Almost linear time differentially private release of synthetic graphs

Zongrui Zou
State Key Laboratory for
Novel Software Technology
New Cornerstone Science Laboratory
Nanjing University

Jingcheng Liu
State Key Laboratory for
Novel Software Technology
New Cornerstone Science Laboratory
Nanjing University

Jalaj Upadhyay
Rutgers University

Abstract

In this paper, we give an almost linear time and space algorithms to sample from an exponential mechanism with an ℓ_1-score function defined over an exponentially large non-convex set. As a direct result, on input an n vertex m edges graph G, we present the *first* $\widetilde{O}(m)$ time and $O(m)$ space algorithms for differentially privately outputting an n vertex $O(m)$ edges synthetic graph that approximates all the cuts and the spectrum of G. These are the *first* private algorithms for releasing synthetic graphs that nearly match this task's time and space complexity in the non-private setting while achieving the same (or better) utility as the previous works in the more practical sparse regime. Additionally, our algorithms can be extended to private graph analysis under continual observation.

1 Introduction

Consider a (hypothetical) for-profit organization that has the following two use cases: (i) The users can purchase (or download) smartphone applications in their online store. (ii) The users can download music albums or subscribe to podcasts in their music store. Both these relationships can be modeled by a sparse graph, where the unweighted graph encodes whether or not a user downloaded an app or music. Such graphs (in fact, for graphs encoding most practical use cases, also see Appendix I) are usually sparse, and unweighted versions of these graphs are not useful for downstream tasks, and might already be known to the organization without a privacy umbrella. Unweighted sparse graphs are also not that interesting in practice for reasons we discuss in more detail in Section 1.2.

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

The more interesting graphs in practice are weighted graphs. For example, it stores information on the frequency of app usage, which app is used more often at a different time of the day, resources an app uses, and in conjunction with which app. Similarly, the more interesting data for a music app is whether or not a user listens to a song (and how frequently) in an album, what part of the podcast is played more often to understand engagement, etc. Analysis of such graphs is useful for content creators and app developers to develop a better application or create content that provides more engagement. These analyses are naturally cast as cut functions on these graphs.

However, providing this information without a robust privacy guarantee can lead to serious privacy concerns and even legal repercussions. To assuage these (future) leakages of information, a natural candidate is to answer these cut functions using *differential privacy* (Dwork et al., 2006b). Parameterized by privacy parameters (ε, δ), differential privacy guarantees that the output distribution of an algorithm is not sensitive to small changes in the input dataset, and has inspired a lot of further research and practical applications in industries and government agencies (see Desfontaines (2021) for an up-to-date list).

Differentially-private graph approximation is one of the most widely studied problems in differential privacy. Even though there are many differentially private algorithms for releasing synthetic graphs preserving combinatorial and algebraic properties of the input graph, these algorithms have no real-world implementations. To understand this discrepancy between the theory and practice, consider the current state-of-the-art differentially private algorithm for answering cut functions for unweighted (Eliáš et al., 2020) and weighted graphs (Liu et al., 2024)[1]. Both algorithms are based on a private mirror descent and run in time $\widetilde{O}(n^7)$, use $O(n^2)$ space, and ensure an expected error of $O(\sqrt{mn}\log^2(n/\delta))$ for constant ε^2. Here n is the number

[1]Similar or closely related issues have been pointed out to us for other known algorithms by practitioners.

[2]Spectral approximation algorithms also suffer from similar limitations of large space and running time.

of nodes and m is the number of edges. Even though it matches the lower bound for a constant probability of error, this algorithm suffers from the following main limitations:

1. Real-world graphs are usually sparse, high edge weighted with n in the orders of 10^9 (see Goswami et al. (2021); Bellingeri et al. (2023) and Appendix I). Therefore, generating a synthetic graph using the private mirror descent is practically infeasible due to the $\widetilde{O}(n^7)$ running time.

2. The error bounds given by private mirror descent are in expectation (Eliáš et al., 2020; Liu et al., 2024). In contrast, developers and creators prefer high probability bounds (or confidence intervals) as fluctuation in output is hard to interpret. One can get a high probability bound by Markov's inequality for both these algorithms but at the cost of significant degradation in accuracy as verified by the authors of Eliáš et al. (2020).

3. The synthetic graphs given by the known state-of-the-art algorithms are dense and do not preserve any combinatorial structures like *sparsity*, making subsequent analysis on them significantly more time-consuming.

Due to these limitations, profit-driven organizations prefer non-private algorithms for processing graph data, which damages user privacy. This motivates the central thesis of this paper:

> Design differentially private algorithms to release a synthetic graph that approximates the spectrum and cut functions of the graph. Further, (i) the computational time required by the curator and the analyst should be as close to the non-private setting, and (ii) the output graph should preserving the sparsity.

Since real-world graphs are sparse and answering cut queries on unweighted sparse graphs is often not that interesting in practice (also see Section 1.2), we focus on sparse-weighted graphs. Our key technical contribution that underpins fast private approximation on graphs is a sampling algorithm shown in Appendix C:

Lemma 1.1 (Informal) *Fix any $m \leq N$. Given a non-negative N-dimensional real vector $x \in \mathbb{R}^N_{\geq 0}$ with sparsity $\|x\|_0 = |\{i : x_i \neq 0\}| \leq m$, let π be the distribution such that $\pi[S] \propto e^{-\varepsilon\|x - x|S\|_1}$ with support $\{S \in \{0,1\}^N : \|S\|_0 = m\}$. Then, there is an algorithm that approximately samples from π in $\widetilde{O}(m)$ time.*

Here, $x|S$ is the restriction of x on S. That is, for any $i \in [N]$, $(x|S)_i = x_i$ if $S_i = 1$ and $(x|S)_i = 0$ otherwise. In particular, if S^* is the support of x, then $x = x|S^*$, and thus S^* maximizes the probability of being chosen by π. Our approach is the first sampling perspective to a problem where optimization perspective has been predominately used Eliáš et al. (2020); Gupta et al. (2012); Liu et al. (2024). Such a study has been very fruitful in the theory of optimization Chewi (2023, 2024); Ganesh et al. (2022). We explore some applications of Lemma 1.1 below:

1. **Spectral approximation in linear time.** One important but difficult task in the context of differential privacy is to approximately preserve the spectrum of the graph Laplacian. This is studied in both non-private (Allen-Zhu et al., 2015; Batson et al., 2012; Lee et al., 2015; Spielman and Srivastava, 2011) and private setting (Arora and Upadhyay, 2019; Blocki et al., 2012; Dwork et al., 2014; Upadhyay et al., 2021). Using Lemma 1.1, we show a linear time algorithm that takes an n vertices m edges graph of maximum *unweighted* degree d_{max}, and outputs the spectral approximation with a purely additive error of $O(d_{\mathsf{max}} \log(n/\delta))$. This improves the error rate in Liu et al. (2024) by $\log(n)$ factor, the running time from $O(n^2 m)$ to $O(m)$. It also improves Dwork et al. (2014) in the more practical setting, $d_{\mathsf{max}} = \widetilde{o}(\sqrt{n})$.

2. **Cut approximation in linear time.** Given a graph with n vertices and m weighted edges, we give *the first $\widetilde{O}(m)$ time algorithms that output a synthetic graph with at most m edges.* This implies that an analyst can compute the answer to any cut query in $O(m)$ time, resolving issues 1 and 3. Our algorithms also achieve an almost linear additive error of $3m\log(n/\delta)$ with probability $1 - o(\delta)$, which means better accuracy in the sparse regime (see Table 2 and Section 1.2), resolving issue 2. We give a more detail comparison in Section 1.2.

3. **Continual release algorithm.** Our linear time algorithms can be used to construct an algorithm for efficiently output a stream of synthetic graphs under continual observation (Chan et al., 2011; Dwork et al., 2010a) while preserving *event-level* privacy. In comparison, combining the previous state-of-the-art algorithms (Eliáš et al., 2020; Liu et al., 2024) and the standard transformation (Chan et al., 2011; Dwork et al., 2010a), one could get continual release algorithms with $O(n^2)$ update time (and $\widetilde{O}(n^{3/2})$ additive error) or $O(n^7 \log n)$ update time (and $O(\sqrt{nm})$ additive error). In section G, we show how to reduce the update time at each round to only $O(\log n)$, if the number of updates, T, is polynomial in n.

1.1 Problem definition.

We consider the class of undirected positively weighted graph defined over n vertices. We set $N = \binom{n}{2}$ throughout this paper. We represent a graph $G = (V, E)$ by a N-dimensional vector encoding the edge weights, i.e., $G \in \mathbb{R}^N_{\geq 0}$, where the e-coordinate, $G[e] = w_e \in \mathbb{R}_{\geq 0}$, is the weight of the edge, $e \in E$, for a canonical ordering on the edges. Throughout the paper, we let $E = \{1 \leq e \leq N | w_e > 0\}$ denote the edge set of G. We let $[n]$ denote the set of integers $\{1, 2, \cdots, n\}$. We use differential privacy as the measure of privacy loss, and the standard notion of neighboring for graphs known as *edge level differential privacy* (Arora and Upadhyay, 2019; Blocki et al., 2012; Dwork and Roth, 2014; Eliáš et al., 2020; Hardt and Roth, 2012; Gupta et al., 2012; Upadhyay et al., 2021): two graphs are *neighboring* if their weights differ in one pair of vertices by at most 1.

Definition 1.2 (Differential privacy (Dwork et al., 2006b)) *Fix any $\varepsilon > 0$ and $\delta \in [0, 1)$. Let $\mathcal{A} : \mathbb{R}^{\binom{n}{2}}_{\geq 0} \to \mathcal{R}$ be an algorithm mapping graphs to any output domain $\mathcal{R}$. Then $\mathcal{A}$ is (ε, δ)-*differentially private *(DP) if for any pair of neighboring graphs G, G' and any output event $S \subseteq \mathcal{R}$, it holds that $Pr[\mathcal{A}(G) \in S] \leq e^{\varepsilon} \cdot Pr[\mathcal{A}(G) \in S] + \delta$. In particular, if $\delta = 0$, we simply say $\mathcal{A}$ is ε-DP.*

We note that in Definition 1.2, the edge set (i.e., the topology) of two neighboring graphs can be different. For example, consider neighboring graphs G and G' where G' has an extra edge e with weight $G'[e] = 1$, while $G[e] = 0$. We do not assume that the number of edges, $|E|$, is public information. However, all our algorithms can be easily adapted if the number of edges is publicly known as in previous works (Arora and Upadhyay, 2019; Blocki et al., 2012; Eliáš et al., 2020; Gupta et al., 2012; Upadhyay et al., 2021).

Utility metric on spectral approximation Given a graph G, its Laplacian is defined as $L_G := D_G - A_G$, where A_G is the weighted adjacency matrix and D_G is the weighted degree matrix. Then the goal is to output a graph, $\widehat{G} = \arg\min_{\widehat{G}} \left\|L_G - L_{\widehat{G}}\right\|_2$. Here $\|\cdot\|_2$ denotes the spectral norm.

Linear queries on a graph. One relaxation of spectral approximation is *linear queries* on a graph. In this problem, we view a graph G as a vector in $\mathbb{R}^N_{\geq 0}$, and the aim to find a $\widehat{G} \in \mathbb{R}^N$ that minimizes $Eval(G, \widehat{G}) = \max_{q \in [0,1]^N} |q^\top G - q^\top \widehat{G}|$.

Utility metric on cut approximation A special case of answering linear queries on graphs is *cut approximation*, where we aim to find a $\widehat{G}$ privately such that for all disjoint $S, T \subseteq V$, it holds that

$$\left|\Phi_G(S,T) - \Phi_{\widehat{G}}(S,T)\right| \leq \xi, \text{ where}$$
$$\Phi_G(S,T) = \sum_{e=(u,v) \in S \times T} G[e]. \tag{1}$$

Here, the goal is to minimize ξ (Dwork et al., 2006b; Dwork and Roth, 2014; Eliáš et al., 2020; Gupta et al., 2012; Liu et al., 2024; Upadhyay et al., 2021). We refer to an error of this form *purely additive error*. Some previous works also require a *multiplicative error* in addition to the additive error[3] (Arora and Upadhyay, 2019; Blocki et al., 2012; McSherry and Talwar, 2007). All our algorithms achieve purely additive error. For brevity, we write $(S, V \backslash S)$-cuts as $\Phi_G(S) = \Phi_G(S, V \backslash S)$.

1.2 Overview of our results

Based on the efficient sampler for exponential mechanisms with ℓ_1 norm scoring function under the sparsity constraint (Lemma 1.1), we give a brief overview of our results on efficient private graph approximation in various settings.

Private spectral approximation in linear time and space. One of our linear time and linear space algorithms (Algorithm 2) also preserves the spectrum of L_G for graphs of maximum unweighted degree $d_{\mathsf{max}} \leq n - 1$. For a weighted graph, the *unweighted* degree of a vertex $u \in [n]$ is the number of (weighted) edges incident to u, instead of the total sum of their weights. For spectral approximation, we improve Liu et al. Liu et al. (2024) by a $\log(n)$ factor and also improve on their run time by at least an $\Omega(n^2)$ factor:

Theorem 1.3 (Informal version of Theorem 2.3) *Given privacy parameters $\varepsilon > 0$, $0 < \delta < 1$ and a graph G, there is a linear time and space (ε, δ)-differentially private algorithm that outputs a graph $\widehat{G}$ such that, with high probability, $\left\|L_G - L_{\widehat{G}}\right\|_2 = O\left(\frac{d_{\mathsf{max}} \log(n/\delta)}{\varepsilon}\right)$.*

The main idea behind the proof is to show that for maximum unweighted degree d_{max}, our algorithm actually preserves *every* normalized (S, T)-cuts (not only the maximum one) with a small purely additive error. We use this fact to bound the spectral radius of $L_G - L_{\widehat{G}}$. We compare this new upper bound with previous results in Table 1. As evident from the table, we only incur an additive error and have the best utility when $d_{\mathsf{max}} = o(\sqrt{n})$ and $\delta = n^{-O(1)}$.

Private cut approximation in linear time and space with linear error. We also give two linear time algorithms that output a synthetic graph that incurs linear error for sparse graphs. That is, we show the following theorem on differentially privately cut approximation:

Theorem 1.4 (Informal version of Theorems 2.2 and E.2) *Given $\varepsilon > 0, \delta \in (0, 1)$ and non-negative weighted n vertices m edges graph $G \in \mathbb{R}^{\binom{n}{2}}_{\geq 0}$ with edge set E and maximum unweighted degree, i.e., the maximum number of edges incident on any vertex being d_{max}. There is an $\widetilde{O}(|E|)$ time, $O(|E|)$ space, (ε, δ)-differentially private algorithm that outputs a synthetic graph $\widehat{G}$ with $O(|E|)$ edges, such that,*

[3]This has been verified by an author of Arora and Upadhyay (2019). The bound using exponential mechanism is shown in Section 3.2.4 in Blocki et al. (2012).

Table 1: The comparison of existing (ε, δ)-differentially private algorithm on spectral approximation.

Method	**Additive error on spectral approximation**	**Preserve sparsity?**	**Purely Additive?**	Run-time
JL mechanism Blocki et al. (2012)	$O\left(\frac{\sqrt{n}\log(n/\delta)}{\varepsilon}\right)$	No	No	$O(n^3)$
Analyze Gauss Dwork and Roth (2014)	$O\left(\frac{\sqrt{n}\log(n/\delta)}{\varepsilon}\right)$	No	Yes	$O(n^2)$
Liu et al. Liu et al. (2024)	$O\left(\frac{d_{\max}\log^2(n)}{\varepsilon}\right)$	Yes	Yes	$O(n^2\lvert E\rvert d_{\max})$
This paper	$O\left(\frac{d_{\max}\log(n/\delta)}{\varepsilon}\right)$	**Yes**	**Yes**	$\widetilde{O}(\lvert E\rvert)$

with high probability, $\forall S, T \subseteq V$ such that $S \cap T = \varnothing$,

$$|\Phi_G(S,T) - \Phi_{\widehat{G}}(S,T)| \leq \min\left\{3|E|, 4d_{\max}|S|, 4d_{\max}|T|\right\} \cdot \frac{\log(2n/\delta)}{\varepsilon}.$$

The known lower bound Eliáš et al. (2020) implies that, if $m = \widetilde{O}(n)$, any $(O(1), \delta)$-differentially private algorithm for answering all cut queries incurs an error $\widetilde{\Omega}(n)$. In contrast, just outputting a trivial graph incurs an $\widetilde{O}(n)$ error if the average weight is $\widetilde{O}(1)$. Therefore, from a theoretical perspective, the interesting case is when the average weight is $\widetilde{\omega}(1)$.

In addition to using minimal resources and being the first almost linear time algorithms, our algorithms also improve on the state-of-the-art algorithm (Eliáš et al., 2020; Liu et al., 2024) in terms of error bound under various settings:

1. Both Eliáš et al. (2020) and Liu et al. (2024) give the bound in expectation. When considering the high probability bound (even for a mild success probability of at least $1 - \sqrt{1/n}$), we improve on Eliáš et al. (2020) and Liu et al. (2024) irrespective of whether the graph is sparse or dense and whether it is weighted or unweighted (also see Theorem 1.5)[4]. The probability requirement here is very mild; in practice, we often require a probability of success to be at least $1 - 1/n$.

2. For $\varepsilon = \Theta(1)$, any weighted graph with average weight $W = \widetilde{\omega}(1)$, the expected error bound in previous works Eliáš et al. (2020); Liu et al. (2024) is $O(\sqrt{Wmn}\log^2(n/\delta))$ and $O(\sqrt{mn}\log^3(n/\delta))$, respectively. That is, when the error bound is guaranteed with a constant probability of error, we improve on Eliáš et al. (2020) whenever the average degree is $o(W\log^2(n/\delta))$ and Liu et al. (2024) whenever the average degree is $o(\log^4(n/\delta))$. We note that naturally occurring graphs have average degree $o(\log^2(n/\delta))$ (see Appendix I for a more detailed description).

Remark 1.5 (Discussion on high probability bound) *The bound in Eliáš et al. (2020) is in expectation and they refer to Nemirovski et al. (2009) to get a high probability bound. Nemirovski et al. (2009) presents two general ways to get a high probability bound from the expectation. The stronger condition (stated as eq. (2.50) in Nemirovski et al. (2009) and stated below) allows a high probability bound (i.e., with probability $1-\beta$) while incurring an extra $\log(1/\beta)$ factor if $\mathbb{E}\left[\exp\left(\frac{\|g\|_\infty^2}{M^2}\right)\right] = \int n^2 \cdot exp(-t) \cdot exp(t^2/M^2)$ is bounded. Here $\|g\|_\infty$ is the ℓ_∞ norm of gradients in the mirror descent. However, the algorithm in Eliáš et al. (2020) does not satisfy this stronger condition; the expectation is unbounded if M is poly-logarithmic in n. Eliáš et al. (2020) satisfies the weaker condition (stated as eq. (2.40) in Nemirovski et al. (2009)); Lemma 4.5 in Eliáš et al. (2020) implies that $\mathbb{E}[\|g\|_\infty^2] = O(\log^2 n)$. This condition though translates an expectation bound to $1-\beta$ probability bound with an extra $\frac{1}{\beta^2}$ factor in the error. In particular, if we want $\beta = o(1/\sqrt{n})$, then Eliáš et al. (2020) results in an error $O(\sqrt{mn^3/\varepsilon}\log^2(n/\delta))$. This is worse than our bound irrespective of whether the graph is weighted or unweighted or is sparse or dense.*

We give a detailed comparison with other works in Table 2. We only compare the results w.r.t. the error on $(S, V\backslash S)$-cuts instead of (S,T)-cuts, since two major results (Blocki et al., 2012; McSherry and Talwar, 2007) do not provide guarantees on (S,T)-cuts. If an algorithm preserves all $(S, V\backslash S)$-cuts with a purely additive error, then it also preserves all (S,T)-cuts with the same error because, for any disjoint $S, T \subseteq V$, $\Phi_G(S,T)$ can be computed using $\Phi_G(S, V\setminus S) + \Phi_G(T, V\setminus T) - 2\Phi_G(S\cup T, V\setminus(S\cup T))$. We also comprehensively summarize by comparing previous techniques in Appendix H.

Remark 1.6 (On the scale invariance) *Note that the work of Eliáš et al. (2020) showed a $\sqrt{\varepsilon^{-1}}$ dependency on ε for unweighted graph, while the dependency for weighted graph is ε^{-1} was recently shown in Liu et al. (2024). Indeed, if we have a dependency on W better than $\sqrt{W}$, we cannot expect to have a better dependency on ε than $\sqrt{\varepsilon^{-1}}$, otherwise, we can scale up the weight and scale down after obtaining the answer, then we will get a more accurate approximation, which violates the scale invariance.*

Optimally answering linear queries. One natural relax-

[4]The authors of Eliáš et al. (2020) verified this. Since Liu et al. (2024) uses Eliáš et al. (2020) as a subroutine, it also applies to Liu et al. (2024).

Table 2: The comparison of existing results on cut approximation (for sparse case, $|E| = \widetilde{O}(n)$).

Method	**Additive error for $(S, V\backslash S)$ cuts**	**Output a sparse graph?**	**Purely additive?**	Run-time
Exponential mechanism McSherry and Talwar (2007)	$O\left(\frac{n \cdot \log n}{\varepsilon}\right)$	Yes	No	Intractable
JL mechanism Blocki et al. (2012)	$O\left(\frac{n^{1.5} \cdot \mathtt{poly}(\log n)}{\varepsilon}\right)$	No	No	$O(n^3)$
Analyze Gauss Dwork et al. (2014)	$O\left(\frac{n^{1.5} \cdot \mathtt{poly}(\log n)}{\varepsilon}\right)$	No	Yes	$O(n^2)$
Private mirror descent Eliáš et al. (2020); Gupta et al. (2012)	$O\left(\frac{n\sqrt{W}\mathtt{poly}(\log n)}{\varepsilon^{1/2}}\right)$	No	Yes	$\widetilde{O}(n^7)$
Liu et al. Liu et al. (2024)	$O\left(\frac{n \cdot \mathtt{poly}(\log n)}{\varepsilon}\right)$	No	Yes	$\widetilde{O}(n^7)$
This paper	$O\left(\frac{n \cdot \mathbf{poly}(\log n)}{\varepsilon}\right)$	**Yes**	**Yes**	$\widetilde{O}(n)$

ation of spectral approximation is answering *linear queries* on a graph, where we view an m edges graph as an $\binom{n}{2}$ dimensional vector formed by the edge-weights, $G \in \mathbb{R}_{\geq 0}^{\binom{n}{2}}$. We show that our algorithm can actually outputs a synthetic graph privately that can be used to answer any linear query formed by a vector $q \in [0,1]^{\binom{n}{2}}$ with error $\widetilde{O}(m)$ in linear time. Note that (S,T)-cut queries are special cases of linear queries $q \in [0,1]^N$ as it can be identified as a linear query $q_{(S,T)} \in \{0,1\}^{\binom{n}{2}}$ whose e-th entry, $q_{(S,T)}[e]$ is 1 only if the edge $e = (u,v) \in S \times T$. By a packing argument, we also show that $\widetilde{O}(|E|)$ error is indeed optimal for answering all possible linear queries:

Theorem 1.7 (Informal version of Theorem F.1) *Fix $\varepsilon > 0$ and $0 < \delta < 1$. Let $\mathcal{M} : \mathbb{R}_{\geq 0}^N \to \mathbb{R}^N$ be an (ε, δ) differentially private algorithm. Then, there exists a graph G, viewed as a vector in $\mathbb{R}^N$, such that* $\mathbb{E}_{\mathcal{M}}[Eval(G, \mathcal{M}(G))] = \Omega\left(\frac{(1-\delta)|E|}{e^{\varepsilon}+1}\right)$.

Extension to continual observation. In Section G, we develop an efficient framework for private cut approximation under continual observation using the *binary mechanism* (Chan et al., 2011) and recent improvements (Chen et al., 2017; Cao et al., 2018; Fichtenberger et al., 2023; Henzinger et al., 2023, 2024; Jain et al., 2021). The main barrier in practice that prohibits transforming previous static graph algorithms into the setting under continual observation is the update-time overhead. For example, in the binary mechanism, we have to resample a synthetic graph from the partial sum of some updates at every round in $[T]$. This means that the running time per iteration will be at least $O(n^2)$ for previous static algorithms on cut approximation (Dwork et al., 2014). We significantly reduce the update-time overhead with our algorithms, thanks to their sparsity-preserving property and linear running time. In the binary mechanism, given $T \in \mathbb{N}_+$ (the number of updates), there are at most $\lceil \log_2 T \rceil$ layers, and the i-th layer (from bottom to top) contains $T/2^i$ graphs where each graph has at most 2^i edges (each update is a leaf in the computation tree). By the time guarantee of our linear time algorithms (see Algorithm 2, for example), outputting a graph with m edges needs time $O(m)$. Thus, to compute all synthetic graphs corresponding to the nodes in all layers, the running time will be $\sum_{i=0}^{\lceil \log_2 T \rceil} O(2^i) \cdot \frac{T}{2^i} = O(T \log_2 T)$. Since there are T rounds in total, then the amortized run-time for each round is $O(\log T)$. In particular, we show that if $T = \text{poly}(n)$, then there is a (ε, δ)-differentially private algorithm under continual observation such that with only $O(\log n)$ amortized run-time each round, it approximates the size of all (S,T)-cuts at each round with error at most $\widetilde{O}(m/\varepsilon \cdot (\log T \log(n/\delta))^{1.5})$, where m is the number of edges in the final state.

2 Private spectral and cut approximation

This section presents our algorithms for differentially private cut and spectral approximation. For the sake of exposition, as in Aumüller et al. (2021); Cormode et al. (2012); Eliáš et al. (2020); Gupta et al. (2010)), we first assume the number of edges, $m = |E|$ is publicly known. However, in analyzing our theorems (Appendix D), we analyze where $|E|$ is confidential. Notably, these two assumptions have no fundamental barrier since we can add a Laplace noise on $|E|$ and run existing algorithms on the perturbed value, which only incurs a small error. We give interested readers a high-level idea of our algorithm design and proofs in Appendix A.

2.1 $\widetilde{O}(m)$ time algorithm using exchange walk

The idea of our first algorithms is based on Lemma 1.1, which promises a linear time algorithm on sampling sparse candidates. In particular, we consider all possible typologies with $\widetilde{O}(m)$ edges as candidates. For any weighted graph $G = ([n], E, w)$, the algorithm can be divided into two stages: (1) Approximately sample a topology $\widehat{E}$, i.e., a subset of $[N]$ according to some predefined distribution with respect to G. (2) Publish the weights of edges in

$\widehat{E}$ by the Laplace mechanism. In particular, we define π be the distribution over the topology of graphs defined as following:

$$\forall S \in 2^{[N]} \text{ and } |S| = k, \pi[S] \propto \prod_{e \in S} \exp(\varepsilon \cdot w_e). \quad (2)$$

We explain in Appendix C this distribution is exactly the distribution defined in Lemma 1.1, if we specify x by the edge weights of the input graph. In stage 1, we sample an $S \subseteq [N]$ of size $|E|$ according to π. Intuitively, edges with larger weights should be more likely to be included. When $\varepsilon = 0$, sampling from the distribution π is equivalent to choosing an $S \in \binom{[N]}{|E|}$ uniformly at random, which satisfies perfect privacy. After that, we let the chosen $|E|$ pair of vertices be the edge set $\widehat{E}$, and add Laplace noise on each edge in $\widehat{E}$ according to their weight in G. This results in a synthetic graph with exactly $|E|$ edges, which gives probability 1 bound on the sparsity of the output graph.

We present the algorithm in Algorithm 1.

Algorithm 1: Private cut approximation by T steps of basis-exchange walk

```
Input: A graph $G \in \mathbb{R}^N_{\geq 0}$, privacy budgets $\varepsilon$, $\delta$.
Output: A synthetic graph $\widehat{G}$.
Let $S_0 \leftarrow E$;
Set $T \leftarrow O(|E| \cdot (\varepsilon + \log n + \log(1/\delta)))$;
for $t = \{1, 2, \cdots, T\}$ do
    Choose an $e \in S_{t-1}$ uniformly at random and
      update $S'_t \leftarrow S_{t-1} \backslash \{e\}$;
    Choose an element $x$ in $[N] \backslash S'_t$ with probability
      $\propto \pi(S'_t \cup \{x\})$, let $S_t \leftarrow S'_t \cup \{x\}$;
end
Let $\widehat{E} \subset [N]$ be the edge set corresponding to $S_T$ ;
for $e \in \widehat{E}$ do
    Draw an independent Laplace noise $Z \sim \mathtt{Lap}(1/\varepsilon)$
      and set $w_e \leftarrow \max\{0, w_e + Z\}$ ;
end
for $e' \in [N] \wedge e' \notin \widehat{E}$ do
    $w_{e'} \leftarrow 0$ ;
end
return $\widehat{G} = (V, \widehat{E})$.
```

Our algorithm matches the non-private setting not only in running time for outputting the synthetic graph but also for post-processing. The condition $|\widehat{E}| = O(|E|)$ means that for any linear queries $q \in [0,1]^N$, we actually need only $O(|E|)$ time to compute the answer by $\widehat{G}$, instead of $O(n^2)$ time. This matches the running time in the non-private setting. The following theorem describes the privacy, utility, and resource usage of Algorithm 1:

Theorem 2.1 *Fix any $\varepsilon > 0$, $\delta \in (0,1)$ and $\beta > 0$. There is an (ε, δ)-differentially private algorithm, such that on input an n node m vertices weighted graph $G = (V, E)$, it runs in $O(m \log(n/\delta) \log n)$ time, $O(m)$ space, and outputs a $\widehat{G} = (V, \widehat{E})$ with exactly m edges such that, with probability at least $1 - 2\beta - \frac{\delta}{1+e^{\varepsilon}}$, $Eval(G, \widehat{G}) = O\left(\frac{m \log(n/\beta)}{\varepsilon}\right)$. Further, we have*

$$\left\| L_G - L_{\widehat{G}} \right\|_2 = O\left(\frac{d_{\max} \cdot \log(n)}{\varepsilon} + \frac{\log(1/\beta) \log^2 n}{n\varepsilon^2}\right).$$

2.2 $\widetilde{O}(m)$-time algorithm using high-pass filter

In this section, we propose a new and simpler linear time algorithm while achieving a slightly worse performance (in terms of the dependency on δ) compared to the algorithm in Section 2.1. The idea is to use a large enough threshold to "filter" the edges with zero weights so that we do not have to add noise on all $O(n^2)$ pairs of vertices. A similar idea is applied in Cormode et al. (2012) to publish a histogram of datasets; however, even if $|E|$ is public, the filtering algorithm in Cormode et al. (2012) does not promise to keep $|E|$-sparsity, and the running time of their algorithm is linear time only in expectation. To resolve both issues, in Algorithm 2, we make two changes to the filtering algorithm in Cormode et al. (2012): for $\delta \in (0,1)$, we use the threshold $\frac{c \log(2n/\delta)}{\varepsilon}$ instead of $\frac{c \log(n)}{\varepsilon}$ and run the filtering algorithm only on E.

Algorithm 2: Releasing a synthetic graph based on filtering algorithm

```
Input: A graph $G \in \mathbb{R}^N_{\geq 0}$, privacy budgets $\varepsilon$, $\delta$.
Output: A synthetic graph $\widehat{G}$.
Set the threshold $t = \frac{2 \log(2n/\delta)}{\varepsilon}$;
for $e \in E$ do
    Draw an independent Laplace noise $Z \sim \mathtt{Lap}(1/\varepsilon)$
      and set $\widehat{w}_e \leftarrow w_e + Z$ ;
    If $\widehat{w}_e \leq t$, then set $\widehat{w}_e \leftarrow 0$.
end
Let $\widehat{E} = \{e \in [N] \ : \ \widehat{w}_e > 0\}$ be the new edge set ;
return $\widehat{G} = (V, \widehat{E})$.
```

Clearly, Algorithm 2 runs in linear time and linear space, and it preserves exact $|E|$-sparsity with probability 1. A less obvious fact is that Algorithm 2 also preserves (ε, δ)-differential privacy, since we run the filtering algorithm only on the confidential edge set. Intuitively, if G' is a neighboring graph of G with an extra edge with a weight less than 1, then this edge will be filtered with high probability (decided by δ). Conditioned on that edge being filtered, the output distribution will be the same for both G and G'. Theorem 2.2 proved in Appendix E formalizes this intuition.

Theorem 2.2 *For any $\varepsilon > 0$ and $\delta \in (0,1)$, Algorithm 2 preserves (ε, δ) differential privacy and, with probabil-*

ity at least $1-\delta$*, outputs a* $\widehat{G}$ *such that* $Eval(G,\widehat{G}) = O\left(\frac{|E|\log(n/\delta)}{\varepsilon}\right)$.

Algorithm 2 also preserves the spectrum of the original graph, namely the quadratic form of L_G, where $L_G \in \mathbb{R}^{n\times n}$ is the Laplacian of G. It is noteworthy that the analysis on spectral approximation is not as straightforward as Algorithm 2's appearance might suggest (see Appendix E for a proof).

Theorem 2.3 *For any* $G = ([n], E)$ *with bounded maximum unweighted degree* $d_{\mathsf{max}} \le n-1$*, with high probability, Algorithm 2 outputs a* $\widehat{G} = (V, \widehat{E})$ *such that for all* $x \in \mathbb{R}^n$ *with* $\|x\|_2 \le 1$, $|x^\top L_G x - x^\top L_{\widehat{G}} x| = O\left(\frac{d_{\mathsf{max}}\log(n/\delta)}{\varepsilon}\right)$.

In Table 3, we compare our two linear additive error approaches on cut approximation when the number of edges is a publicly known parameter. The basis-exchange walk approach and the truncated filtering approach show different trade-offs with respect to δ. As $\delta \to 0$, the accuracy of the truncated filtering algorithm gets worse, but the accuracy for the basis-exchange walk is unchanged. In contrast, the run-time of the basis-exchange walk gets worse when $\delta \to 0$, while the run-time of the truncated filtering algorithm is not affected by δ. Indeed, neither algorithms dominate the other. In Section 3, we also show that the algorithm based on the basis-exchange walk empirically achieves better accuracy in spectral approximation, albeit with a slightly larger time consumption.

3 Empirical Simulations

In this section, we implement a synthetic experiment to demonstrate the advantages of our algorithms in terms of both efficiency and accuracy. In particular, we compare our algorithms with 3 major previous algorithms for outputting a synthetic graph for cut or spectral approximation: Johnson-Lindenstrauss Transformation Blocki et al. (2012), Analyze Gauss Dwork et al. (2014) and Liu et al. Liu et al. (2024). The reason we do not compare with private mirror descent Eliáš et al. (2020) is because that Eliáš et al. (2020) does not provide any theoretical guarantee on graph spectral approximation, and evaluating the maximum error over all 2^n cuts would be intangible.

We set privacy budget as $\varepsilon = 1, \delta = n^{-10}$. For JL mechanism, we set the multiplicative factor η be 0.1. The experiment is conducted on MacBook Air M1. All test results are the average of 5 runs.

Run-time. We first evaluate the run-time for outputting a synthetic graph. For the dataset, we use Erdős Rényi random graph model $G := G(n, c/n)$ (for constant c) to generate a synthetic graph. At every different scale of the input, we always set the expected average degree be constant to make sure that with high probability, the graph is sparse. We test the run time of different algorithms on 5 different scales (from 10^2 to 10^5). From Table 4, we see that: (i) all previous algorithms become rather slow even on moderately large graphs (with 10^4 vertices) and (ii) on different scales of the input, the run-times of our algorithms exhibit an obvious linear growth trend, which verifies our theory on time complexity.

Accuracy. We show the error only for spectral approximation as estimating worst-case cut approximation is not feasible. To better study the accuracy of algorithms on weighted graphs, we use three different scales of edge weights: $W = 1$ (which is also the unweighted case), $W = \sqrt{n}$ and $W = n$. For the sake of time-consuming, we set $n = 100$. The results on spectral error $\|L_G - L_{\widehat{G}}\|_2$ are listed in Table 5. From the table, we see that with the average weight value increasing, the spectral error of our algorithms (as well as Analyze Gauss) shows virtually no change. Notably, our algorithms consistently focus its spectral error on the average degree even for graphs with significantly high weights, thus confirming our Theorem 2.3. The superiority of our algorithm over Analyze Gauss might be from its novel approach to incorporating noise. Our Algorithm 1 achieves better accuracy, our conjecture is that compared to Algorithm 2, it may have a smaller constant or that the error of Algorithm 1 does not depend on δ (see also the discussions in Section 2.2). Also, even though the JL mechanism is supposed to have asymptotically the same additive error on spectral approximation, the experiment shows that the JL mechanism produces a rather large error compared to other mechanisms. This is because the JL mechanism includes a multiplicative approximation, which worsens as the weight increases. Further, the constant in the error bound of the JL mechanism is large.

To understand the growth rate of the spectral error for different sizes, we choose to test our algorithms and Analyze Gauss on unweighted graphs within the input range from $n = 200$ to $n = 1000$. The results are listed in Table 6. That is, the error of our algorithm does not increase (and perform better) if the maximum unweighted degree is a constant as theory predicts.

4 Conclusion

In this paper, we propose private algorithms on additive cut approximation and spectral approximation that run in linear time and linear space, such that the compute time and space required by both the curator and the analyst is almost the same as in the non-private setting. One of our methods can also be applied to efficiently sample from the exponential mechanism defined over candidates with the sparsity constraint, which is a non-convex set. Given that resource matters in large-scale real industrial deployments and that real-world graphs are sparse, we believe our results will positively impact private graph analysis in real-world

Table 3: The comparison of our efficient and purely additive error approaches.

Method	**Privacy**	**Accuracy**	**Run-time**	**Space**
Basis-exchange walk (Thm. 2.1)	(ε, δ)-DP	$O\left(\frac{m \log n}{\varepsilon}\right)$	$O\left(m \log(\frac{n}{\delta})\right)$	$2m$
Truncated Filtering algorithm (Thm. 2.2)	(ε, δ)-DP	$O\left(\frac{m \log(n/\delta)}{\varepsilon}\right)$	$O(m)$	m

Table 4: The comparison in run-time.

n	**Algorithm 1**	**Algorithm 2**	**Analyze Gauss** Dwork et al. (2014)	**JL mechanism** Blocki et al. (2012)	**Liu et al.** Liu et al. (2024)
100	0.043 s	0.011 s	0.028 s	1.584 s	2.376 s
10^3	0.362 s	0.093 s	2.722 s	$>$ 300 s	$>$ 300 s
10^4	2.103 s	0.983 s	284.263 s	$>$ 300 s	$>$ 300 s
10^5	22.742 s	10.029 s	$>$ 300 s	$>$ 300 s	$>$ 300 s

industrial deployments. While the setting studied in this paper captures most of the practical query sets and naturally occurring graphs in industry, the utility of our algorithms on cut approximation is sub-optimal. it remains an open question whether there is an efficient (or even linear time) algorithm that, with probability at least $1 - o(n^{-c})$, answers all (S, T)-cut queries on a graph with the same $\widetilde{O}(\sqrt{n|E|})$ error rate as in Liu et al. Liu et al. (2024).

Table 5: The comparison in accuracy of spectral approximation (I).

W	**Algorithm 1**	**Algorithm 2**	**Analyze Gauss** Dwork et al. (2014)	**JL mechanism** Blocki et al. (2012)	**Liu et al.** Liu et al. (2024)
1	23.935	32.638	43.973	817.930	19.249
$\sqrt{n}$	30.127	31.931	45.812	1861.872	29.587
n	25.347	28.189	41.723	2769.758	25.328

Table 6: The comparison in accuracy of spectral approximation (II).

n	**Algorithm 1**	**Algorithm 2**	**Analyze Gauss**Dwork et al. (2014)
200	24.413	34.589	59.559
400	24.466	38.262	87.396
600	24.874	36.599	130.889
800	25.097	38.250	161.522
1000	25.875	38.395	178.136

Acknowledgments

Jingcheng Liu and Zongrui Zou have been supported by National Science Foundation of China under Grant No. 62472212 and the New Cornerstone Science Foundation.

References

Adiprasito, K., Huh, J., and Katz, E. (2018). Hodge theory for combinatorial geometries. *Annals of Mathematics*, 188(2):381–452.

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. and Naor, A. (2004). Approximating the cut-norm via grothendieck's inequality. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 72–80.

Anari, N., Gharan, S. O., and Vinzant, C. (2018). Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 35–46. IEEE.

Anari, N., Liu, K., Gharan, S. O., and Vinzant, C. (2019). Log-concave polynomials ii: high-dimensional walks and an fpras for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–12.

Anari, N., Liu, K., Gharan, S. O., Vinzant, C., and Vuong, T.-D. (2021). Log-concave polynomials iv: approximate exchange, tight mixing times, and near-optimal sampling of forests. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 408–420.

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

Aumüller, M., Lebeda, C. J., and Pagh, R. (2021). Differentially private sparse vectors with low error, optimal space, and fast access. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1223–1236.

Aumuller, M., Lebeda, C. J., and Pagh, R. (2022). Representing sparse vectors with differential privacy, low error, optimal space, and fast access. *Journal of Privacy and Confidentiality*, 12(2).

Bassily, R., Smith, A., and Thakurta, A. (2014). Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE.

Batson, J., Spielman, D. A., and Srivastava, N. (2012). Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721.

Bellingeri, M., Bevacqua, D., Sartori, F., Turchetto, M., Scotognella, F., Alfieri, R., Nguyen, N., Le, T., Nguyen, Q., and Cassi, D. (2023). Considering weights in real social networks: A review. *Frontiers in Physics*, 11:242.

Benczúr, A. A. and Karger, D. R. (1996). Approximating st minimum cuts in õ (n 2) time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 47–55.

Bilu, Y. and Linial, N. (2006). Lifts, discrepancy and nearly optimal spectral gap. *Combinatorica*, 26(5):495–519.

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.

Cao, Y., Yoshikawa, M., Xiao, Y., and Xiong, L. (2018). Quantifying differential privacy in continuous data release under temporal correlations. *IEEE transactions on knowledge and data engineering*, 31(7):1281–1295.

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

Chen, Y., Machanavajjhala, A., Hay, M., and Miklau, G. (2017). Pegasus: Data-adaptive differentially private stream processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1375–1388.

Chewi, S. (2023). *An optimization perspective on log-concave sampling and beyond*. PhD thesis, Massachusetts Institute of Technology.

Chewi, S. (2024). *Log Concave Sampling*. `https://chewisinho.github.io/main.pdf`.

Cormode, G., Procopiuc, C., Srivastava, D., and Tran, T. T. (2012). Differentially private summaries for sparse data. In *Proceedings of the 15th International Conference on Database Theory*, pages 299–311.

Cryan, M., Guo, H., and Mousa, G. (2019). Modified log-sobolev inequalities for strongly log-concave distributions. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1358–1370. IEEE.

Desfontaines, D. (2021). A list of real-world uses of differential privacy. `https://desfontain.es/blog/real-world-differential-privacy.html`. Ted is writing things (personal blog).

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. (2006a). Our data, ourselves: Privacy via distributed noise generation. In *Annual International Con-*

ference 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*, pages 715–724.

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., Rothblum, G. N., and Vadhan, S. (2010b). Boosting and differential privacy. In *Proc. of the 51st Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 51–60.

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.

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.

Fan, C. and Li, P. (2022). Distances release with differential privacy in tree and grid graph. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 2190–2195. IEEE.

Feder, T. and Mihail, M. (1992). Balanced matroids. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 26–38.

Fichtenberger, H., Henzinger, M., and Upadhyay, J. (2023). Constant matters: Fine-grained error bound on differentially private continual observation. In *International Conference on Machine Learning*, pages 10072–10092. PMLR.

Ganesh, A., Thakurta, A., and Upadhyay, J. (2022). Langevin diffusion: An almost universal algorithm for private euclidean (convex) optimization. *arXiv preprint arXiv:2204.01585*.

Ganev, G., Oprisanu, B., and De Cristofaro, E. (2022). Robin hood and matthew effects: Differential privacy has disparate impact on synthetic data. In *International Conference on Machine Learning*, pages 6944–6959. PMLR.

Goswami, S., Das, A. K., and Nandy, S. C. (2021). Sparsity of weighted networks: Measures and applications. *Information Sciences*, 577:557–578.

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.

Hardt, M. and Roth, A. (2012). Beating randomized response on incoherent matrices. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 1255–1268.

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.

Henzinger, M., Upadhyay, J., and Upadhyay, S. (2023). Almost tight error bounds on differentially private continual counting. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5003–5039. SIAM.

Henzinger, M., Upadhyay, J., and Upadhyay, S. (2024). A unifying framework for differentially private sums under continual observation. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 995–1018. SIAM.

Jain, P., Raskhodnikova, S., Sivakumar, S., and Smith, A. (2021). The price of differential privacy under continual observation. *arXiv preprint arXiv:2112.00828*.

Kotz, S., Kozubowski, T., and Podgórski, K. (2001). *The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance*. Number 183. Springer Science & Business Media.

Koufogiannis, F., Han, S., and Pappas, G. J. (2015). Optimality of the laplace mechanism in differential privacy. *arXiv preprint arXiv:1504.00065*.

Lee, Y. T., Peng, R., and Spielman, D. A. (2015). Sparsified cholesky solvers for sdd linear systems. *arXiv preprint arXiv:1506.08204*.

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, J., Upadhyay, J., and Zou, Z. (2024). Optimal bounds on private graph approximation. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1019–1049. SIAM.

Martinelli, F. (1999). Lectures on glauber dynamics for discrete spin models. *Lectures on probability theory and statistics (Saint-Flour, 1997)*, 1717:93–191.

McSherry, F. and Talwar, K. (2007). Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE.

Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. (2009). Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609.

Roth, E., Newatia, K., Ma, Y., Zhong, K., Angel, S., and Haeberlen, A. (2021). Mycelium: Large-scale distributed graph queries with differential privacy. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pages 327–343.

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.

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

Upadhyay, J., Upadhyay, S., and Arora, R. (2021). Differentially private analysis on graph streams. In *International Conference on Artificial Intelligence and Statistics*, pages 1171–1179. PMLR.

Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *J. of the American Statistical Association*, 60(309):63–69.

Checklist

1. For all models and algorithms presented, check if you include:
    (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
    (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
2. For any theoretical claim, check if you include:
    (a) Statements of the full set of assumptions of all theoretical results. [Yes]
    (b) Complete proofs of all theoretical results. [Yes]
    (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
    (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
    (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
    (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
    (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
    (a) Citations of the creator If your work uses existing assets. [Not Applicable]
    (b) The license information of the assets, if applicable. [Not Applicable]
    (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
    (d) Information about consent from data providers/curators. [Not Applicable]
    (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
    (a) The full text of instructions given to participants and screenshots. [Not Applicable]
    (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
    (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Technical Overview

Here, we discuss the technical ingredients behind our linear time algorithms in detail. Our two algorithms that achieve additive error in linear run-time come from simple intuitions described next: (i) efficient sampling from the exponential mechanism and (ii) high pass filter.

(1) Efficient sampling from exponential mechanism Our first sampling-based algorithm is based on simulating the exponential mechanism with a proper score function and is defined over a non-convex set. To the best of our knowledge, *this is the first algorithm that simulates the exponential mechanism over a non-convex set in almost linear time and linear space* and might be of independent interest.

For the exposition, first, we consider an input graph $G \in \mathbb{N}^{\binom{n}{2}}$ with m edges, each with a non-negative integer weight bounded by $W \in \mathbb{N}$. If m and W are publicly known, then there are at most $Q = \binom{N}{m} \cdot W^m$ (where $N = \binom{n}{2}$) ways to re-arrange edges and their weights, each of them is a possible output. We consider each of them as a candidate, and sample one of the candidates according to the exponential mechanism, with the scoring function being the maximum error in (S,T)-cuts. This scoring function is 1-Lipchitz, so privacy follows from that of the exponential mechanism. The utility follows from the standard analysis and the error is $O(\log(Q)) = O(m \cdot \log(Wn^2))$ (Dwork and Roth, 2014). For graphs in $\mathbb{R}^N_{\geq 0}$ with real-valued weights, a continuous version of the exponential mechanism gives a similar utility bound.

The above algorithm has two issues: it is computationally infeasible and the error has a dependency on $W = \|G\|_\infty$. We resolve both issues in two steps:

1. Sample and publish a topology (i.e., a new edge set) according to a predefined distribution.
2. Re-weigh the edges in the new edge set.

Step (a) can be regarded as sampling a topology according to the distribution induced by the exponential mechanism. Liu et al. (2024) shows that one can find a distribution of topologies that are computationally efficient to sample from while preserving both privacy and utility. In particular, they sample a new edge set S of size k ($k \geq |E|$) with probability proportional to

$$\forall S \subseteq [N] \text{ and } |S| = k, Pr[S] \propto \prod_{e \in S} \exp(\varepsilon \cdot w_e). \tag{3}$$

Compared to naively applying the exponential mechanism, sampling from this distribution removes the $\log(W)$ dependency on the maximum weight W. Liu et al. (2024) showed that it takes $O(mn^2)$ time to exactly sample from the distribution in Equation (3). In this paper, we show that using approximate sampling (which results in $\delta \neq 0$), this can be done in linear time.

To get a linear time algorithm that samples from such distribution, we use the Markov Chain Monte Carlo (MCMC) method, specifically, the *basis-exchange walk* on all edge sets with size k. We show that the distribution defined in Equation (3) is *strongly log-concave*. Using standard modified log-Sobolev inequality for strongly log-concave distributions (Cryan et al., 2019; Anari et al., 2021), the basis-exchange walk is rapidly mixing, and we get an almost linear time approximate sampler. To be more precise, we introduce some useful definitions and facts that help us analyze the sampling subroutine. We consider the task of sampling a configuration from the two-spin system $\{0,1\}^N$ (namely a subset S of $[N]$) according to a distribution π, where $N = \binom{n}{2}$. One of the methods to characterize distribution π is to use the generating polynomial, where

$$\text{for any } z \in \mathbb{R}^N, \quad g_\pi(z) := \sum_{S \in [N]} \pi(S) \prod_{i \in S} z_i.$$

If the degree of the polynomial $g(\cdot) : \mathbb{R}^N \to \mathbb{R}$ of each term is a constant k, then we say g is k-homogeneous. If the degree of each variable $z_1, \cdots, z_N$ is at most 1 in $g(\cdot)$, then we say g is *multiaffine*. Clearly for any distribution π on $\{0,1\}^N$, $g_\pi(\cdot)$ is multiaffine. An important property we are interested in is log-concavity. A polynomial $g(\cdot)$ with non-negative coefficients is log-concave if its logarithm is concave at $z \in \mathbb{R}^N_{\geq 0}$, i.e., the Hessian of $\log g$, $\nabla^2 \log g := (\partial_i \partial_j \log g)_{ij}$ is negative semi-definite for any $z \in \mathbb{R}^N_{\geq 0}$. Here, $\partial_i g$ is the partial derivative of g with respect to its i-th coordinate. Another equivalent condition of log-concavity is that for any $z, y \in \mathbb{R}^N_{\geq 0}$ and $\lambda \in (0,1)$, $g(z + (1-\lambda)y) \geq g(z)^\lambda g(y)^{1-\lambda}$.

We need a stronger condition than log-concavity. We say g is *strongly log-concave* if for any $I \in [N]$ and $I = \{i_1, \cdots, i_k\}$, $\partial_{i_1} \cdots \partial_{i_k} g$ is log-concave at $z \in \mathbb{R}^N_{\geq 0}$.

Given $k \in \mathbb{N}$ and $k \leq N$, basis-exchange walk is a commonly used Markov chain Monte Carlo method to sample one of the $\binom{N}{k}$ subsets of $[N]$ according to some given distribution π. We define the basis-exchange walk by the following procedure:

- Initialize $S_0 \subset [N]$ such that $|S_0| = k$.
- At time $t = 1, 2, \cdots, \infty$:
  1. Choose an $e \in S_{t-1}$ uniformly at random, let $S'_t = S_{t-1} \backslash \{e\}$.
  2. Choose an element y in $[N] \backslash S'_t$ with probability $\propto \pi(S'_t \cup \{y\})$, let $S_t = S'_t \cup \{y\}$.

It has been known that the basis-exchange walk enjoys rapid mixing if the target distribution is strongly log-concave (see also Lemma B.18).

Remark A.1 *By the analysis of privacy in Section D.2 and Theorem D.6, one can find that our technique also gives an almost linear time algorithm for approximately simulating the exponential mechanism which samples from exponentially many candidates (each represented by a vector in $\mathbb{R}^N$ for some $N \in \mathbb{N}$) with ℓ_1 norm as its scoring function under a k-sparsity ($k \leq N$) constraint. This results in an almost linear time (ε, δ)-differentially private algorithm for this task. We believe such auxiliary property of our technique is of independent interest.*

(2) High-pass filter Although it has not yet been applied to private graph analysis, for sparse datasets, by adding Laplace noise to its histogram and filtering the output by setting a threshold, it is possible to obtain a linear additive error in terms of ℓ_1 norm (Cormode et al., 2012). If we apply such ideas to the graph settings, then there is a natural algorithm that outputs a synthetic graph while preserving differential privacy and takes $O(n^2)$ time:

1. Add independent Laplace noise from $\texttt{Lap}(1/\varepsilon)$ on each pair of vertices.
2. Set $t = \frac{c \log(n)}{\varepsilon}$ for some large enough constant c.
3. For each $1 \leq e \leq \binom{n}{2}$, if the perturbed weight $\widehat{w}_e \leq t$, then set $\widehat{w}_e = 0$.

We call this strategy the "naive filtering algorithm". By an elementary argument in Section 2.2, we see that this simple scheme achieves linear error on approximating cut size.

Now if we set the threshold large enough, then with high probability, if the edge e was not present in G, then it will be filtered out. This allows one to get an expected linear time algorithm. Formally, if the number of edges, m, is public, then one can first sample an integer k from the binomial distribution $\texttt{Bin}(q, p)$, where $q = \binom{n}{2} - m$ is the number of edges with zero weight and p be the probability that $z \sim \texttt{Lap}(1/\varepsilon)$ exceeds the threshold. Cormode et al. (2012) applied this strategy to approximate the histogram of some given dataset, instead of approximating the size of (S, T)-cuts. However, since k is a random variable, such a scheme only provides an algorithm that runs in expected linear time and it cannot guarantee to preserve the m-sparsity of the output graph with probability 1, two major requirements in large-scale graph analysis.

To address this issue, we propose a truncated version of the filtering algorithm which only relies on two simple modifications: (a) Given ε, δ, change the threshold t from $\frac{c \log(n)}{\varepsilon}$ to $\frac{2c \log(n/\delta)}{\varepsilon}$, and (b) project the naive filtering algorithm to the edge set E.

We show in Appendix E that this scheme provides a differentially private algorithm that always terminates in linear time while achieving linear error on approximating the size of all (S, T)-cuts.

B Technical background

B.1 Differential privacy

Differential privacy, proposed by Dwork et al. (2006b), is a widely accepted notion of privacy. Here, we formally define differential privacy. Let $\mathcal{Y}$ be the data universe where $|\mathcal{Y}| = d$. Every dataset $D \in \mathcal{Y}^n$ with cardinality n can be equivalently written as a histogram $D \in \mathbb{N}^d$, where each coordinate of D records the number of occurrences of the corresponding element in $\mathcal{Y}$. Here, we write $\mathcal{X} \subset \mathbb{N}^d$ as the collection of datasets we are interested in. Also for $i \in [d]$, we use $D[i]$ to denote the value in i-th coordinate of D.

Definition B.1 (Neighboring dataset) *For any two datasets $x, y \in \mathcal{X}$, we say that x and y are neighboring datasets if and only if x and y differ in at most one coordinate by 1. Namely, there is an $i \in [d]$ such that $|x[i] - y[i]| \le 1$, and $x[j] = y[j]$ for all $j \in [d] \backslash \{i\}$.*

If x and y are neighboring datasets, we also write $x \sim y$. The differential privacy measures privacy loss by distributional stability between neighboring datasets:

Definition B.2 ((ε, δ)-differential privacy (Dwork et al., 2006a)) *Let $\mathcal{A} : \mathcal{X} \to \mathcal{R}$ be a (randomized) algorithm, where $\mathcal{R}$ is the output domain. For fixed $\varepsilon > 0$ and $\delta \in [0, 1)$, we say that $\mathcal{A}$ preserves (ε, δ)-differential privacy if for any measurable set $S \subset \mathcal{R}$ and any $x, y \in \mathcal{X}$ be a pair of neighboring datasets, it holds that*

$$Pr[\mathcal{A}(x) \in S] \le Pr[\mathcal{A}(y) \in S] \cdot e^{\varepsilon} + \delta.$$

Here is a useful definition related to (ε, δ)-DP:

Definition B.3 *((ε, δ)-indistinguishable) Let Ω be a ground set and μ_1, μ_2 be two distributions with support $\Omega_1 \subset \Omega$, $\Omega_2 \subset \Omega$ respectively. We say that μ_1 and μ_2 are (ε, δ)-indistinguishable for $\varepsilon > 0$ and $\delta \in (0, 1)$ if for any $S \subset \Omega$, it holds that*

$$Pr_{\mu_1}(S) \le Pr_{\mu_2}(S) \cdot e^{\varepsilon} + \delta$$

and

$$Pr_{\mu_2}(S) \le Pr_{\mu_1}(S) \cdot e^{\varepsilon} + \delta.$$

A key feature of differential privacy algorithms is that they preserve privacy under post-processing. That is to say, without any auxiliary information about the dataset, any analyst cannot compute a function that makes the output less private.

Lemma B.4 (Post processing (Dwork and Roth, 2014)) *Let $\mathcal{A} : \mathcal{X} \to \mathcal{R}$ be a (ε, δ)-differentially private algorithm. Let $f : \mathcal{R} \to \mathcal{R}'$ be any function, then $f \circ \mathcal{A}$ is also (ε, δ)-differentially private.*

Sometimes we need to repeatedly use differentially private mechanisms on the same dataset, and obtain a series of outputs.

Lemma B.5 (Basic composition lemma, (Dwork et al., 2006b)) *let D be a dataset in $\mathcal{X}$ and $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_k$ be k algorithms where $\mathcal{A}_i$ (for $i \in [k]$) preserves $(\varepsilon_i, \delta_i)$ differential privacy, then the composed algorithm $\mathcal{A}(D) = (\mathcal{A}_1(D), \cdots, \mathcal{A}_2(D))$ preserves $(\sum_{i \in [k]} \varepsilon_i, \sum_{i \in [k]} \delta_i)$-differential privacy.*

Lemma B.6 (Advanced composition lemma, (Dwork et al., 2010b)) *For parameters $\varepsilon > 0$ and $\delta, \delta' \in [0, 1]$, the composition of k (ε, δ) differentially private algorithms is a $(\varepsilon', k\delta + \delta')$ differentially private algorithm, where*

$$\varepsilon' = \sqrt{2k \log(1/\delta')} \cdot \varepsilon + k\varepsilon(e^{\varepsilon} - 1).$$

Now, we introduce some basic mechanisms that preserve differential privacy. First, we define the sensitivity of query functions.

Definition B.7 (ℓ_p-sensitivity) *Let $f : \mathcal{X} \to \mathbb{R}^k$ be a query function on datasets. The sensitivity of f (with respect to $\mathcal{X}$) is*

$$\mathsf{sens}_p(f) = \max_{\substack{x, y \in \mathcal{X}, \\ x \sim y}} \|f(x) - f(y)\|_p.$$

The Laplace mechanism is one of the most basic mechanisms to preserve differential privacy for numeric queries, which calibrates a random noise from the Laplace distribution (or double exponential distribution) according to the ℓ_1 sensitivity of the function.

Definition B.8 (Laplace distribution) *Given parameter b, Laplace distribution (with scale b) is the distribution with probability density function*

$$\mathtt{Lap}(x|b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right).$$

Sometimes we use $\mathtt{Lap}(b)$ to denote the Laplace distribution with scale b.

Lemma B.9 (Laplace mechanism) *Suppose $f : \mathcal{X} \to \mathbb{R}^k$ is a query function with ℓ_1 sensitivity $\mathsf{sens}_1(f) \le \mathsf{sens}$. Then the mechanism*

$$\mathcal{M}(D) = f(D) + (Z_1, \cdots, Z_k)^\top$$

is $(\varepsilon, 0)$-differentially private, where $Z_1, \cdots, Z_k$ are i.i.d random variables drawn from $\mathtt{Lap}(\mathsf{sens}/\varepsilon)$.

In many cases the output domain of the query function is discrete. For example, according to a private dataset, we would like to output a candidate with the highest score. In those cases, the Laplace mechanism is not applicable. A more fundamental mechanism for choosing a candidate is the *exponential mechanism*. Let $\mathcal{R}$ be the set of all possible candidates, and let $s : \mathcal{X} \times \mathcal{R} \to \mathbb{R}_{\ge 0}$ be a scoring function. We define the sensitivity of s be

$$\mathsf{sens}_s = \max_{\substack{x,y \in \mathcal{X} \\ x \sim y}} |s(x) - s(y)|.$$

Now if we want to output a candidate that minimizes the scoring function, we define the exponential mechanism $\mathcal{E}$ for input dataset $D \in \mathcal{X}$ to be $\mathcal{E}(D) :=$ Choose a candidate $r \in \mathcal{R}$ with probability proportional to $\exp\left(-\frac{\varepsilon \cdot s(D,r)}{2\mathsf{sens}_s}\right)$ for all $r \in \mathcal{R}$. We have the following lemmas for the privacy guarantee and utility of $\mathcal{E}(\cdot)$.

Lemma B.10 *The exponential mechanism $\mathcal{E}(\cdot)$ sampling in some discrete space $\mathcal{R}$ is ε-differentially private.*

Lemma B.11 *Let $\mathcal{E}(\cdot)$ be an exponential mechanism sampling in some discrete space $\mathcal{R}$. For any input dataset D and $t > 0$, let $OPT = \min_{r \in \mathcal{R}} s(D, r)$, then we have that*

$$Pr\left[s(D, \mathcal{E}(D)) \ge OPT + \frac{2\mathsf{sens}_s}{\varepsilon}(\log(|\mathcal{R}|) + t)\right] \le \exp(-t).$$

The exponential mechanism $\mathcal{E}(\cdot)$ can also be generalized to sample from in a continuous space $\mathcal{R}$. Then Lemma B.10 and Lemma B.11 can be naturally generalized to the following two lemmas:

Lemma B.12 *The exponential mechanism $\mathcal{E}(\cdot)$ sampling in some continuous space $\mathcal{R}$ is ε-differentially private.*

Fix a $D \in \mathcal{X}$, for any $t \ge 0$, we define $\mathcal{R}_t = \{r \in \mathcal{R}, s(D, r) \le OPT + t\} \subset \mathcal{R}$, and for a uniform distribution π over $\mathcal{R}$, we write $\pi(\mathcal{R}_t)$ as the probability volume of $\mathcal{R}_t$.

Lemma B.13 *Let $\mathcal{E}(\cdot)$ be an exponential mechanism sampling in some continuous space $\mathcal{R}$, and π be the uniform distribution on $\mathcal{R}$. For any input dataset D, let $OPT = \min_{r \in \mathcal{R}} s(D, r)$, then we have that*

$$Pr\left[s(D, \mathcal{E}(D)) \ge OPT + \frac{4\mathsf{sens}_s}{\varepsilon}\left(\log\left(\frac{1}{\pi(\mathcal{R}_t)}\right) + t\right)\right] \le \exp(-t).$$

B.2 Spectral graph theory

In this paper, the main application of our algorithm is to output a synthetic graph with n vertices that approximately answers all cut queries. A weighted graph $G = (V, E)$ with non-negative edges can be equivalently written as a vector $G \in \mathbb{R}^N_{\ge 0}$, where we write $N = \binom{n}{2}$, and let $G[i]$ ($i \in [N]$) be the edge weight in i-th pair of vertices. If some pair of vertices have no edge, the weight is 0. Let $A_G \in \mathbb{R}^n \times \mathbb{R}^n$ be the symmetric adjacency matrix of G. That is, $A_G[u, v] = A_G[v, u] = w_{uv}$, where $u, v \in V$ are vertices, and $w_{u,v}$ is the weight of edge uv. (If u is not adjacent with v, then $w_{uv} = 0$.) For vertex $v \in V$, denote by d_v the weighted degree of v, namely $d_v = \sum_{u \ne v} w_{vu}$. Let $D_G \in \mathbb{R}^n \times \mathbb{R}^n$ be a diagonal matrix where $D_G[i, i]$ is the weighted degree of node $i \in [n]$. Here, we define the edge adjacency matrix:

Definition B.14 (Edge adjacency matrix) *Let G be an undirected graph of n vertices and m edges. Consider an arbitrary orientation of edges, then $E_G \in \mathbb{R}^{m \times n}$ is the edge adjacency matrix where*

$$E_G[e, v] = \begin{cases} +\sqrt{w_e}, & \textit{if } v \textit{ is } e\textit{'s head,} \\ -\sqrt{w_e}, & \textit{if } v \textit{ is } e\textit{'s tail,} \\ 0, & \textit{otherwize.} \end{cases}$$

An important object of interest in graph theory is the Laplacian of a graph:

Definition B.15 (Laplacian matrix) *For an undirected graph G with n vertices and m edges, the Laplacian matrix $L_G \in \mathbb{R}^{n \times n}$ of G is*

$$L_G = E_G^\top E_G.$$

Equivalently, one can verify that $L_G = D_G - A_G$. Also, we note that for any graph G, L_G is a positive semi-definite (PSD) matrix and $L_G \mathbf{1} = 0$, where $\mathbf{1} \in \mathbb{R}^n$ is the all one vector. Let $0 = \lambda_1(G) \leq \lambda_2(G) \leq \cdots \leq \lambda_n(G)$ be the non-negative eigenvalues of L_G. The following lemma illustrates the monotonicity of eigenvalues of Laplacian:

For any vector x in $\mathbb{R}^n$, the quadratic form of L_G is $x^\top L_G x \geq 0$. In particular, one can verify that

$$x^\top L_G x = \sum_{(u,v) \in E} w_{u,v}(x(u) - x(v))^2.$$

Our main task is to preserve the cut size. For any $S, T \subset V$, we simply write $\Phi_G(S)$ be the size of $(S, V \backslash S)$-cut for a graph G, and $\Phi_G(S, T)$ be the size of (S, T)-cut. For any vertex $v \in V$, we use $\mathbf{1}_v \in \{0,1\}^n$ to denote the column vector with 1 in v-th coordinate and 0 anywhere else. Also, for a subset of vertices S we use $\mathbf{1}_S \in \{0,1\}^n$ to denote the identity vector of S. It is easy to verify that

$$\Phi_G(S) = \mathbf{1}_S^\top L_G \mathbf{1}_S.$$

Therefore, if we obtain an approximation for the quadratic form of the Laplacian matrix L_G, it is equivalent to obtaining an approximation for the $(S, V \backslash S)$-cut size.

We also need the following lemma to transform cut approximation bounds to a spectral one.

Lemma B.16 (Bilu and Linial (2006)) *Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix such that $\|A\|_\infty \leq \ell$ and all diagonal entries of A has absolute value less than $O(\alpha(\log(\ell/\alpha) + 1))$. If for any non-zero $u, v \in \{0,1\}^n$ with $u^\top v = 0$, it holds that*

$$\frac{|u^\top A v|}{\|u\|_2 \|v\|_2} \leq \alpha,$$

then the spectral radius of A is $\|A\|_2 = O(\alpha(\log(\ell/\alpha) + 1))$.

B.3 Basics in Markov Chain Monte Carlo

Let Ω be a finite state space. We write $\{X_t\}_{t \geq 0}$ be a Markov chain on Ω whose transition is determined by the transition matrix $P \in \mathbb{R}^{\Omega \times \Omega}$. We also use P to denote a Markov chain if it doesn't arise any ambiguity. The Markov chain on Ω is *irreducible* if for all $X, Y \in \Omega$, there exists a t such that $P^t(X, Y) > 0$, and it is *aperiodic* if for all $X \in \Omega$, $\gcd\{t > 0 | P^t(X, X) > 0\} = 1$. If a Markov chain is both irreducible and aperiodic, then it has a unique stationary distribution π on Ω such that $\pi = P\pi$. If a distribution π satisfies that

$$\forall X, Y \in \Omega : \pi(X)P(X, Y) = \pi(Y)P(Y, X)$$

then we say the Markov chain P is reversible with respect to π, which also implies that π is the stationary distribution of P.

We use the *mixing time* to measure the convergence rate of a Markov chain. For two distributions μ, ν on Ω, let

$$\|\mu, \nu\|_{TV} = \frac{1}{2}\|\mu - \nu\|_1 = \max_{S \subset \Omega} |\mu(S) - \nu(S)|$$

denote the total variation distance. Then, the mixing time of P with stationary distribution π is defined by

$$\forall 0 < \varepsilon < 1 : T_{\text{mix}}(\varepsilon) = \max_{x \in \Omega} \min \left\{ t \mid d_{TV}(\pi, P^t(x, \cdot)) \right\}.$$

Here, $P^t(x, \cdot)$ is the distribution of X_t in $\{X_t\}_{t \geq 0}$ starting with $X_0 = x$.

In this paper, given a positive integer, $N \in \mathbb{N}_+$, we are interested in sampling a subset of $[N]$ with some specific size. For some $k \leq N$, we use $\binom{[N]}{k}$ to denote the collection of $S \subset [N]$ and $|S| = k$. A widely used Markov chain for sampling a subset of $[N]$ is *basis-exchange walk* (Feder and Mihail (1992); Anari et al. (2019); Cryan et al. (2019); Anari et al. (2021)), which has been formally defined in Appendix A. It can actually be used to accomplish a more general task: sampling independent sets in a matroid. One can verify that the basis-exchange walk is a special case of *Glauber dynamics* (Martinelli (1999)), and thus it is irreducible and aperiodic. Also, its unique stationary distribution is π. Note that π is a distribution on $2^{[N]}$. We use the generating polynomial $g_\pi(\cdot)$ to characterize distribution π, where

$$g_\pi(z) := \sum_{S\in[N]} \pi(S) \prod_{i\in S} z_i.$$

We are interested in the strong log-concavity of $g(\cdot)$. A non-negative function $g : \mathbb{R}^n \to \mathbb{R}$ is *log-concave* if its domain is a convex set, and if it satisfies the inequality

$$g(tx + (1-t)y) \geq (g(x))^t (g(y))^{1-t}$$

for all x, y in the domain of g. Now, strong log-concavity is defined as follows:

Definition B.17 (Strong log-concavity) *Let $g : \mathbb{R}^N \to \mathbb{R}$ be a polynomial. We say g is strongly log-concave if for any $I \in [N]$, $\partial_I g$ is log-concave at $z \in \mathbb{R}^N_{\geq 0}$. Here, $\partial_I g := \partial_{i_1} \cdots \partial_{i_k} g$ for $I = \{i_1, \cdots, i_k\}$.*

Let $P \in \mathbb{R}^{2^N \times 2^N}$ be the Markov chain that transits subsets in $[N]$. It has been known that distributions with strong log-concavity satisfy good properties for mixing time:

Lemma B.18 (Cryan et al. (2019); Anari et al. (2021)) *Let π be a k-homogeneous strongly log-concave distribution on $\binom{[N]}{k}$, and P be the corresponding basis-exchange walk, then for any $\alpha > 0$ and $S \in \binom{[N]}{k}$,*

$$\|P^t(S,\cdot), \pi\|_{TV} \leq \alpha$$

as long as

$$t \geq r \cdot (\log\log(\pi(S)^{-1}) + 2\log(\alpha^{-1}) + \log 4).$$

Thus, we see that the mixing time of basis-exchange walk on subsets of size r can be bounded roughly by $O(r(\log\log \pi(S)^{-1}) + \log(\alpha^{-1}))$, where S is the initial state.

C Proof of Lemma 1.1

We first give a formal restatement of Lemma 1.1 here:

Lemma C.1 (Restatement of Lemma 1.1) *Fix k, N, ε such that $\varepsilon \geq 0$, $k, N \in \mathbb{N}$ and $k \leq N$. Given a vector $x \in \mathbb{R}^N$ in the positive quadrant such that $\|x\|_0 \leq k$, let π be the distribution*

$$\pi[S] \propto e^{-\varepsilon\|x - x|S\|_1}$$

with support $\{S \in \{0,1\}^N : \|S\|_0 = k\}$. Then, for any $0 < \delta < 1$, there exists an $\widetilde{O}_\delta(k)$ time random algorithm $\mathcal{A}$ that outputs an element in the same support such that if π' is the output distribution of $\mathcal{A}$, then $\|\pi - \pi'\|_{TV} \leq \frac{\delta}{e^{2\varepsilon}+1}$.

Here, $x|S$ is the restriction of x on S. That is, for any $i \in [N]$, $(x|S)_i = x_i$ if $S_i = 1$ and $(x|S)_i = 0$ otherwise. In particular, if S^* is the support of x, then $x = x|S^*$ and thus S^* maximizes the probability of been chosen. We first show how the distribution defined in Lemma C.1 is connected to the distribution in Equation (2).

For any fixed $x \in \mathbb{R}^N_{\geq 0}$, we have that

$$\begin{aligned}
\mathsf{Pr}[S] \propto \exp(-\varepsilon\|x - x|S\|_1) &= \exp\left(-\varepsilon \sum_{i\in[N]} x_i - (x|S)_i\right) \\
&= \exp\left(-\varepsilon \sum_{i\notin S} x_i\right) \propto \exp\left(\varepsilon \sum_{i\in[N]} x_i\right) \cdot \exp\left(-\varepsilon \sum_{i\notin S} x_i\right) \\
&= \exp\left(\varepsilon \sum_{i\in S} x_i\right) = \prod_{i\in S} \exp(\varepsilon \cdot x_i).
\end{aligned}$$

Here, we write $i \in S$ if $S_i = 1$ otherwise $i \notin S$. Thus, if we let $N = \binom{n}{2}$ and x be the weights of the input graph, then sampling from the distribution in Lemma C.1 is equivalent to sampling from the distribution in Equation (2). To sample from the distribution

$$\forall S \in 2^{[N]} \text{ and } |S| = k, \pi[S] \propto \prod_{e \in S} \exp(\varepsilon \cdot x_e), \tag{4}$$

we apply the basis-exchange walk which is commonly used in Markov chain Monte Carlo method. The algorithm is summarized in Algorithm 3.

Algorithm 3: Basis-exchange walk

```
Input: A parameter $k$, time limit $T$
Output: A subset $S \subset [N]$.
Initialize $S_0 \subseteq N$ such that $|S_0| = k$;
for $t = \{1, 2, \cdots, T\}$ do
    Choose an $e \in S_{t-1}$ uniformly at random, let $S'_t \leftarrow S_{t-1} \backslash \{e\}$;
    Choose an element $y$ in $[N] \backslash S'_t$ with probability $\propto \pi(S'_t \cup \{y\})$, let $S_t \leftarrow S'_t \cup \{y\}$;
end
return $S_T$.
```

To analyze how the distribution of S_T approximates π in Equation (4), we first show that the target distribution π satisfies strong log-concavity.

Lemma C.2 *Fix any $x \in \mathbb{R}^N_{\geq 0}$, $k \geq 0, \varepsilon > 0$, the generating polynomial of distribution π defined in Equation* (4) *is strongly log-concave at any $z \in \mathbb{R}^N_{\geq 0}$.*

Proof: We use the celebrated log-concavity of the basis generating polynomial of any matroids (see, e.g., Anari et al. (2018); Adiprasito et al. (2018)). They imply that, for any k, N such that $k \leq N$, if $\mathcal{B} \subseteq \binom{[N]}{k}$ is a collection of subsets of size k, then the polynomial

$$\widehat{g}(z_1, \cdots, z_N) := \sum_{S \in \mathcal{B}} \prod_{e \in S} z_e \tag{5}$$

is log-concave. Let $g(\cdot) : \mathbb{R}^N_{\geq 0} \to \mathbb{R}$ be the generating polynomial of distribution π defined in Equation (7). Let $Z(G, \varepsilon)$ be the partition function of π with respect to G and ε, and let $f(e) = \exp(\varepsilon \cdot w_e)$ for any $e \in [N]$. That is, $Z(G, \varepsilon) = \sum_{S \in \binom{N}{k}} \prod_{e \in S} f(e)$. We use $M : \mathbb{R}^N \times \mathbb{R}^N$ to denote a diagonal matrix where $M_{ee} = f(e)$ for $e \in [N]$. Then, we have that for any $z \in \mathbb{R}^N_{\geq 0}$,

$$\begin{aligned} g(z) &= \sum_{S \in \binom{N}{k}} \pi(S) \prod_{e \in S} z_e = \frac{\sum_{S \in \binom{N}{k}} \prod_{e \in S} f(e) \prod_{e \in S} z_e}{Z(G, \varepsilon)} \\ &= \frac{1}{Z(G, \varepsilon)} \sum_{S \in \binom{N}{k}} \prod_{e \in S} f(e) z_e \end{aligned}$$

Thus, for any $z, y \in \mathbb{R}^N_{\geq 0}$ and $\lambda \in (0, 1)$, let $\mathcal{B} = \binom{[N]}{k}$, then for $\bar{z} = \lambda z + (1 - \lambda) y$,

$$\begin{aligned} g(\bar{z}) &= \frac{1}{Z(G, \varepsilon)} \sum_{S \in \binom{N}{k}} \prod_{e \in S} f(e)(\lambda z_e + (1 - \lambda) y_e) \\ &= \frac{1}{Z(G, \varepsilon)} \cdot \widehat{g}(\lambda M z + (1 - \lambda) M y) \\ &\geq \left(\frac{1}{Z(G, \varepsilon)} \cdot \widehat{g}(M z) \right)^{\lambda} \left(\frac{1}{Z(G, \varepsilon)} \cdot \widehat{g}(M y) \right)^{1 - \lambda} \\ &= g^{\lambda}(z) \cdot g^{1 - \lambda}(y). \end{aligned}$$

Here, $\widehat{g}(\cdot)$ is defined in Equation (5). Thus, $g(\cdot)$ is log-concave. Further, it's easy to verify that for any $i \in [N]$, $\partial_i g$ is the generating polynomial of distribution on $\binom{[N] \backslash \{e\}}{k-1}$ and is $(k-1)$-homogeneous. By the same argument, $\partial_i g$ is also log-concave. By induction, we have that for any $I \in [N]$, $\partial_I g$ is log-concave at $z \in \mathbb{R}^N_{\geq 0}$. Thus, $g(\cdot)$ is strongly log-concave at $z \in \mathbb{R}^N_{\geq 0}$. $\square$

This allows us to say that the distribution of the output of the basis-exchange walk in Algorithm 1 is close to π under total variation distance. Fix a $T \in \mathbb{N}$, recall that S_T is the edge set chosen by Algorithm 3. The following lemma shows rapid mixing for strongly log-concave distributions.

Lemma C.3 *Given $k \geq 0$. Let π_T be the distribution of S_T, then there is a c such that if $T = c \cdot k(\varepsilon + \log n + \log(1/\delta))$, we have that $\|\pi_T - \pi\|_{TV} \leq \frac{\delta}{e^{2\varepsilon}+1}$.*

Proof: Let $P(G)$ be the basis exchange described in Algorithm 1. By Cryan et al. (2019); Anari et al. (2021) (see Lemma B.18), we just need to set

$$T = k\left(\log\log\left(\frac{1}{\pi(S_0)}\right) + 2\log\left(\frac{1}{\alpha}\right) + \log 4\right), \tag{6}$$

then the total variation distance between π_T and π is at most $\alpha := \delta/(e^{\varepsilon'}+1)$. Note that there are at most N^k elements in $\mathtt{supp}(\pi)$ where $N = O(n^2)$, and since $|E| \subset S_0$, then

$$\pi(S_0) = \max_{S \in \binom{[N]}{k}} \pi(S).$$

Thus, $\log\log(\pi(S_0)^{-1}) \leq O(\log k + \log N) = O(\log n)$. Setting $\alpha = \frac{\delta}{e^{2\varepsilon}+1}$ in equation (6) completes the proof. $\square$

Finally, we analysis the run-time of Algorithm 3. In particular, we show that after proper linear time initialization, we need only $O(|E|)$ space and $O(\log n)$ time to perform each step of the basis-exchange walk, which finalize the proof of Lemma C.1.

Run time analysis of Algorithm 3. We make three standard assumptions on our computation model:

1. Storing the weight on each edge requires $O(1)$ space. (For example, in a floating-point style.) Note that this assumption is typically used in the non-private setting as well.
2. Tossing a biased coin (i.e., sampling from the Bernoulli distribution) with bias p needs time $O(1)$ for any $p \in (0,1)$.
3. Given $b > 0$, sampling a value from the random variable with density function $\mathtt{PDF}(x) = \frac{1}{2b}\exp(-|x|/b)$ needs time $O(1)$.

Let $f(e) = \exp(\varepsilon \cdot w_e)$ for any $e \in [N]$. We first prove the running time and space requirement of our algorithm, i.e, with high probability, for any $\delta \in (0,1)$, Algorithm 3 runs in time $O(|E|\log(n/\delta)\log n)$, and the space complexity of Algorithm 3 is $O(|E|)$. First, in line 1 of Algorithm 3, we choose a subset S_0 arbitrarily, which can be implemented efficiently.

Apart from that, the only non-trivial part lies in line 3 and line 4, where the algorithm drops an element uniformly at random and samples an element in $[N]\backslash S'_t$. Note that for any $y \in [N]\backslash S'_t$,

$$\pi(S'_t \cup \{y\}) \propto \prod_{e \in S'_t \cup \{y\}} f(e) \propto f(y),$$

since $\prod_{e \in S'_t} f(e)$ is same for all $y \in [N]\backslash S'_t$. Recall that for any $e \notin E$, the probabilistic weight is always $f(e) = \exp(0 \cdot \varepsilon) = 1$, then we only need to read the weights in the edge set at the start of the algorithm, which needs time complexity of $O(|E|)$ (here, we assume that we need $O(1)$ space to store the weight of each edge). In the t-th step of the basis-exchange walk, we actually do the following:

1. Let $E_t = ([N]\backslash S'_t) \cap E$, and $\bar{E}_t = ([N]\backslash S'_t)\backslash E_t$.
2. Let y^* be any element such that $y^* \in \bar{E}_t$ (we note that $f(a) = 1$ for all $a \in \bar{E}_t$). Define μ_t be the distribution on $Z_t = E_t \cup \{y^*\}$, where

$$\mu_t(z) \propto \begin{cases} f(z), & z \in E_t, \\ |\bar{E}_t|, & z = y^*. \end{cases}$$

3. Sample an element z in Z_t according to μ_t.
4. If $z \in E_t$, let $y \leftarrow z$ (in line 4). If $z = y^*$, then sample an $e \in \bar{E}_t$ uniformly at random, and let $y \leftarrow e$.

By repeatedly tossing a biased coin, we can sample an element from a collection of r elements in time $O(\log r)$, if the probability density is already known. Thus, after the initialization, we need only $O(|E|)$ space and $O(\log n)$ time to perform each step of the basis-exchange walk.

D Missing Proofs in Section 2.1

In this section, we give the proof Theorem 2.1. Throughout this section, we actually analyze a harder case where the number of edges is confidential. Then, theorems in this section can also be easily adopted to the setting where $|E|$ is not confidential (Algorithm 1). For the case where $|E|$ is confidential, we give the following algorithm that is very similar with Algorithm 1 except the initialization phase. The algorithm is summarized in Algorithm 4.

```
Algorithm 4: Private cut approximation by $T$ steps of basis-exchange walk(with confidential $|E|$)
Input: A graph $G \in \mathbb{R}^N_{\geq 0}$, privacy budgets $\varepsilon$, $\delta$ and parameter $\beta \in (0,1)$.
Output: A synthetic graph $\widehat{G}$.
Let $c = \log(1/\beta)$ and $k \leftarrow \min\{N, |E| + \mathtt{Lap}(1/\varepsilon) + c/\varepsilon\}$;
Choose an arbitrary $S_0 \subset [N]$ and $|S_0| = k$ which maximizes $|E \cap S_0|$;
Set $T \leftarrow O(k(\varepsilon + \log n + \log(1/\delta)))$;
for $t = \{1, 2, \cdots, T\}$ do
    Choose an $e \in S_{t-1}$ uniformly at random, let $S'_t \leftarrow S_{t-1} \backslash \{e\}$;
    Choose an element $x$ in $[N] \backslash S'_t$ with probability $\propto \pi(S'_t \cup \{x\})$, let $S_t \leftarrow S'_t \cup \{x\}$;
end
Let $\widehat{E} \subset [N]$ be the edge set corresponding to $S_T$ ;
for $e \in \widehat{E}$ do
    Draw an independent Laplace noise $Z \sim \mathtt{Lap}(1/\varepsilon)$ ;
    $w_e \leftarrow \max\{0, w_e + Z\}$ ;
end
for $e' \in [N] \wedge e' \notin \widehat{E}$ do
    $w_{e'} \leftarrow 0$ ;
end
return $\widehat{G} = (V, \widehat{E})$.
```

D.1 Run time analysis of Algorithm 4

Here, we show that Algorithm 4 runs in almost linear time and linear space. First, in line 2 of Algorithm 4, if $k \geq |E|$, we then choose an arbitrary S_0 that covers E. Otherwise, we just choose an arbitrary subset of E with size k. Both cases can be implemented efficiently. Then, when implementing the basis-exchange walk, the run time basically follows the analysis on the run time of basis exchange walk in Appendix C. After obtaining the topology, since with high probability, $|\widehat{E}| = O(|E|)$. Then the loop in line 9 completes in time $O(|E|)$. Also, the loop in line 13 can be efficiently implemented with a proper data structure. Therefore, Algorithm 4 does run in $O(|E| \log(n/\delta) \log n)$ time and needs $O(|E|)$ space with high probability.

D.2 Analysis on the privacy guarantee of Algorithm 4

In this section, we prove the privacy guarantee of Algorithm 4.

Theorem D.1 *Given $\varepsilon, \delta > 0$, Algorithm 4 is $(4\varepsilon, \delta)$-differentially private.*

For this, we first show that for a given k, if the distribution of outputting a topology $S \in \binom{[n]}{k}$, denoted by π, satisfies

$$\pi(S) \propto \prod_{e \in S} \exp(\varepsilon \cdot w_e), \tag{7}$$

then π is $(2\varepsilon, 0)$-differentially private (Lemma D.2). Next, we show that the distribution π defined in Equation (7) is *strongly log-concave* (see Lemma C.2). This is our main technical insight and it ensures that the basis-exchange walk mixes rapidly and its output distribution after t iterations, denoted by π_t, is close to the target distribution π in total variation distance. Then, by a standard argument, this means that π_t is (ε, δ)-differentially private. Note that without a provable guarantee on mixing time, we cannot guarantee differential privacy. After obtaining a private topology distributed as π_t, we use the basic composition lemma to complete the proof. The following lemma can be shown using standard techniques.

Lemma D.2 *Given $k \geq 0$, If there is an algorithm $\mathcal{A}'$ outputs a topology $S \in \binom{[n]}{k}$ according to the distribution in Equation (7), then it is $(2\varepsilon, 0)$-differential privacy.*

Proof: [Proof of Lemma theorem D.2] Let G and G' be a pair of neighboring graphs with n vertices where there is an $i \leq \binom{n}{2}$ such that $|G[i] - G'[i]| \leq 1$. Since both graphs have n vertices, then the sampling space is the same. Furthermore, for any $S \in \binom{[n]}{k}$,

$$\begin{aligned}\frac{Pr[\mathcal{A}'(G) = S]}{Pr[\mathcal{A}'(G') = S]} &= \frac{\prod_{e \in S} \exp(\varepsilon \cdot G[e])}{\prod_{e \in S} \exp(\varepsilon \cdot G'[e])} \cdot \frac{\sum_{S' \in \binom{[N]}{k}} \prod_{e \in S'} \exp(\varepsilon \cdot G'[e])}{\sum_{S' \in \binom{[N]}{k}} \prod_{e \in S'} \exp(\varepsilon \cdot G[e])} \\ &\leq \exp(2\varepsilon |G[i] - G[i']|) \leq e^{2\varepsilon},\end{aligned}$$

which completes the proof. □

We now show that Algorithm 1 outputs a topology with distribution close to π (in terms of the total variation distance). To do this, we first show that the target distribution π is strongly log-concave on $\mathbb{R}^N_{\geq 0}$ (Theorem B.17) so that the corresponding basis-exchange walk enjoys rapid mixing. Applying Lemma C.2 directly implies the following:

Lemma D.3 *For any graph $G \in \mathbb{R}^N_{\geq 0}$ and $k \geq 0, \varepsilon > 0$, the generating polynomial of distribution π defined in Equation (7) is strongly log-concave at $z \in \mathbb{R}^N_{\geq 0}$.*

This allows us to say that the distribution of the output of the basis-exchange walk in Algorithm 1 is close to π under total variation distance. Fix a $T \in \mathbb{N}$, recall that S_T is the edge set chosen by Algorithm 1. Let $\varepsilon' = 2\varepsilon$, and π_T be the distribution of S_T. Then Lemma C.3 in Appendix C implies that there is a c such that if $T = c \cdot k(\varepsilon + \log n + \log(1/\delta))$, we have

$$\|\pi_T - \pi\|_{TV} \leq \frac{\delta}{e^{\varepsilon'} + 1}.$$

Finally, we use Lemma D.4 to compare two distributions and get an approximate DP result.

Lemma D.4 *Let μ_1, μ_2, μ_1' and μ_2' be distributions on Ω. If μ_1 and μ_2 are a pair of (ε', δ^*)-indistinguishable distributions and $d_{TV}(\mu_1, \mu_1') \leq \delta'$, $d_{TV}(\mu_2, \mu_2') \leq \delta'$, then μ_1' and μ_2' are $(\varepsilon', (e^{\varepsilon'} + 1)\delta' + \delta^*)$-indistinguishable.*

Proof: For any $S \subset \Omega$, we see that

$$\begin{aligned}Pr_{\mu_1'}(S) &\leq Pr_{\mu_1}(S) + \delta' \leq e^{\varepsilon'} \cdot Pr_{\mu_2}(S) + \delta^* + \delta' \\ &\leq e^{\varepsilon'} \cdot (Pr_{\mu_2'}(S) + \delta') + \delta^* + \delta' \\ &\leq e^{\varepsilon'} \cdot Pr_{\mu_2'}(S) + (e^{\varepsilon'} + 1)\delta' + \delta^*.\end{aligned}$$

The other side is symmetric. □

Lemma D.5 *Given $k \geq 0$, Algorithm 1 with $T = c \cdot k(\varepsilon + \log n + \log(1/\delta))$ outputs a topology S while preserving $(2\varepsilon, \delta)$-differential privacy.*

Proof: Combining Lemma D.2 and Lemma C.3, we see that for neighboring graphs, the conditions in Lemma D.4 are satisfied with $\delta' = \delta/(e^{\varepsilon'} + 1)$ and $\delta^* = 0$. Thus, by Lemma D.4, Algorithm 1 outputs a topology S while preserving $(2\varepsilon, \delta)$-differential privacy. □

D.3 Analysis on the utility guarantee of Algorithm 4

In this section, we mainly prove the following theorem on the utility of Algorithm 4:

Theorem D.6 *Given any weighted graph $G = (V, E)$, $\varepsilon > 0$ and $\delta, \beta \in (0, 1)$, then with probability at least $1 - 3\beta - \delta/(1 + e^{\varepsilon})$, Algorithm 4 outputs a synthetic graph $\widehat{G}$ with at most $|E| + \frac{2\log(1/\beta)}{\varepsilon}$ edges such that*

$$Eval(G, \widehat{G}) \le |E| \cdot \frac{3k \log(n/\beta)}{\varepsilon} + \frac{6 \log(n/\beta) \log(1/\beta)}{\varepsilon^2}.$$

Further, for spectral error we have $\left\| L_G - L_{\widehat{G}} \right\|_2 = O\left(\frac{d_{\max} \cdot \log(n)}{\varepsilon} + \frac{\log(1/\beta) \log^2 n}{n\varepsilon^2} \right)$.

We note that, in Theorem D.6, we do not assume the number of edges is publicly known as in Algorithm 4. Also, the utility guarantee of D.6 on spectral approximation basically follows the analysis in Liu et al. Liu et al. (2024), given that we have shown in Appendix D.2 that in terms of the total variation distance, the distribution of the topology sampled by Algorithm 4 is close to the distribution defined in Equation (7). Therefore, we only focus on the utility analysis on cut approximation.

Proof of Theorem 2.1. Combining analysis in Section D.2, Theorem D.6 and the discussions in Section D.1 directly yields Theorem 2.1, by simply rescaling the privacy budget ε by a constant.

In Theorem 2.1, the error bound becomes $Eval(G, \widehat{G}) \le O\left(\frac{|E| \log(n/\beta)}{\varepsilon} \right)$, this is because we do not have to pay an extra additive logarithmic term on perturbing the number of edges. To complete the proof of Theorem D.6, we use the following fact on tail inequalities of independent Laplace noise:

Fact D.7 *Given $k, b \ge 0$ and $k \in \mathbb{N}$. Let $\{Z_i\}_{i \in [k]}$ be k i.i.d. Laplace random variable from $\mathtt{Lap}(b)$. Then with probability at least $1 - e^{-t}$ for any $t > 0$, it holds that*

$$\max_{i \in [k]} |Z_i| \le (t + \log k) b.$$

Let

$$\mathcal{K}_{\text{good}} = \left\{ k \in \mathbb{N} : \ |E| \le k \le |E| + \frac{2\log(1/\beta)}{\varepsilon} \right\}.$$

Recall that $k = \min\left\{ N, |E| + \mathtt{Lap}(1/\varepsilon) + \frac{\log(1/\beta)}{\varepsilon} \right\}$, then from the tail inequality of Laplace distribution, we have that

$$Pr[k \in \mathcal{K}_{\text{good}}] \ge 1 - \beta.$$

Next, we consider the case where $k \in \mathcal{K}_{\text{good}}$. Given $G = ([n], E)$ and any topology $S \in \binom{[N]}{k}$ (recall that $N = \binom{n}{2}$), we write $G|S \in \mathbb{R}^N_{\ge 0}$ to denote the restriction of G on S. That is, $G|S$ is a graph with n vertices whose edge set is $S \cap E$, and $w_e(G|S) = w_e(G)$ for any $e \in S \cap E$. Next, we show that the topology S_T given by Algorithm 4 is good:

Lemma D.8 *In Algorithm 4 with $\delta \in (0, 1)$, given $k \in \mathcal{K}_{good}$, then with probability at least $1 - \beta - \delta/(1 + e^{\varepsilon})$, it holds that*

$$\|(G|S_T) - G\|_1 \le k \log(n/\beta).$$

Proof: Let $f(e) = \exp(\varepsilon \cdot w_e)$ for any $e \in [N]$. For any $t > 0$ and $k \in \mathcal{K}_{\text{good}}$, let $\mathcal{S}_t = \{S \subset [N] : |S| = k \wedge \|(G|S) - G\|_1 \ge t\}$. Recall that the distribution π defined on $\binom{[N]}{k}$ satisfies

$$\pi(S) \propto \prod_{e \in S} f(e) = \prod_{e \in S} \exp(\varepsilon \cdot w_e). \tag{8}$$

Therefore, let $S_{OPT} \in \binom{[N]}{k}$ be any topology that $E \in S_{OPT}$ (note that $k \ge |E|$ if $k \in \mathcal{K}_{\text{good}}$). Then we have that

$$\begin{aligned}
\pi(\mathcal{S}_t) \le \frac{\pi(\mathcal{S}_t)}{\pi(S_{OPT})} &\le \binom{N}{k} \cdot \frac{\max_{S \in \mathcal{S}_t} \prod_{e \in S} f(e)}{\prod_{e \in E} f(e)} \\
&= \binom{N}{k} \cdot \frac{\max_{S \in \mathcal{S}_t} [\exp(0 \cdot \varepsilon)]^{|S \setminus E|} \prod_{e \in S \cap E} f(e)}{\prod_{e \in E} f(e)} \\
&= \binom{N}{k} \cdot \left[\max_{S \in \mathcal{S}_t} \prod_{e \in E \setminus S} f^{-1}(e) \right] \\
&= \binom{N}{k} \exp(-\varepsilon \|(G|S) - G\|_1) \\
&\le \exp(k \log n - t) \le \beta
\end{aligned}$$

if we set $t = k\log(n/\beta)$. Let the distribution of S_T outputted by Algorithm 4 be π_T. By Lemma C.3, we have that $\|\pi - \pi_T\|_{TV} \le \frac{\delta}{e^\varepsilon+1}$. That is to say:

$$\pi_T(\mathcal{S}_t) \le \beta + \frac{\delta}{e^\varepsilon+1},$$

where $t = k\log(n/\beta)$. This completes the proof of Lemma D.8. □

Proof of Theorem D.6. By Fact D.7, we further have that with probability at least $1-\beta$, it holds that

$$\left\|(G|S_T) - \widehat{G}\right\|_1 \le k\frac{\log(n^2/\beta)}{\varepsilon} \le 2k\frac{\log(n/\beta)}{\varepsilon}.$$

Therefore, using the union bound, we have that with probability at least $1-2\beta$ for some small β, it holds that $\left\|(G|S_T) - \widehat{G}\right\|_1 \le 2k\log(n/\beta)/\varepsilon$ for some $k \le |E| + 2\log(1/\beta)/\varepsilon$. Therefore, we have that

$$\begin{aligned}
Eval(G,\widehat{G}) &= \max_{q\in[0,1]^N} |q^\top G - q^\top \widehat{G}| = q^\top|G-\widehat{G}| \\
&\le \sum_{i\in[N]} |G[i] - \widehat{G}[i]| = \|G-\widehat{G}\|_1 \\
&\le \|G - (G|S_T)\|_1 + \|(G|S_T) - \widehat{G}\|_1 \\
&\le k\cdot\left(\frac{\log(n/\beta)}{\varepsilon} + \frac{2\log(n/\beta)}{\varepsilon}\right) \\
&= |E|\cdot\frac{3k\log(n/\beta)}{\varepsilon} + \frac{6\log(n/\beta)\log(1/\beta)}{\varepsilon^2}
\end{aligned} \tag{9}$$

holds with probability at least $1-3\beta-\delta/(1+e^\varepsilon)$ due to the union bound.

E Missing proofs in Section 2.2

E.1 Proof of Theorem 2.2

Here, we first restate Theorem 2.2.

Theorem E.1 *For any $\varepsilon > 0$ and $\delta \in (0,1)$, Algorithm 2 preserves (ε,δ) differential privacy. Also, with probability at least $1-\delta$, Algorithm 2 outputs a $\widehat{G}$ such that*

$$Eval(G,\widehat{G}) = O\left(\frac{|E|\log(n/\delta)}{\varepsilon}\right).$$

Proof: We first show the privacy guarantee. Given any graph $G = ([n],E)$, we consider two types of neighboring graphs:

1. $G_1 = ([n],E_1)$ has an extra edge e' with weight 1 outside the edge set $|E|$, and
2. $G_2 = ([n],E_2)$ has edge set $E_2 \subset E$.

Firstly, by the tail inequality of Laplace distribution (Lemma D.7), we see that by setting $t = \frac{2\log(2n/\delta)}{\varepsilon}$, for every edge with weight less or equal than 1, it holds with probability at least $1-\frac{\delta}{n^2}$ that this edge will be filtered. Let the extra edge in G_1 be e', then by the union bound, with probability at least $1-\delta$, this edge will be set to 0. Suppose it happens, then Algorithm 2 preserves $(\varepsilon,0)$ differential privacy due to the Laplace mechanism and the post-processing property of differential privacy. For case 2 where G_2 has the same or smaller edge set, clearly Algorithm 2 also preserves (ε,δ) differential privacy. Thus, Algorithm 2 preserves (ε,δ) differential privacy for $\varepsilon > 0$ and $0 < \delta < 1$.

For the utility part, again by the tail inequality of Laplace distribution and the union bound, we see that the difference between G and $\widehat{G}$ in terms of ℓ_1 norm is

$$Pr\left[\left\|G-\widehat{G}\right\|_1 \le 2|E|\cdot\frac{2\log(2n/\delta)}{\varepsilon}\right] \ge 1-\delta,$$

and this means that

$$\begin{aligned} Eval(G,\widehat{G}) &= \max_{q\in[0,1]^N} |q^\top G - q^\top \widehat{G}| = q^\top |G-\widehat{G}| \\ &\le \sum_{i\in[N]} |G[i]-\widehat{G}[i]| = \|G-\widehat{G}\|_1 \\ &\le 4|E|\cdot\frac{\log(2n/\delta)}{\varepsilon} \end{aligned} \tag{10}$$

holds with probability at least $1-\delta$. This completes the proof of Theorem 2.2. □

In particular, if the unweighted degree is bounded by $d_{\mathsf{max}} \le n-1$, Algorithm 2 will have a more accurate approximation on the size of small cuts. For this property, we give the following theorem:

Theorem E.2 *Given $\varepsilon > 0$ and $0<\delta<1$. For any $G=([n],E)$ with maximum unweighted degree $d_{\mathsf{max}} \le n-1$, with probability at least $1-\delta$, Algorithm 2 outputs a $\widehat{G}$ such that for any $S\in 2^{[n]}$,*

$$|\Phi_G(S)-\Phi_{\widehat{G}}(S)| \le \frac{4d_{\mathsf{max}}|S|\log(2n/\delta)}{\varepsilon}.$$

Proof: In Algorithm 2, we first add a Laplace noise on each edge in the edge set, then filter the edges with weights below or equal to the threshold t. Thus, for any $e\in E$, we have that with probability at least $1-\delta$,

$$|w_e-\widehat{w}_e| \le 2t,$$

while for every $e\notin E$, $w_e=\widehat{w}_e=0$. Consider any $S\in 2^{[n]}$, since $E(S,[n]\backslash S)$ has at most $d_{\mathsf{max}}|S|$ edges, then we have that

$$|\Phi_G(S)-\Phi_{\widehat{G}}(S)| \le 2t\cdot d_{\mathsf{max}}|S| = \frac{4d_{\mathsf{max}}|S|\log(2n/\delta)}{\varepsilon}.$$

This concludes the proof of Theorem E.2. □

Note that, For any disjoint $S,T\in 2^{[n]}$, there are at most $d_{\mathsf{max}}\cdot\min(|S|,|T|)$ edges crossing through the (S,T)-cut. Thus, we immediately have the following corollary:

Corollary E.3 *Given $\varepsilon>0$ and $0<\delta<1$. For any $G=([n],E)$ with maximum unweighted degree $d_{\mathsf{max}}\le n-1$, with probability at least $1-\delta$, Algorithm 2 outputs a $\widehat{G}$ such that for any disjoint $S,T\in 2^{[n]}$,*

$$|\Phi_G(S,T)-\Phi_{\widehat{G}}(S,T)| = O\left(\frac{d_{\mathsf{max}}\cdot\min(|S|,|T|)\log(n/\delta)}{\varepsilon}\right).$$

E.2 Proof of Theorem 2.3

Here, we first restate Theorem 2.3.

Theorem E.4 *For any $G=([n],E)$ with bounded maximum unweighted degree $d_{\mathsf{max}}\le n-1$, with high probability, Algorithm 2 outputs a $\widehat{G}=(V,\widehat{E})$ such that for all $x\in\mathbb{R}^n$ with $\|x\|_2\le 1$,*

$$|x^\top L_G x - x^\top L_{\widehat{G}} x| = O\left(\frac{d_{\mathsf{max}}\log(n/\delta)}{\varepsilon}\right).$$

Proof: Recall that we set the threshold $t=\frac{c\log(n/\delta)}{\varepsilon}$ in Algorithm 2. By the analysis in Theorem 2.2, we have that

$$|w_e-\widehat{w}_e|\le 2t, \text{for all } e\in E$$

with probability at least $1-\delta$.

We use Lemma B.16 to convert the approximation on (S,T)-cuts to the approximation on the spectrum. Let $L_{\widetilde{G}} = L_G - L_{\widehat{G}}$. Clearly $L_{\widetilde{G}}\in\mathbb{R}^{n\times n}$ is a symmetric matrix such that $L_{\widetilde{G}}\mathbf{1}=\mathbf{0}$. We then check the conditions in Lemma B.16 for $L_{\widetilde{G}}$. First, with probability at least $1-\delta$, we have that

$$\max_{i\in[n]}|L_{\widetilde{G}}[i,i]| = \max_{i\in[n]}|\Phi_G(\{i\})-\Phi_{\widehat{G}}(\{i\})| \le 2t\cdot\min\{d_{\mathsf{max}},|E|\}\le 2t\cdot d_{\mathsf{max}}. \tag{11}$$

Since $\mathbf{1}_S[i]\mathbf{1}_T[j]$ and $\mathbf{1}_S[j]\mathbf{1}_T[i]$ cannot be 1 simultaneously, for all $\mathbf{1}_S, \mathbf{1}_T \in \{0,1\}^n$ such that $S \cap T = \varnothing$, we have

$$\begin{aligned}
\frac{|\mathbf{1}_S^\top L_{\widetilde{G}} \mathbf{1}_T|}{\|\mathbf{1}_S\|_2 \|\mathbf{1}_T\|_2} &= \frac{1}{\|\mathbf{1}_S\|_2 \|\mathbf{1}_T\|_2} \sum_{\{i,j\} \in E} w_{\{i,j\}} |\mathbf{1}_S[i]\mathbf{1}_T[i] + \mathbf{1}_S[j]\mathbf{1}_T[j] - \mathbf{1}_S[i]\mathbf{1}_T[j] - \mathbf{1}_S[j]\mathbf{1}_T[i]| \\
&= \frac{|\Phi_{\widetilde{G}}(S,T)|}{\|\mathbf{1}_S\|_2 \|\mathbf{1}_T\|_2} = \frac{|\Phi_G(S,T) - \Phi_{\widehat{G}}(S,T)|}{\|\mathbf{1}_S\|_2 \|\mathbf{1}_T\|_2} \\
&\leq \frac{1}{\|\mathbf{1}_S\|_2 \|\mathbf{1}_T\|_2} \cdot 2t \cdot \min\left\{ d_{\mathsf{max}} \cdot \min\left\{|S|, |T|\right\}, |E| \right\} \\
&= 2t \cdot \min\left\{ d_{\mathsf{max}} \cdot \min\left\{ \sqrt{\frac{|S|}{|T|}}, \sqrt{\frac{|T|}{|S|}} \right\}, \frac{|E|}{\sqrt{|S| \cdot |T|}} \right\} \leq 2t \cdot d_{\mathsf{max}}.
\end{aligned} \tag{12}$$

holds simultaneously.

We now let the α in Lemma B.16 be $\alpha = 2td_{\mathsf{max}}$. Combing the fact that $L_{\widetilde{G}}\mathbf{1} = \mathbf{0}$ and Equation (11) implies that $\|L_{\widetilde{G}}\|_\infty \leq 2\alpha$, and all diagonal entries of A has absolute value α which is less than $\alpha(\log(\ell/\alpha) + 1)$. Here, ℓ is the infinity norm of $L_{\widetilde{G}}$, and it is bounded by 2α. Also, Equation 12 implies that for all non-zero vectors $u, v \in \{0,1\}^n$ and $u^\top v = 0$, it holds that

$$\frac{|u^\top L_{\widetilde{G}} v|}{\|u\|_2 \|v\|_2} \leq \alpha.$$

Thus, Theorem 2.3 follows by applying Lemma B.16. with $\alpha = 2td_{\mathsf{max}} = O(d_{\mathsf{max}} \log(n/\delta)/\varepsilon)$. □

F Optimality of our algorithms

We have proposed our algorithms for approximating cut queries by actually preserving the answers to all linear queries on the given graph. In this section, we show that for answering any linear queries in $[0,1]^N$ by a synthetic graph $\widehat{G}$ under (ε, δ)-differential privacy, the error w.r.t. $G = (V, E)$ is at least $\Omega((1-\delta)|E|/(e^\varepsilon + 1))$ for $\varepsilon > 0$ and $0 < \delta < 1$. Thus, both our (ε, δ)-differentially private algorithms for answering linear queries are optimal.

Theorem F.1 *Recall that $N = \binom{n}{2}$. Let $\mathcal{M} : \mathbb{R}^N_{\geq 0} \to \mathbb{R}^N$ be a (ε, δ)-algorithm such that for any given graph G, it outputs a $\widehat{G}$ which satisfies*

$$\mathbb{E}[Eval(G, \widehat{G})] = \mathbb{E}\left[\max_{q \in [0,1]^N} |q(G) - (\widehat{G})|\right] \leq \alpha,$$

then $\alpha = \Omega\left(\frac{(1-\delta)|E|}{e^\varepsilon + 1}\right)$.

Proof: We first note that preserving the ℓ_1 norm of G can be reduced to answering any collection $\mathcal{Q}$ of linear queries in $[0,1]^N$. (It is easy to verify the opposite direction is also correct). Let $\widehat{G}$ be the output. Then we construct two linear queries

$$q_+ := \begin{cases} 1, & i \in [N] \text{ and } G[i] \geq \widehat{G}[i], \\ 0, & i \in [N] \text{ and } G[i] < \widehat{G}[i]; \end{cases}$$

and $q_- = \{1\}^N - q_+$, which flips the answer of q_+. Then we see that

$$\langle q_+, (G - \widehat{G}) \rangle + \langle q_-, (G - \widehat{G}) \rangle = \|G - \widehat{G}\|_1,$$

where $\langle \cdot, \cdot \rangle$ is the inner product of two vectors. Since $q_+, q_- \in [0,1]^N$, then

$$\begin{aligned}
Eval(G, \widehat{G}) &= \max_{q \in [0,1]^N} |q(G - \widehat{G})| \\
&\geq \max\{\langle q_+, (G - \widehat{G}) \rangle, \langle q_-, (G - \widehat{G}) \rangle\} \\
&\geq \|G - \widehat{G}\|_1 / 2.
\end{aligned} \tag{13}$$

Now we aim to find the lower bounds on minimizing $\|G - \widehat{G}\|_1$. Consider the dense case where $|E| = N$, there is a widely accepted sense in which the Laplace mechanism, i.e., adding a random perturbation on each pair of vertices achieves optimal error on preserving G in terms of ℓ_1 norm for mechanisms adding oblivious noise (see Aumuller et al. (2022) and Theorem 6 in Koufogiannis et al. (2015)). Since each magnitude of the i.i.d. Laplace noise added by the Laplace mechanism is $O(1/\varepsilon)$, then the error in terms of ℓ_1 norm is $O(N\log n/\varepsilon) = O(|E|\log n/\varepsilon)$. Here, the $\log n$ factor is due to the union bound. For general algorithms in the sparse case, we prove the following lemma for the sake of completeness:

Lemma F.2 *Given $\varepsilon > 0$ and $0 < \delta < 1$. Let $\mathcal{M} : \mathbb{R}^N_{\geq 0} \to \mathbb{R}^N$ be a (ε,δ)-algorithm such that for any given graph G, it outputs a $\widehat{G}$ which satisfies $\mathbb{E}\left[\|G-\widehat{G}\|_1\right] \leq \alpha$, then $\alpha = \Omega\left(\frac{(1-\delta)|E|}{e^\varepsilon+1}\right)$.*

Proof: We first consider the dense case where $|E| = N$, and the goal is to show that the error α is at least $\Omega(N)$. Let $\mathcal{M}_i : \mathbb{R}^N_{\geq 0} \to \mathbb{R}$ be the projection of $\mathcal{M}$ on the i-th position for $i \in [N]$, namely $\widehat{G}[i] = \mathcal{M}_i(G)$ if $\mathcal{M}(G) = \widehat{G}$. By the post-processing lemma, each $\mathcal{M}_i$ is simultaneously (ε,δ)-differential privacy. For any $G \in \mathbb{R}^N_{\geq 0}$, let $G^{(i)}$ for $i \in [N]$ be N different neighboring graphs of G such that $G^{(i)}[i] = G[i]+1$. Let $0 < \beta < 1-\delta$ be a parameter. Suppose that for any fixed $i \in [N]$, with probability at least $1-\beta$, the error $|G[i]-\widehat{G}[i]|$ is at most α_i, where α_i is any constant satisfies $0 < \alpha_i < 1/2$. Since $\mathcal{M}_i$ is (ε,δ)-differentially private, then

$$\begin{aligned}
\beta &\geq Pr[|\mathcal{M}_i(G^{(i)}) - G^{(i)}[i]| > \alpha_i] \\
&\geq Pr[\mathcal{M}_i(G^{(i)}) < G^{(i)}[i] - \alpha_i] \\
&\geq Pr[\mathcal{M}_i(G^{(i)}) < G^{(i)}[i] - 1/2] \\
&\geq Pr[\mathcal{M}_i(G^{(i)}) \leq G[i] + \alpha_i] \\
&\geq e^{-\varepsilon}\cdot(Pr[\mathcal{M}_i(G) \leq G[i]+\alpha_i] - \delta) \\
&\geq (1-\beta-\delta)\cdot e^{-\varepsilon}.
\end{aligned} \tag{14}$$

This inequality is violated for any $\beta < \frac{e^{-\varepsilon}(1-\delta)}{1+e^{-\varepsilon}}$. Therefore, for all $0 < \varepsilon, \delta < 1$, if we choose a β satisfying $0.99\cdot\frac{1-\delta}{e^\varepsilon+1} < \beta < \frac{e^{-\varepsilon}(1-\delta)}{1+e^{-\varepsilon}} < 1-\delta$, then we have that for any $i \in [N]$,

$$Pr\left[|G[i]-\widehat{G}[i]| > \alpha_i\right] > \beta > 0.99\cdot\frac{1-\delta}{e^\varepsilon+1}.$$

Then, the expected error on ℓ_1 norm will be

$$\mathbb{E}\left[\left\|G-\widehat{G}\right\|_1\right] \geq \sum_{i\in[N]}\alpha_i\cdot Pr\left[|G[i]-\widehat{G}[i]| > \alpha_i\right] \geq 0.99\cdot\sum_{i\in[N]}\frac{\alpha_i(1-\delta)}{e^\varepsilon+1},$$

which will be at least $\Omega\left(\frac{(1-\delta)N}{(e^\varepsilon+1)}\right)$.

We now move to the sparse case, i.e., $|E| = o(N)$. For the sake of contradiction, suppose there is an algorithm that outputs a graph, $\widehat{G}$, such that $\mathbb{E}\left[\|\widehat{G}-G\|_1\right] = o(|E|)$, then consider the density case with $N' = \Theta(|E|)$, we can always borrow some dimensions with zero weight to make it a sparse case with respect to N, where $N' = o(N)$. The above argument shows that the error for the dense case is at least $\Omega(N') = \Omega(|E|)$. Note that we construct the sparse case by borrowing new dimensions, and it will only increase the error in terms of ℓ_1 norm. That is, the error in terms of ℓ_1 norm in the sparse case w.r.t. N is also at least $\Omega(|E|)$, which leads to a contraction and concludes the proof of Lemma F.2. □ Combining Equation (13) and Lemma F.2 completes the proof of Theorem F.1. □

G Extension to Continual Observation

In previous sections, we mainly discuss static algorithms on confidential graphs. However, there is a long line of work on the dynamic setting known as continual observation model (Chan et al., 2011; Dwork et al., 2010a). Here, we introduce a general setting for private graph analysis under continual observation. Consider that we start from an empty graph $\widetilde{G}_0 = G_{\text{empty}} = ([n], \varnothing)$. Given a time upper bound T, a private algorithm $\mathcal{M}$ on graphs and a utility function $s : \mathbb{R}^{\binom{n}{2}}_{\geq 0} \times \mathbb{R}^{\binom{n}{2}}_{\geq 0} \to \mathbb{R}_{\geq 0}$, in each round $i = 1, \cdots, T$:

1. An oblivious adversary propose an update $G_i = (e, w) \in [\binom{n}{2}] \times \mathbb{R}_{\geq 0}$, and let $\widetilde{G}_i = \widetilde{G}_{i-1} + G_i$, which means increasing the weight of the e-th pair of vertices by w.

Algorithm 5: Binary mechanism (in the graph setting)

```
Input: A sequence of edges $\{(e_i, w_i)\}_{i\in[T]}$, a mechanism $\mathcal{M} : \mathbb{R}_{\geq 0}^{\binom{n}{2}} \to \mathbb{R}_{\geq 0}^{\binom{n}{2}}$.
Output: A sequence of synthetic graphs $\{\widehat{G}_i\}_{i\in[T]}$.
for $t = \{1, 2, \cdots, T\}$ do
    Express $t$ in its binary form $t = \sum_{l\geq 0} k_l(t) \cdot 2^l$;
    // Here, $k_l(t) \in \{0, 1\}$ is the $(l+1)$-th bit of $t$ (in binary form) starting from
       the right hand.
    Let $j \leftarrow \min_{l\geq 0}\{k_l(t) \neq 0\}$;
    Let $\alpha_j \leftarrow \sum_{l<j} \alpha_l + G_t$;
    for $0 \leq l \leq j - 1$ do
        $\alpha_i = G_{\text{empty}}$
    end
    Run $\mathcal{M}$ on $\alpha_j$, and set $\widehat{\alpha}_j = \mathcal{M}(\alpha_j)$ ;
    Release $\widehat{G}_t = \sum_{l\geq 0} \widehat{\alpha}_l$ ;
end
```

2. $\mathcal{M}$ outputs a synthetic graph that approximates $\widetilde{G}_i$ with respect to the utility function s.

To apply Definition 1.2 in continual observation, we view streams as the input datasets and two streams are neighboring if they differ in one update by at most 1 (on same edge e). This is also called the *event-level* privacy. We note that under such a privacy notion, two neighboring streams could still have different topology (consider two updates $(e, 0)$ and $(e, 1)$ where $e \leq \binom{n}{2}$).

The algorithm. We use the binary mechanism (Chan et al., 2011; Dwork et al., 2010a). A trivial approach is to initialize $\binom{n}{2}$ copies of binary mechanism on each pair of vertices, which needs $O(n^2)$ time per round. Our algorithm speeds up the procedure using the sparsity of partial sums of graphs. In Algorithm 5, we use G_t ($t \in [T]$) to denote the graph whose only edge is the t-th coming edge (with weight w_t) in the stream. We use G_{empty} to denote the empty graph.

Theorem G.1 *Given $T \in \mathbb{N}$, for any stream $S \in ([\binom{n}{2}] \times \mathbb{R}_{\geq 0})^T$ with m be the number of edges in the final state $\widetilde{G}_T$, let $\widetilde{G}_t$ denote the graph formed by the stream until time t. Then for any $\varepsilon > 0$ and $0 < \delta < 1$, Algorithm 5 is an $O(\log T)$ amortized update time, $(\varepsilon, (T + 1)\delta)$ differentially private algorithm under continual observation that at every time epoch, outputs a graph $\widehat{G}_t$, such that with probability at least $1 - T\delta$,*

$$\max_{t\in[T]} Eval\left(\widehat{G}_t, \widetilde{G}_t\right) = O\left(\frac{m \log(\frac{n}{\delta})\sqrt{\log^3(T)\log(\frac{1}{\delta})}}{\varepsilon}\right).$$

We first give the privacy guarantee of Algorithm 5, which is a direct application of the advanced composition lemma (see Lemma B.6).

Theorem G.2 *Given that $\mathcal{M} : \mathbb{R}_{\geq 0}^{\binom{n}{2}} \to \mathbb{R}_{\geq 0}^{\binom{n}{2}}$ is a (ε, δ) edge level differentially private algorithm, then Algorithm 5 (with $\mathcal{M}$ as input) preserves $(\varepsilon', \delta' + T\delta)$ edge level differential privacy for any $\delta' \in (0, 1)$, where*

$$\varepsilon' = O\left(\varepsilon \cdot \sqrt{\log(T)\log(1/\delta')}\right).$$

Proof: (Sketch.) Let $S = \{(e_i^S, w_i^S)\}_{i\in[T]}$ and $S' = \{(e_i^{S'}, w_i^{S'})\}_{i\in[T]}$ be a pair of neighboring streams of length T that only differs in one item by 1. Let t be the time such that $w_t^S \neq w_t^{S'}$ and $|w_t^S - w_t^{S'}| \leq 1$. Note that in Algorithm 5, the t-th edge can occur in at most $O(\log T)$ different partial sums. Since the distribution of the released partial sums formed by graphs that don't contain the t-th edge is identical between two neighboring inputs, then every output in time $i \in [T]$ can be considered as the outcome of post-processing over the released partial sums formed by graphs that contain the t-th edge. Therefore, Theorem G.2 can be proved by applying the advanced composition lemma. □

For the accuracy part, we show that if the utility function s is "sub-additive", then it's possible to have a good utility bound.

Definition G.3 (Sub-additive measure) *Let $s : \mathbb{R}_{\geq 0}^{\binom{n}{2}} \times \mathbb{R}_{\geq 0}^{\binom{n}{2}} \to \mathbb{R}_{\geq 0}^{\binom{n}{2}}$ be a symmetric measure. We say that s is sub-additive if for any $\ell \in \mathbb{N}_{\geq 0}$ and $G_1, \cdots, G_\ell, G_{\ell+1}, G_{2\ell} \in \mathbb{R}_{\geq 0}^{\binom{n}{2}}$, it holds that*

$$s\left(\sum_{i=1}^{\ell} G_i, \sum_{j=\ell+1}^{2l} G_j\right) \leq \sum_{i=1}^{l} s(G_i, G_{i+l}).$$

For any $t \in [T]$, let $\widetilde{G}_t = \sum_{i \in [t]} G_i$. For any $G \in \mathbb{R}_{\geq 0}^N$, we also define

$$[G] = \{G' : \forall i \in [N], G'[i] \leq G[i]\}.$$

We now discuss the utility of Algorithm 5. Note that in each round, the output is simply the sum of synthetic graphs given by the private algorithm $\mathcal{M}$. For $t \in [T]$ and $0 \leq i \leq \lceil \log T \rceil$, let $\alpha_i^{(t)}$ be the graph represented by the partial sum α_i in Algorithm 5 at round t (in a same way, we define $\widehat{\alpha}_i^{(t)}$ be the perturbed partial sum). Suppose the measurement s that we are interested in is sub-additive. Since there are at most $\lceil \log T \rceil$ such partial sums (with corresponding synthetic graphs), then the error on any round $t \in [T]$ is

$$\begin{aligned} s\left(\widehat{G}_t, \widetilde{G}_t\right) &= s\left(\sum_{i=0}^{\lceil \log T \rceil} \widehat{\alpha}_i^{(t)}, \sum_{i=0}^{\lceil \log T \rceil} \alpha_i^{(t)}\right) \\ &\leq \log T \left(\max_{G \in [\widetilde{G}_T]} \mathtt{err}(G)\right). \end{aligned}$$

Thus, we directly give the following theorem on utility:

Theorem G.4 *Let $s : \mathbb{R}_{\geq 0}^{\binom{n}{2}} \times \mathbb{R}_{\geq 0}^{\binom{n}{2}} \to \mathbb{R}$ be any sub-additive measure. If $\mathcal{M} : \mathbb{R}_{\geq 0}^{\binom{n}{2}} \to \mathbb{R}_{\geq 0}^{\binom{n}{2}}$ is a (ε, δ) edge level differentially private algorithm and for any G, with probability at least $1 - \beta$, it outputs $\widehat{G}$ such that $s(G, \widehat{G}) \leq err(G)$, then if Algorithm 5 outputs $\widehat{G}_1, \cdots, \widehat{G}_T$, it holds with probability at least $1 - T\beta$ such that*

$$\max_{t \in [T]} s\left(\widehat{G}_t, \widetilde{G}_t\right) \leq \log T \left(\max_{G \in [\widetilde{G}_T]} err(G)\right).$$

Remark G.5 *In Theorem G.4, we see that we need to set $\beta = o(1/T)$ so that we can get a high probability bound.*

Now, we substitute the static algorithm $\mathcal{M}$ by Algorithm 2 and let s be $s(\widehat{G}_t, \widetilde{G}_t) = Eval(\widehat{G}_t, \widetilde{G}_t)$ for any t. By specifying parameters, we have the following corollary, by the fact that $Eval(\cdot, \cdot)$ is obviously sub-additive.

Corollary G.6 *Given $T \in \mathbb{N}$, $\varepsilon > 0$ and $0 < \delta < 1$, for any stream $S \in ([\binom{n}{2}] \times \mathbb{R}_{\geq 0})^T$ with m be the number of edges in the final state $\widetilde{G}_T$, there exists an $O(\log(T))$ amortized run-time $(\varepsilon, (T+1)\delta)$ differentially private algorithm under continual observation such that with probability at least $1 - \delta T$, it holds that:*

$$\max_{t \in [T]} Eval\left(\widehat{G}_t, \widetilde{G}_t\right) \leq O\left((\log T)^{1.5} \cdot \left(\frac{m \log(n/\delta)\sqrt{\log(1/\delta)}}{\varepsilon}\right)\right).$$

Proof: In each resample procedure, we call the static algorithm $\mathcal{M}$ (Algorithm 2) with parameter

$$\varepsilon_0 = \frac{\varepsilon}{\sqrt{\log T \log(1/\delta)}} \text{ and } \delta_0 = \delta.$$

By Theorem G.2, we see that Algorithm 5 preserves $(\varepsilon, (T+1)\delta)$-differential privacy under a stream of length T. By the utility guarantee of Algorithm 2 (Theorem 2.2), we have that

$$
\begin{aligned}
\max_{t\in[T]} Eval\left(\widehat{G}_t, \widetilde{G}_t\right) &\leq \log T \left(\max_{G\in[\widetilde{G}_T]} \mathtt{err}(G)\right) \\
&= \log T \cdot \left(\frac{m\log(n/\delta_0)}{\varepsilon_0}\right) \\
&= O\left((\log T)^{1.5} \cdot \left(\frac{m\log(n/\delta)\sqrt{\log(1/\delta)}}{\varepsilon}\right)\right)
\end{aligned}
$$

holds with probability at least $1 - T\delta$ due to the union bound (recall that Algorithms 2 succeeds with probability at least $1-\delta$).

For the time complexity, let's assume $T = 2^a$ for some $a \in \mathbb{N}$ for simplicity. In this case, we have to compute $T = 2^a$ synthetic graphs, and there are a layers in the binary mechanism (Algorithm 5). In the i-th layer (from bottom to top), each partial sum contains at most 2^i edges, but there are $T/2^i$ nodes. By the time guarantee of Algorithm 2, for an m edges graph, the algorithm needs $O(m)$ time even if m is not publicly known. Thus, to compute all synthetic graphs corresponding to the nodes in all layers, the time needed is

$$\sum_{i=0}^{a} O(2^i) \cdot \frac{T}{2^i} = O(T\log_2 T).$$

Since there are T rounds in total, then the average run-time for each round is $O(\log T)$. Note that the case where $\log_2 T \notin \mathbb{N}$ can be reduced to $T' = 2^{\lceil \log_2 T\rceil} \leq 2T$, which completes the proof. □

H A detailed comparison with other approaches

In this section, we briefly summarize previous techniques on differentially private cut approximation, and make a detailed comparison between the utility (in addition to the resource required) from those techniques and ours (see Table 7). In Table 7, $W = \frac{\|G\|_1}{\|G\|_0}$ is the average weight.

Table 7: The comparison of existing results (for constant ε).

Method	**Additive error for all $(S, V\backslash S)$ cuts**	**Efficient?**	**Purely additive?**	**Preserve sparsity?**
Dwork et al. (2006b)	$O\left(2^{n/2}\right)$	No	Yes	No
Hardt and Rothblum (2010)	$O\left(n\sqrt{\lvert E\rvert W} \cdot \mathtt{poly}(\log n)\right)$	No	Yes	No
Gupta et al. (2012)	$O\left(\sqrt{n\lvert E\rvert W} \cdot \mathtt{poly}(\log n)\right)$	No	Yes	No
Blocki et al. (2012)	$O\left(n^{1.5} \cdot \mathtt{poly}(\log n)\right)$	Yes	No	No
McSherry and Talwar (2007)	$O(n \cdot \mathtt{poly}(\log n))$	No	No	Yes
Warner (1965)	$O(n^{1.5} \cdot \mathtt{poly}(\log n))$	Yes	Yes	No
Upadhyay et al. (2021)	$O(n^{1.5} \cdot \mathtt{poly}(\log n))$	Yes	Yes	No
Eliáš et al. (2020)	$O(\sqrt{n\lvert E\rvert W}\log^2 n)$	Yes	Yes	No
Liu et al. (2024)	$O(\sqrt{n\lvert E\rvert}\log^3 n)$	Yes	Yes	No
(Ours)	$O(\lvert E\rvert \log n)$	Yes	Yes	Yes

H.1 Online algorithms

Much as our task is to output a synthetic graph that preserves cut approximation, a group of algorithms that we are also interested in answering all cut queries in the interactive setting, in which the algorithm works like a stateful API that handles each cut query. Compared to the offline setting (i.e., output a synthetic graph), the interactive setting is usually considered less demanding (see, Gupta et al. (2012); Ganev et al. (2022)). A naive approach in such an online setting is to add an independent Laplace noise on the answer to each cut query. Recall that under the edge level differential privacy, the sensitivity of cut queries is 1. Therefore, by the advanced composition lemma of differential privacy in Dwork and Roth (2014), to answer k cut queries, we need to add a noise $Z \sim \mathtt{Lap}(\sqrt{k}/\varepsilon)$ to preserve (ε, δ)-differential privacy, where δ is negligible. Since there are at most $O(2^n)$ different $(S, V\backslash S)$-cut queries, then the Laplace mechanism has noise proportional to $2^{n/2}/\varepsilon$.

An important improvement in the online analysis of private datasets is the private multiplicative weights method (see Hardt and Rothblum (2010)). In Gupta et al. (2012), the authors use a multiplicative weights algorithm (as a component of the iterative database construction approach) to answer linear queries normalized in the range $[0, 1]$ as we have defined before. Given parameter ε, this approach gives an $O(\sqrt{n|E|/\varepsilon})$ bound on the purely additive error on answering $(S, V\backslash S)$-cut queries if the input graph is unweighted. However, it's easy to verify that their algorithm can be easily extended to an $O(\sqrt{n|E|W/\varepsilon})$ upper bound for weight graphs whose average weight is W. This is because the error bound relies on the ℓ_1 norm the vector representation of the input graph, namely the sum of weights.

We note that the multiplicative weights approach answers each cut query efficiently, but it cannot reconstruct a synthetic graph efficiently, since there are exponentially many (S, T)-cut queries. In our algorithm, we construct a new graph in almost linear time. Note that, our algorithms can also answer linear queries with a purely additive error, which implies that our algorithms can approximate both $(S, V\backslash S)$-cut queries and (S, T)-cut queries with a purely additive error like the multiplicative weights approach. Moreover, for sparse weighted graphs where $|E| = O(n)$, our algorithm outperforms the online algorithms. However, for dense graphs where $|E| = \omega(n)$, the utility bound is not directly comparable.

H.2 Multiplicative Gaussian

In Blocki et al. (2012), the authors use the Johnson-Lindenstrauss transformation on the square root of the Laplacian matrix of graphs to preserve differential privacy. Formally speaking, fix an integer r, they pick a random Gaussian matrix $M \in \mathbb{R}^{r\times\binom{n}{2}}$ whose entries are independently drawn from $\mathcal{N}(0, 1)$, and output a private sketch $\widehat{L}_G = \frac{1}{r}E_G^\top M^\top M E_G$, where $E_G \in \mathbb{R}^{\binom{n}{2}\times n}$ is the edge adjacency matrix of the input graph. In the worst case, the multiplicative Gaussian method gives $\widetilde{O}(n^{1.5})$ additive error, with a constant multiplicative error (caused by the Johnson-Lindenstrauss transformation).

Compared to our algorithm, the private sketch $\widehat{L}_G$ produced by the multiplicative Gaussian method isn't necessarily a Laplacian matrix of any graph. Moreover, when $|E| = o(n^{1.5})$, the multiplicative Gaussian method pays more additive error.

H.3 Additive Gaussian/random flipping

Instead of multiplying a random Gaussian matrix, a much more direct method is to overlay a random Gaussian graph on the original graph G. That is, adding a Gaussian noise drawn from $\mathcal{N}(0, O(\log(1/\delta)/\varepsilon^2))$ on each pair of vertices independently, and releasing the new graph as $\widehat{G}$. By the Gaussian mechanism (Dwork and Roth, 2014), we see that it preserves (ε, δ) differential privacy. For the utility part, since each cut contains at most $O(n^2)$ Gaussian noise, by the property of sub-Gaussian, one can verify that for each cut S, $|\Phi_G(S) - \Phi_{\widehat{G}}(S)| \leq \widetilde{O}(n/\varepsilon)$ with high probability. Using the union bound over all 2^n different $(S, V\backslash S)$-cuts, the total error becomes $\max_{S\subset V} |\Phi_G(S) - \Phi_{\widehat{G}}(S)| \leq \widetilde{O}(n^{1.5}/\varepsilon)$ with high probability. Again, for sparse graphs when $|E| = o(n^{1.5})$, the additive Gaussian mechanism performs worse than our algorithm.

Another noise-adding strategy is to use the randomized response. In particular, for each pair of vertices $\{u, v\}$ of an unweighted graph, we choose its weight to be 1 or -1 independently, with probability $Pr[w'_{\{u,v\}} = 1] = \frac{1+\varepsilon w_{\{u,v\}}}{2}$ and $Pr[w'_{\{u,v\}} = -1] = \frac{1-\varepsilon w_{\{u,v\}}}{2}$. It is easy to verify that randomized response gives an $\widetilde{O}(n^{1.5}/\varepsilon)$ purely additive error, which is the same as the additive Gaussian method (see Blocki et al. (2012) for the calculation). The main flaw of the randomized response mechanism is that it only works on unweighted graphs.

H.4 Exponential mechanism

A well-known fact in graph theory is that each weighted graph has a cut-sparsifier that approximates the sizes of all $(S, V\backslash S)$-cuts (Benczúr and Karger, 1996). A graph G' is an α cut-sparsifier for G if it holds that for any $S \subseteq V$, $(1-\alpha)\Phi_G(S) \leq \Phi_{G'}(S) \leq (1+\alpha)\Phi_G(S)$. There have been a lot of works that have shown that each weighted graph with non-negative weights has an α edge-sparsifier with at most $O(n/\alpha^2)$ edges (Allen-Zhu et al., 2015; Batson et al., 2012; Lee and Sun, 2015). If the maximum weight is polynomial in n, we can always describe any such sparse edge-sparsifier in at most $\widetilde{O}(n)$ bits, and thus there are at most $2^{\widetilde{O}(n)}$ such sparsifiers. Therefore, we can consider all edge sparsifiers to be candidates, and as suggested in (Blocki et al., 2012), for any input graph G and sparsifier H, we use

$$q(G,H) = \max_{S}\{\min_{\eta:|\eta-1|\leq\alpha} |\Phi_H(S) - \Phi_G(S)|\}$$

as the scoring function. Clearly, such a scoring function has sensitivity 1 under edge level differential privacy; therefore, is $(\varepsilon, 0)$-differentially private. Further, using Lemma B.11, it is easy to verify that this instantiation of the exponential mechanism results in an additive error of $\widetilde{O}(n)$ with a constant multiplicative error, the latter due to sparsification. However, it is not clear how to sample from this instantiation of the exponential mechanism; while the score function is convex, the space is not convex, and therefore, algorithms such as the ones that give mixing time of sampling from log-concave distribution (Bassily et al., 2014) do not apply. Furthermore, in the exponential mechanism, not only do we need to sample in a space of exponential size, but computing the scoring function for a specified H is also hard.

H.5 Mirror descent

The mirror descent methods optimize a convex function $f(x)$ over a convex domain. Since Liu et al. (2024) uses the mirror-descent based algorithm of Eliáš et al. (2020), in what follows, we just discuss Eliáš et al. (2020). In Eliáš et al. (2020), for any input weighted graph G and candidate G', the authors use the cut norm Alon and Naor (2004):

$$d_{cut}(G,G') = \max\{|x^\top(A - A')y|; x, y \in \{0,1\}^n\}$$

as the loss function over the space of a semi-definite cone of Laplacian matrices. They then use private mirror descent to optimize G'. If we use the entropy mirror map, i.e., $\Phi(x) = \sum_{e\in\binom{V}{2}} x_e \log x_e$, it can be shown that private mirror descent gives $O(\sqrt{n|E|W/\varepsilon})$ purely additive error in expectation, which is same as the result of multiplicative weights approach (using entropy regularizer). However, the mirror descent approach outputs a synthetic graph in polynomial time, while the multiplicative weights method doesn't. Thus, this algorithm can be seen as an efficient implementation of *private multiplicative weight* (PMW) algorithm.

Compared to the mirror descent approach, we do not need any dependency on the maximum weight $W = \|G\|_\infty$. However, it's still an interesting question whether the square root dependency on W in mirror descent approaches is a fundamental barrier. Recall the scale invariance property (Remark 1.6), the dependency on W in Eliáš et al. (2020) seems necessary due to the trade-off between ε and edge weights. Also, in Lemma 4.3 of Eliáš et al. (2020), the authors show that the error between the cut norm and its convex approximation has relevance to the sum of weights, and hence W also comes into play. Another important property of our algorithm is that it preserves the sparsity. That is, if we are promised that the input graph is sparse, then the output graph is also sparse. While in Eliáš et al. (2020), the mirror descent method always outputs a weighted complete graph.

Remark H.1 (Discussion regarding high probability bound.) *The bound in Eliáš et al. (2020) is in expectation and they refer to Nemirovski et al. (2009) to get a high probability bound. Nemirovski et al. (2009) presents two general ways to get a high probability bound from the expectation. The stronger condition (stated as eq. (2.50) in Nemirovski et al. (2009) and stated below) that allows a bound that gives* $1-\beta$ *while incurring an extra* $\log(1/\beta)$ *factor is achieved if*

$$\mathbb{E}\left[\exp\left(\frac{\|g\|_\infty^2}{M^2}\right)\right] = \int n^2 \cdot exp(-t) \cdot exp(t^2/M^2)$$

is bounded. Here $\|g\|_\infty$ *is the* ℓ_∞ *norm of gradients in the mirror descent. However, the algorithm in Eliáš et al. (2020) does not satisfy this stronger condition. In fact, the expectation is unbounded if* M *is poly-logarithmic in* n. *Eliáš et al. (2020) satisfies the weaker condition (stated as eq. (2.40) in Nemirovski et al. (2009)). In fact, Lemma 4.5 in Eliáš et al. (2020) implies that* $\mathbb{E}[\|g\|_\infty^2] = O(log^2 n)$. *This condition though translates an expectation bound to* $1-\beta$ *probability bound with an extra* $\frac{1}{\beta^2}$ *factor in the error. In particular, if we want* $\beta = o(1/\sqrt{n})$, *then Eliáš et al. (2020) results in an error* $O(\sqrt{mn^3/\varepsilon}\log^2(n/\delta))$. *This is worse than our bound irrespective whether the graph is weighted or unweighted or is sparse or dense.*

I Some Real World Examples

In this section, we enumerate some of the other real-world examples of graphs that are sparse and heavily weighted, and there are natural reasons to study cut queries. This list is by no means exhaustive, but it serves to show that, in practice, one almost always encounters sparse highly weighted very large graphs in terms of nodes, making $\widetilde{O}(n^7)$ time algorithm infeasible in practice.

Three of the main applications of graph analysis are in understanding social network graphs, financial transactions, and internet traffic. In what follows, we give representative examples of each of these settings and supplement our claims that these examples result in sparse weighted graphs with publicly available information.

Social network graph There are four major social network graphs in the current age: three of them are subsidiaries to Facebook: Instagram, WhatsApp, and messenger, and one to Apple: iMessage. For each of these, there is a natural way to construct graphs.

- WhatsApp is an undirected weighted graphs, where the nodes are users and there is an edge if two people interact using WhatsApp with the edge weight representing the number of messages exchanged between them. According to reports, in the year 2021, there were approximately 2 billion users and about 36 trillion total messages (or the sum of weights on the edges) were sent only in 2021.
- Instagram is a directed graph, where a user corresponds to an account and there is a directed edge if a user interacts (in terms of likes on the post or comments) with the other user. In 2022, there were about 2.35 billion (active or inactive) Instagram accounts and just on a single personal (Christiano Ronaldo), there are more than 90 billion likes.
- Facebook messenger is another undirected graph with about 3 billion users in 2022 and a total of about 5 trillion messages exchanged just in the year 2022.
- Finally iMessage has currently 1.3 billion users and total 18.2 trillion messages were exchanged in 2021.

In all these cases, it is easy to verify that the degree of any node (that is other users any user interacts with) is significantly less. Here, the natural notion of privacy is whether, at a given time, two users interact or not. This corresponds to edge-level differential privacy. Moreover, in social network, like Instagram or Messenger, unless an account is private, one can always infer who is connected to whom. Moreover, the companies holding these data already know the connection; only the messages are end-to-end encrypted.

Financial graphs We use the example of Chase Bank and transactions on its ATM. According to reports, there are approximately 18 million accounts in Chase Bank and $16,000$ ATMs, while the total number of ATM transactions done in 2021 is more than 600 million. To the best of our knowledge, Chase Bank has never published the total number of transactions between its accounts or transactions done through other methods, like Zelle or Venmo.

Internet Activity Graph Currently, there are about 4.3 billion active IP addresses. However, just one search engine, Google, there are about $90,000$ web-search per second. The internet is a sparse graph because the number of connections between nodes (websites, servers, etc.) is much smaller than the total number of possible connections.

J An improved upper bound for fixed topologies

In this section, we relax edge-level differential privacy to a slightly weaker notion and prove a better utility bound under this setting. In short, we assume the topology (i.e., the edge set) is already public. This setting aligns with practical scenarios where data analysts already know the connections, one of the use cases that has been elaborated in Appendix I. Two graphs with the same edge set are neighboring if the edge weight w_e differs by 1 for exactly one $e \in E$. This assumption has also been well studied in the literature on private graph analysis (see Sealfon (2016); Roth et al. (2021); Fan and Li (2022) for example). In this setting, we just assume $|E| = \Omega(n)$ without the loss of generality. Otherwise, we can always remove the isolated vertices since the topology is known.

Given that we don't have to achieve privacy on topology, we just need to add independent Laplace noise on each edge in E:

1. For any $e \in E$, draw an independent Laplace noise $Z \sim \texttt{Lap}(1/\varepsilon)$;
2. Let $w_e \leftarrow w_e + Z$.

The application of the Laplace mechanism yields the following theorem for the utility of private cut approximation:

Theorem J.1 *Given $\varepsilon > 0$, there is a $(\varepsilon, 0)$-differentially private algorithm (under the assumption that the topology is public) such that for any input weighted graph $G = (V, E)$, with high probability, it outputs a $\widehat{G} = (V, \widehat{E})$ such that for any (S, T)-cut where $S, T \subset V$ and $S \cap T = \varnothing$, $\widehat{G}$ satisfies*

$$|\Phi_G(S,T) - \Phi_{\widehat{G}}(S,T)| \le \min\left\{ O\left(n\log n \cdot \sqrt{\min\{|S|,|T|\}}\right), O\left(\sqrt{n|E|\log^2 n}\right)\right\}.$$

Remark J.2 *We note that under the assumption where the topology is public, the second bound of Theorem J.1 entirely removes the dependency on the maximum edge weight W, compared to the $O(\sqrt{n|E|W})$ upper bound proposed in Eliáš et al. (2020). Here, we pay an extra $\log n$ factor in both bounds due to the union bound, while Eliáš et al. (2020) only guarantees that the expected accuracy is $O(\sqrt{n|E|W})$.*

Clearly, the algorithm given in this section is $(\varepsilon, 0)$-differentially private (see Lemma B.9). We prove the utility part of this theorem by the tail inequality of independent combinations of Laplace noise, which can be proved by standard techniques:

Lemma J.3 (Kotz et al. (2001); Gupta et al. (2012)) *Given $k \in \mathbb{N}_{>0}$ and $b > 0$. Let $\{Z_i\}_{i\in[k]}$ be i.i.d random variables such that $Z_i \in \texttt{Lap}(b)$, then for any $q_1, \cdots q_k \in [0,1]$,*

$$Pr\left[\sum_{i\in[k]} q_i Z_i > \alpha\right] \le \begin{cases} \exp\left(-\frac{\alpha^2}{6kb^2}\right), & \alpha \le kb, \\ \exp\left(-\frac{\alpha}{6kb}\right), & \alpha > kb. \end{cases}$$

Now, we are ready to prove Theorem J.1.

Proof: [Proof of Theorem J.1]

We first give the bound $O\left(n\log n\sqrt{\min\{|S|,|T|\}}\right)$. For any set of vertices S, let $E(S,T)$ be the collection of edges in the given graph that crosses S and T. Given G and any disjoint $S, T \in 2^V$ such that $|S| \le |T|$, the error in the synthetic graph can be written as $\texttt{err}(S,T) = \sum_{e\in E(S,T)} Z_e$, where $\{Z_e\}_{e\in|E|}$ are i.i.d Laplace noise from $\texttt{Lap}(1/\varepsilon)$.

Let $\alpha(S) = \min\{n|S| \cdot \lceil \log n\rceil, \binom{n}{2}\}$. One can further find a set of $\alpha(S)$ edges in a complete graph that includes $E(S,T)$, let that set be $\tau(S)$. Then, the error can be re-written as

$$\texttt{err}(S,T) = \sum_{e\in\tau(S)} q_e Z_e, \quad \text{where} \quad q_e = \begin{cases} 1 & e \in \tau(S)\cap E(S,T) \\ 0 & \text{otherwise} \end{cases}.$$

By Lemma J.3, we see that

$$Pr\left[\sum_{e\in\tau(S)} q_e Z_e > \frac{\sqrt{n\log n|\tau(S)|}}{\varepsilon}\right] \le \exp\left(-\frac{n\log n|\tau(S)|/\varepsilon^2}{6|\tau(S)|/\varepsilon^2}\right) = \exp\left(-\frac{n\log n}{6}\right) \tag{15}$$

since $\frac{\sqrt{n\log n|\tau(S)|}}{\varepsilon} \le \frac{1}{\varepsilon}\min\left\{n\log n\sqrt{|S|}, O(n^{1.5}\sqrt{\log n})\right\} \le \left(\frac{|\tau(S)|}{\varepsilon}\right)$. Since there are at most 2^{2n} different (S,T)-cuts, then we see that

$$\begin{aligned} &Pr\left[\exists \text{ disjoint } S,T \in 2^V \ \texttt{s.t.}|S| \le |T| \wedge \texttt{err}(S,T) > n\log n\sqrt{|S|}\right] \\ &\le Pr\left[\exists \text{ disjoint } S,T \in 2^V \ \texttt{s.t.}|S| \le |T| \wedge \texttt{err}(S,T) > \sqrt{n\log n|\tau(S)|}\right] \\ &\le \exp\left(-\Theta(\log n)\right) = O\left(\frac{1}{\texttt{poly}(n)}\right). \end{aligned} \tag{16}$$

The case where $|S| \ge |T|$ is symmetric. Thus, we have that with probability at least $1 - O(1/n^c)$, the error is at most $O\left(n\log n\sqrt{\min\{|S|,|T|\}}\right)$.

The argument for the second bound is almost the same, in which we just replace the size of $\tau'(S)$ by $\min\{|E| \cdot \lceil \log n \rceil, \binom{n}{2}\}$. This is because that $|E| \geq |E(S,T)|$, thus we can still find such $\tau'(S)$ that includes $E(S,T)$. Then, we have that

$$\frac{\sqrt{n \log n \cdot |\tau'(S)|}}{\varepsilon} \leq \frac{1}{\varepsilon} \min\{\log n \cdot \sqrt{n|E|}, O(n^{1.5}\sqrt{\log n})\} \leq \binom{|\tau'(S)|}{\varepsilon}$$

since we have assumed that $|E| = \Omega(n)$. Similarly, we have that

$$\begin{aligned} &Pr\left[\exists \text{ disjoint } S, T \subseteq V \ \ \texttt{s.t.} |S| \leq |T| \wedge \texttt{err}(S,T) > \log n \sqrt{n|E|}\right] \\ &\leq Pr\left[\exists \text{ disjoint } S, T \subseteq V \ \ \texttt{s.t.} |S| \leq |T| \wedge \texttt{err}(S,T) > \sqrt{n \log n |\tau'(S)|}\right] \\ &\leq \exp\left(-\Theta(\log n)\right) = O\left(\frac{1}{\texttt{poly}(n)}\right), \end{aligned} \tag{17}$$

which is what is claimed in Theorem J.1. □