$. The privacy guarantee follows from the Gaussian mechanism and that we just output the graph once. Finally, for the efficiency guarantee, the most expensive step is Step 4. This can be done efficiently using the recent work on testing PSD ordering (Bakshi et al., 2020).

**Remark 1.** *In this section, we have concentrated on one-shot algorithm as per the definition of (Dwork et al., 2010a). That is, we publish results only at the end of the stream based on the graph formed in the last $W$ updates. Our algorithm can be easily extended to continually release (Dwork et al., 2010a) a graph at every time epoch and pan-privacy model (Dwork et al., 2010b) using standard techniques. We cover it in more details in the supplementary material.*

## 4 Other applications of Theorem 3

Other consequences of Theorem 3 is simple to describe efficient algorithms for private graph analysis that also provides better accuracy. Simplicity and adaptability are highly desirable features from a practical viewpoint and we believe that our algorithms can be employed on a large scale. We explore this with respect to cut function. For the privacy notion, we consider the standard *edge-level privacy* (Blocki et al., 2012; Dwork et al., 2014; Eliáš et al., 2020; Gupta et al., 2012).

**Cut queries.** We first consider cut queries, one of the most widely studied problems in private graph analysis. For two disjoint subsets of vertices, $S$ and $T$, of an $n$ vertex weighted graph $\mathcal{G}$, the size of $(S,T)$-cut, denoted by $\Phi_{S,T}(\mathcal{G})$, is the sum of weights of edges crossing between $S$ and $T$. Let $|S|$ denotes the number of nodes in $S$. Our algorithm, PRIV-GRAPH($\mathcal{G}; \epsilon, \delta$), outputs a synthesized graph : (i) sample a graph, $\mathcal{R}$, whose edges are sampled i.i.d. from a Gaussian distribution with variance required for $(\epsilon, \delta)$-differential privacy, and (ii) overlay $\mathcal{R}$ on $\mathcal{G}$ to get $\widetilde{\mathcal{G}}$. In other words, $L_{\widetilde{\mathcal{G}}} = L_{\mathcal{G}} + L_{\mathcal{R}}$. We show the following result:

**Theorem 4.** *Let $\mathcal{G}$ be the input graph. Then* PRIV-GRAPH *is an $(\epsilon, \delta)$-differentially private algorithm*

| | $(S, V\backslash S)$-cut | $(S,T)$-queries | Run-time |
|---|---|---|---|
| McSherry and Talwar (2007) | $n/\epsilon$ | $n/\epsilon$ | – |
| Gupta et al. (2012) | $n^{3/2}/\epsilon$ | $\sqrt{n\lvert S\rvert\lvert T\rvert}/\epsilon$ | $n^2$ |
| Blocki et al. (2012) | $\rho\Phi_S(\mathcal{G}) + \sqrt{n}\lvert S\rvert/\rho^2\epsilon$ | – | $n^{2.37}$ |
| Upadhyay (2013) | $\rho\Phi_S(\mathcal{G}) + \sqrt{n}\lvert S\rvert/\rho^2\epsilon$ | – | $n^{2+o(1)}$ |
| Dwork et al. (2014) | $\lvert S\rvert\sqrt{n}/\epsilon$ | – | $n^2$ |
| Eliáš et al. (2020) | $\sqrt{\frac{n}{\epsilon}\sum_{e\in E} w_e}$ | $\sqrt{\frac{n}{\epsilon}\sum_{e\in E} w_e}$ | poly$(n)$ |
| This work | $\lvert S\rvert\sqrt{n}/\epsilon$ | $\frac{1}{\epsilon\sqrt{\lvert S\rvert+\lvert T\rvert}}\lvert S\rvert\lvert T\rvert$ | $n^2$ |

Table 1: Comparison of algorithms for answering cut query for $\delta = \Theta(n^{-\log(n)})$. Bounds ignore multiplicative constants and $\log(n)$ term. $\mathcal{G} := (V, E)$ denotes a graph with weights $\{w_e\}_{e\in E}$, $S, T \subseteq V$ are disjoint subsets, $\Phi_S(\mathcal{G})$ is the size of cut $(S, V\backslash S)$, and $\rho, \epsilon > 0$ are small constants.

*such that for any $S \subset V$, the output $\widetilde{\mathcal{G}} \leftarrow$ Priv-Graph $(\mathcal{G}; (\epsilon, \delta))$ satisfy:*

$$\left|\Phi_{S,T}(\widetilde{\mathcal{G}}) - \Phi_{S,T}(\mathcal{G})\right| \le O\left(\frac{\sqrt{\log(n)\log(1/\delta)}|S||T|}{\epsilon\sqrt{|S|+|T|}}\right).$$

This improves upon the result of Gupta et al. (2012) for all possible sizes of $S$ and $T$ and match their bound when $|S|$ and $|T|$ are both $O(n)$. In a recent work, Eliáš et al. (2020) gave an algorithm to answer cut queries with running time that has large polynomial dependence on $n$ and additive error of $\widetilde{O}\left(\sqrt{\frac{n}{\epsilon}\sum_{e\in E} w_e}\right)$ even on a sparse graph, i.e., with $\widetilde{O}(n)$ edges. Our bound is instance-dependent but independent of the weights on the edges – we achieve better bound whenever $|S|^2 = o(\sum_{e\in E} w_e)$. As an example, for unweighted graph, we improve their result for all $(S, V\backslash S)$ queries if $|S| = o(\sqrt{|E|})$. In particular, even for sparse unweighted graph with constant degree bound, we get better result whenever $|S| = o(\sqrt{n})$.

A specical case of cut queries when $T = V\backslash S$ has been also studied (Blocki et al., 2012; Dwork et al., 2014). We improve Blocki et al. (2012) over all parameters. To compare our result with Dwork et al. (2014), let us consider their output $C = L_{\mathcal{G}} + N$. Here $N$ is a Gaussian matrix with appropriate noise to preserve differential privacy. For a set of $q$ cut queries $(S, V\backslash S)$, standard concentration bounds on Gaussian distribution implies that their approach incur an additive error of order $O(|S|\sqrt{\log(q)}/\epsilon)$. In particular, if we wish to answer all possible cut queries, it leads to $O(|S|\sqrt{n}/\epsilon)$ additive error. We refer to Table 1 for a succinct comparison.

**Spectral sparsification of graphs.** Spectral sparsification was recently studied by Arora and Upadhyay (2019). They gave a polynomial-time differentially private algorithm to output spectral sparsification of graphs by estimating the effective resistance privately and then sampling edges from a graph overlayed with an appropriately weighted complete graph. While efficient, their technique is arguably complicated and less implementation friendly. We give a significantly simpler algorithm.

Our idea is to use Priv-Graph and then apply a non-private spectral sparsification algorithm on its output. However, known non-private algorithms require a positively weighted graph. To ensure that, we use a semi-definite program that outputs a graph with only positive edge weights and achieves asymptotically the same loss in accuracy. To do this, we define the following convex set that will be used in Algorithms 5 and 6:

$$\mathbb{L}^n(\varrho) = \Bigg\{ X : X[i,j] \le 0 \quad \forall i,j \in [n] : i \ne j \\ \text{and } 0 \le \sum_{j=1}^{n} X[i,j] \le \varrho \quad \forall i \in [n] \Bigg\}. \tag{3}$$

We show the following result.

**Theorem 5.** *Let $\mathcal{G}$ be the input graph on $n$ vertices. Then* Priv-Sparsify$(\mathcal{G}; (\epsilon, \delta))$ *is a polynomial-time $(\epsilon, \delta)$-differentially private algorithm. Moreover, $\widetilde{\mathcal{G}} \leftarrow$*Priv-Sparsify$(\mathcal{G}; (\epsilon, \delta))$ *has $O\left(n/\rho^2\right)$ edges such that, with probability at least $9/10$,*

$$(1-\rho)L_{\mathcal{G}} + \tau L_{K_n} \preceq L_{\widetilde{\mathcal{G}}} \preceq (1+\rho)L_{\mathcal{G}} + \tau L_{K_n},$$

*where $\tau := O\left(\sqrt{\frac{\log(n)\log(W/\delta)}{n\epsilon^2}}\right)$.*

**Optimization problems on graphs.** Finally, we study three popular optimization problems related to cut functions of the graph: Max-Cut, Sparsest-Cut, and Edge-Expansion. We show the following:

**Theorem 6.** *There is a polynomial-time $(\epsilon, \delta)$-differentially private algorithm,* Priv-Comb-Opt, *that, for an $n$-vertex graph $\mathcal{G} := (V, E)$, outputs a partition of nodes $(S, V\backslash S)$, such that, if* flag $= 0$, *then it*

**Algorithm 5** PRIV-SPARSIFY $(\mathcal{G}; (\epsilon, \delta))$

**Input:** An input graph $\mathcal{G} = (V, E)$, privacy parameter $(\epsilon, \delta)$, non private algorithm for spectral sparsification of a graph, SPARSIFY$(\cdot)$.

**Output:** A sparse graph, $\widetilde{\mathcal{G}}$.

1: **Compute** $\widehat{\mathcal{G}} \leftarrow$ PRIV-GRAPH $(\mathcal{G}; (\epsilon, \delta))$. Let $\left\|L_{\widehat{\mathcal{G}}} - L_{\mathcal{G}}\right\|_2 = \frac{4c}{\epsilon}\sqrt{n\log(n)\log(1/\delta)}$ for some constant $c$.

2: **Compute** $L_{\widetilde{\mathcal{H}}} \leftarrow L_{\widehat{\mathcal{G}}} + \sqrt{\frac{2C\log(n)\log(1/\delta)}{n\epsilon^2}} L_{K_n}$ for a positive constant $C > 3c$.

3: **Solve** the following semidefinite program to obtain the optimal solution pair $(\bar{\gamma}, \bar{L}_{\mathcal{G}'})$:

$$\begin{aligned} \text{minimize: } & \gamma \\ \text{subject to: } & L_{\widetilde{\mathcal{H}}} - L_{\mathcal{G}'} \preceq \gamma \mathbb{1}_n, \\ & L_{\mathcal{G}'} - L_{\widetilde{\mathcal{H}}} \preceq \gamma \mathbb{1}_n, \\ & \gamma \geq 0, \\ & L_{\mathcal{G}'} \in \mathbb{L}^n(1/n), \end{aligned}$$

where $\mathbb{L}^n(\cdot)$ be as defined in equation (3).

4: **Construct** $\bar{\mathcal{G}}$ from $\bar{L}_{\mathcal{G}'}$ by setting weights for each edge $(u, v)$ of $\bar{\mathcal{G}}$ as $-\bar{L}_{\mathcal{G}'}[u, v]$.

5: **Output** $\widetilde{\mathcal{G}} \leftarrow$ SPARSIFY $(\bar{\mathcal{G}})$.

*solves* MAX-CUT *with the following guarantee:*

$$\Phi_S(\mathcal{G}) \geq (0.87856 - \rho) \cdot \max_{S \subseteq V} \Phi_S(\mathcal{G}) - O\left(\frac{\sqrt{n\log(n)\log(1/\delta)}\,|\bar{S}|}{\epsilon}\right).$$

*and if the* flag $= 1$*, then it solves* SPARSEST-CUT *with the following guarantee*

$$\frac{\Phi_S(\mathcal{G})}{|S|(n-|S|)} \leq O(\sqrt{\log n}) \cdot \min_{S \subseteq V}\left(\frac{\Phi_S(\mathcal{G})}{|S|(n-|S|)}\right) + O\left(\sqrt{\frac{\log^2 n \log(W/\delta)}{\epsilon^2 n}}\right).$$

Since EDGE-EXPANSION is a constant factor of SPARSEST-CUT (Arora et al., 2009), the above result also gives an approximation algorithm for EDGE-EXPANSION. Moreover, we get optimal multiplicative approximation, improving upon the result of Arora and Upadhyay (2019).

## 5 Conclusion and big picture

In this paper, we lay the foundation of private graph analysis in the sliding window model, a preferred model of computation for various real-world implementations. Our result shed some theoretical light on the empirical observations by setting $\epsilon \to \infty$; these observations suggest that recent data are more predictive of recent trends in dynamic social graph (Tylenda et al., 2009). Given the prevalence of graphs in big data analysis, and more specifically spectral analysis, we believe that our work would lead to more future work and more substantiation of similar empirical observations in many other areas such as modern recommendation systems (Campos et al., 2014), collaborative filtering (Koren, 2009), and financial transactions (Tsay, 2005). Two major open problems left by our work are lower bounds on the space and accuracy of private spectral approximation of a graph.

**Algorithm 6** PRIV-COMB-OPT $(\mathcal{G}; (\epsilon, \delta); \mathsf{flag})$

**Input:** An input graph $\mathcal{G} = (V, E)$, privacy parameter $(\epsilon, \delta)$, flag : 0 for MAX-CUT and 1 for SPARSEST-CUT, non-private approximation algorithms MAX$(\cdot)$ and SPARSEST$(\cdot)$.

**Output:** A partition of nodes, $S$.

1: **Compute** $\widehat{\mathcal{G}} \leftarrow$ PRIV-GRAPH $(\mathcal{G}; (\epsilon, \delta))$. Let $\left\|L_{\widehat{\mathcal{G}}} - L_{\mathcal{G}}\right\|_2 = c\sigma\sqrt{n\log(n)}$ for some constant $c$ and $\sigma = \frac{4}{\epsilon}\sqrt{\log(1/\delta)}$.

2: **Compute** $L_{\widetilde{\mathcal{H}}} \leftarrow L_{\widehat{\mathcal{G}}} + \sqrt{\frac{2C\log(n)\log(W/\delta)}{n\epsilon^2}} L_{K_n}$ for a positive constant $C > 3c$.

3: **Solve** the following semidefinite program to obtain the optimal solution pair $(\bar{\gamma}, \bar{L}_{\mathcal{G}'})$:

$$\begin{aligned} \text{minimize: } & \gamma \\ \text{subject to: } & L_{\widetilde{\mathcal{H}}} - L_{\mathcal{G}'} \preceq \gamma \mathbb{1}_n, \\ & L_{\mathcal{G}'} - L_{\widetilde{\mathcal{H}}} \preceq \gamma \mathbb{1}_n, \\ & \gamma \geq 0, \\ & L_{\mathcal{G}'} \in \mathbb{L}^n(1/n). \end{aligned}$$

4: **Construct** $\bar{\mathcal{G}}$ from $\bar{L}_{\mathcal{G}'}$ by setting weights for each edge $(i, j)$ of $\bar{\mathcal{G}}$ as $-\bar{L}_{\mathcal{G}'}[i, j]$.

5: **if** flag $= 0$ **then**

6: $\quad \bar{S} \leftarrow$ MAX$(\bar{\mathcal{G}})$

7: **else**

8: $\quad \bar{S} \leftarrow$ SPARSEST$(\bar{\mathcal{G}})$.

9: **end if**

10: **Output:** $\bar{S}$.

**Acknowledgements.** This research was supported, in part, by NSF BIGDATA award IIS-1838139, NSF CAREER award IIS-1943251, and DARPA award W911NF1820267. This work was done when JU was a postdoctoral researcher at the Johns Hopkins University and visiting Simons Institute for the Theory of Computing. Authors would like to thank Adam Smith, Michael Dinitz, and Cynthia Steinhardt for insightful discussions during the early stages of the project.

## References


Agrawal, A., Raskar, R., and Chellappa, R. (2006). What is the range of surface reconstructions from a gradient field? In *European Conference on Computer Vision*, pages 578–591. Springer.

Allen-Zhu, Z., Liao, Z., and Orecchia, L. (2015). Spectral sparsification and regret minimization beyond matrix multiplicative updates. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 237–245. ACM.

Alon, N., Makarychev, K., Makarychev, Y., and Naor, A. (2006). Quadratic forms on graphs. *Inventiones Mathematicae*, 163(3):499–522.

Arora, R. and Upadhyay, J. (2019). On differentially private graph sparsification and applications. In *Advances in Neural Information Processing Systems*, pages 13378–13389.

Arora, S., Rao, S., and Vazirani, U. (2009). Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):5.

Bai, Z. and Yin, Y. (2008). Limit of the smallest eigenvalue of a large dimensional sample covariance matrix. In *Advances In Statistics*, pages 108–127. World Scientific.

Bakshi, A., Chepurko, N., and Jayaram, R. (2020). Testing positive semi-definiteness via random submatrices. In *Foundations of Computer Science (FOCS), 2020 IEEE 61st Annual Symposium on*, pages 1191–1202. IEEE.

Bansal, N., Blum, A., and Chawla, S. (2004). Correlation clustering. *Machine learning*, 56(1-3):89–113.

Blocki, J., Blum, A., Datta, A., and Sheffet, O. (2012). The johnson-lindenstrauss transform itself preserves differential privacy. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 410–419. IEEE.

Blocki, J., Blum, A., Datta, A., and Sheffet, O. (2013). Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96. ACM.

Bolot, J., Fawaz, N., Muthukrishnan, S., Nikolov, A., and Taft, N. (2013). Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory*, pages 284–295. ACM.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Braverman, V., Drineas, P., Musco, C., Musco, C., Upadhyay, J., Woodruff, D. P., and Zhou, S. (2020). Near optimal linear algebra in the online and sliding window models. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 517–528. IEEE.

Braverman, V., Grigorescu, E., Lang, H., Woodruff, D. P., and Zhou, S. (2018). Nearly optimal distinct elements and heavy hitters on sliding windows. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 7:1–7:22.

Braverman, V. and Ostrovsky, R. (2010). Effective computations on sliding windows. *SIAM J. Comput.*, 39(6):2113–2131.

Campos, P. G., Díez, F., and Cantador, I. (2014). Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1-2):67–119.

Chan, T. H., Shi, E., and Song, D. (2011a). Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24.

Chan, T. H., Shi, E., and Song, D. (2011b). Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24.

Chan, T.-H. H., Li, M., Shi, E., and Xu, W. (2012). Differentially private continual monitoring of heavy hitters from distributed streams. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 140–159. Springer.

Charikar, M. and Wirth, A. (2004). Maximizing quadratic programs: extending grothendieck's inequality. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 54–60. IEEE.

Cohen, M. B., Lee, Y. T., Musco, C., Musco, C., Peng, R., and Sidford, A. (2015). Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190. ACM.

Cohen, M. B., Musco, C., and Musco, C. (2017). Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1758–1777.

Datar, M. and Motwani, R. (2007). The sliding-window computation model and results. In Aggarwal, C. C., editor, *Data Streams - Models and Algorithms*, pages 149–167. Springer.

Ding, B., Kulkarni, J., and Yekhanin, S. (2017). Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. (2006a). Our data, ourselves: Privacy

via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006b). Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, pages 265–284.

Dwork, C., Naor, M., Pitassi, T., and Rothblum, G. N. (2010a). Differential privacy under continual observation. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 715–724.

Dwork, C., Naor, M., Pitassi, T., Rothblum, G. N., and Yekhanin, S. (2010b). Pan-private streaming algorithms. In *ICS*, pages 66–80.

Dwork, C., Nikolov, A., and Talwar, K. (2015). Efficient algorithms for privately releasing marginals via convex relaxations. *Discrete & Computational Geometry*, 53(3):650–673.

Dwork, C. and Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

Dwork, C., Talwar, K., Thakurta, A., and Zhang, L. (2014). Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 11–20. ACM.

Eliáš, M., Kapralov, M., Kulkarni, J., and Lee, Y. T. (2020). Differentially private release of synthetic graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 560–578. SIAM.

Erlingsson, Ú., Pihur, V., and Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067. ACM.

Frank, M. R., Wang, D., Cebrian, M., and Rahwan, I. (2019). The evolution of citation graphs in artificial intelligence research. *Nature Machine Intelligence*, 1(2):79.

Girvan, M. and Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826.

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

Gupta, A., Hardt, M., Roth, A., and Ullman, J. (2013). Privately releasing conjunctions and the statistical query barrier. *SIAM Journal on Computing*, 42(4):1494–1520.

Gupta, A., Ligett, K., McSherry, F., Roth, A., and Talwar, K. (2010). Differentially private combinatorial optimization. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1106–1125. Society for Industrial and Applied Mathematics.

Gupta, A., Roth, A., and Ullman, J. (2012). Iterative constructions and private data release. In *Theory of cryptography conference*, pages 339–356. Springer.

Haney, S., Machanavajjhala, A., Abowd, J. M., Graham, M., Kutzbach, M., and Vilhuber, L. (2017). Utility cost of formal privacy for releasing national employer-employee statistics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1339–1354. ACM.

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

Hay, M., Rastogi, V., Miklau, G., and Suciu, D. (2010). Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1-2):1021–1032.

Javanmard, A., Montanari, A., and Ricci-Tersenghi, F. (2016). Phase transitions in semidefinite relaxations. *Proceedings of the National Academy of Sciences*, 113(16):E2218–E2223.

Kapralov, M., Nouri, N., Sidford, A., and Tardos, J. (2019). Dynamic streaming spectral sparsification in nearly linear time and space. *arXiv preprint arXiv:1903.12150*.

Karp, R., Elson, J., Estrin, D., and Shenker, S. (2003). Optimal and global time synchronization in sensornets. *Technical report, UCLA*.

Karwa, V., Raskhodnikova, S., Smith, A., and Yaroslavtsev, G. (2011). Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157.

Kasiviswanathan, S. P., Nissim, K., Raskhodnikova, S., and Smith, A. (2013). Analyzing graphs with node differential privacy. In *Theory of Cryptography*, pages 457–476. Springer.

Kenthapadi, K., Ambler, S., Zhang, L., and Agarwal, D. (2017). Bringing salary transparency to the world: Computing robust compensation insights via linkedin salary. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 447–455. ACM.

Khot, S., Kindler, G., Mossel, E., and ODonnell, R. (2007). Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357.

Koren, Y. (2009). Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456.

Kyng, R., Pachocki, J., Peng, R., and Sachdeva, S. (2017). A framework for analyzing resparsification algorithms. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2032–2043. SIAM.

Lee, J. D. and Maggioni, M. (2011). Multiscale analysis of time series of graphs. In *International Conference on Sampling Theory and Applications (SampTA)*.

Lee, Y. T. and Sidford, A. (2014). Path finding methods for linear programming: Solving linear programs in o (vrank) iterations and faster algorithms for maximum flow. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 424–433. IEEE.

Lee, Y. T. and Sun, H. (2015). Constructing linear-sized spectral sparsification in almost-linear time. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 250–269. IEEE.

Liu, K., Das, K., Grandison, T., and Kargupta, H. (2008). Privacy-preserving data analysis on graphs and social networks.

McSherry, F. and Talwar, K. (2007). Mechanism design via differential privacy. In *null*, pages 94–103. IEEE.

Newman, M. E. (2004). Detecting community structure in networks. *The European Physical Journal B*, 38(2):321–330.

Nikolov, A., Talwar, K., and Zhang, L. (2013). The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 351–360. ACM.

Peng, R. and Spielman, D. A. (2014). An efficient parallel solver for sdd linear systems. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 333–342. ACM.

Raskhodnikova, S. and Smith, A. (2016). Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 495–504. IEEE.

Sealfon, A. (2016). Shortest paths and distances with differential privacy. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 29–41.

Singer, A. (2011). Angular synchronization by eigenvectors and semidefinite programming. *Applied and computational harmonic analysis*, 30(1):20–36.

Song, S., Little, S., Mehta, S., Vinterbo, S., and Chaudhuri, K. (2018). Differentially private continual release of graph statistics. *arXiv preprint arXiv:1809.02575*.

Spielman, D. A. and Srivastava, N. (2011). Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926.

Spielman, D. A. and Teng, S.-H. (2011). Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025.

Tao, T. (2012). *Topics in random matrix theory*, volume 132. American Mathematical Soc.

Thakurta, A. G., Vyrros, A. H., Vaishampayan, U. S., Kapoor, G., Freudiger, J., Sridhar, V. R., and Davidson, D. (2017a). Learning new words. US Patent 9,594,741.

Thakurta, A. G., Vyrros, A. H., Vaishampayan, U. S., Kapoor, G., Freudinger, J., Prakash, V. V., Legendre, A., and Duplinsky, S. (2017b). Emoji frequency detection and deep link frequency. US Patent 9,705,908.

Tropp, J. A. (2015). An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230.

Tsay, R. (2005). *Analysis of financial time series*. Wiley series in probability and statistics. Wiley-Interscience.

Tylenda, T., Angelova, R., and Bedathur, S. (2009). Towards time-aware link prediction in evolving social networks. In *Proceedings of the 3rd workshop on social network mining and analysis*, pages 1–10.

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

Upadhyay, J. (2019). Sublinear space private algorithms under the sliding window model. In *International Conference on Machine Learning*, pages 6363–6372.

Vershynin, R. (2012). *Introduction to the non-asymptotic analysis of random matrices*, page 210268. Cambridge University Press.

Vershynin, R. (2018). *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge University Press.

Xiao, Q., Chen, R., and Tan, K.-L. (2014). Differentially private network data release via structural inference. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 911–920. ACM.