Differentially Private Algorithms for Graphs Under Continual Observation

Hendrik Fichtenberger
Universität Wien, Austria

Monika Henzinger
Universität Wien, Austria

Wolfgang Ost
Universität Wien, Austria

Abstract

Differentially private algorithms protect individuals in data analysis scenarios by ensuring that there is only a weak correlation between the existence of the user in the data and the result of the analysis. Dynamic graph algorithms maintain the solution to a problem (e.g., a matching) on an evolving input, i.e., a graph where nodes or edges are inserted or deleted over time. They output the value of the solution after each update operation, i.e., continuously. We study (event-level and user-level) differentially private algorithms for graph problems under continual observation, i.e., differentially private dynamic graph algorithms. We present event-level private algorithms for partially dynamic counting-based problems such as triangle count that improve the additive error by a polynomial factor (in the length T of the update sequence) on the state of the art, resulting in the first algorithms with additive error polylogarithmic in T.

We also give ε-differentially private and partially dynamic algorithms for minimum spanning tree, minimum cut, densest subgraph, and maximum matching. The additive error of our improved MST algorithm is $O(W \log^{3/2} T/\varepsilon)$, where W is the maximum weight of any edge, which, as we show, is tight up to a $(\sqrt{\log T}/\varepsilon)$-factor. For the other problems, we present a partially-dynamic algorithm with multiplicative error $(1+\beta)$ for any constant $\beta > 0$ and additive error $O(W \log(nW) \log(T)/(\varepsilon\beta))$. Finally, we show that the additive error for a broad class of dynamic graph algorithms with user-level privacy must be linear in the value of the output solution's range.

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms; Security and privacy → Pseudonymity, anonymity and untraceability

Keywords and phrases differential privacy, continual observation, dynamic graph algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.42

Related Version *Full Version*: `https://arxiv.org/abs/2106.14756`

Funding Research supported by the Austrian Science Fund (FWF) and netIDEE SCIENCE project P 33775-N. Wolfgang Ost gratefully acknowledges the financial support from the Vienna Graduate School on Computational Optimization which is funded by the Austrian Science Fund (FWF project no. W1260-N35).

1 Introduction

Differential privacy aims to protect individuals whose data becomes part of an increasing number of data sets and is subject to analysis. A differentially private algorithm guarantees that its output depends only very little on an individual's contribution to the input data. Roughly speaking, an algorithm is ϵ-*differentially private* if the probability that it outputs O on data set D is at most an e^{ϵ}-factor of the probability that it outputs O on any adjacent data set D'. Two data sets are *adjacent* if they differ only in the data of a single user. Differential privacy was introduced in the setting of databases [9, 11], where users (entities) are typically represented by rows and data is recorded in columns. An important notion

that allowed for the development of generic techniques and tools (like the Laplace and the exponential mechanism) is the *sensitivity* of a function f: the static sensitivity ρ of f is the maximum $|f(D) - f(D')|$ over all adjacent pairs D, D'. Differential privacy was later generalized to a more challenging setting, where data evolves over time [12, 6]: a differentially private algorithm under *continual observation* must provide the same privacy guarantees as before, but for a sequence (or stream) of data sets instead of just a single data set. Often, this sequence results from updates to the original data set that arrive over time. In this setting, the presence or absence of a single user in one update can affect the algorithm's output on all future data sets, i.e., two adjacent databases can differ on all future outputs and, thus, have infinite sensitivity.

In this paper, we study differentially private graph algorithms under continual observation, i.e., for *dynamic* graph problems. The input is a sequence of graphs that results from node or edge updates, i.e., insertions or deletions. *Partially dynamic* algorithms only allow either insertions or deletions, *fully dynamic* algorithms allow both. After each update, the algorithm has to output a solution for the current input, i.e., the algorithm outputs a sequence of answers that is equally long as the input sequence. For differentially private graph algorithms two notions of *adjacency of graph sequences* exist: node-adjacency and edge-adjacency. Two graph sequences are *edge-adjacent* if they only differ in a single insertion or deletion of an edge. Similarly, two graph sequences are *node-adjacent* if they only differ in an insertion or deletion of a node.[1]

We initiate the study of differentially private algorithms for *non-local* partially dynamic graph problems. We consider a problem *non-local* if its (optimum) value cannot be derived from the graph's frequency histogram of constant-size subgraphs and call it *local* otherwise. Non-local problems include the cost of the minimum spanning tree, the weight of the global and s-t minimum cut, and the density of the densest subgraph. We also give improved algorithms for local graph problems and show various lower bounds on the additive error for differentially private dynamic graph algorithms.

Local problems. The only prior work on differentially private dynamic algorithms is an algorithm by Song et al. [29] for various *local* graph problems such as counting high-degree nodes, triangles and other constant-size subgraphs. We present an algorithm for these local problems that improves the additive error by a factor of $\sqrt{T}/\log^{3/2} T$, where T is the length of the update sequence. We also give the first differentially private partially-dynamic algorithm for the value of the minimum spanning tree. Table 1 lists upper and lower bounds for these results, where n is the number of nodes in the graph, W is the maximum edge weight (if applicable), D is the maximum node degree, ϵ is an arbitrarily small positive constant, and δ is the failure probability of the algorithm. We state below our main contributions in more detail. The update time of all our algorithms is linear in $\log T$ plus the time needed to solve the corresponding non-differentially private dynamic graph problem.

▶ **Theorem 1** (see Section 3). *Let $\varepsilon, \delta > 0$. There exist an ε-edge-differentially as well as an ε-node-differentially private algorithm for partially-dynamic minimum spanning tree, edge count, the number of high-degree nodes, the degree histogram, triangle count and k-star count that with probability at least $1 - \delta$ give an answer with additive error as shown in Table 1.*

[1] Of course, a graph can also be represented by a database, where, e.g., every row corresponds to an edge, but as we present algorithms that solve graph algorithmic problems we use the graph-based terminology through the paper.

Table 1 Additive errors for partially-dynamic ε-differentially private algorithms with failure probability δ. We use D for the maximum degree and W for the maximum edge weight, n for the maximum number of nodes of any graph in the input sequence, and $\Lambda = \log(1/\delta)/\varepsilon$. The upper bounds follow from Corollary 13 and Table 3 on page 11. See Section 5 for results on event-level lower bounds and Section 6 for user-level lower bounds.

Graph function	partially dynamic		fully dynamic	
	edge-adj. event-level	node-adj. event-level	edge-adj. event-level	edge-adj. user-level
min. spanning tree	$\Omega(W \log T)$, $O(W \log^{3/2} T \cdot \Lambda)$	$\Omega(W \log T)$, $O(DW \log^{3/2} T \cdot \Lambda)$	$\Omega(W \log T)$	$\Omega(nW)$
min. cut, max. matching	$\Omega(W \log T)$	$\Omega(W \log T)$	$\Omega(W \log T)$	$\Omega(nW)$
edge count	$\Omega(\log T)$, $O(\log^{3/2} T \cdot \Lambda)$	$\Omega(\log T)$, $O(D \log^{3/2} T \cdot \Lambda)$	$\Omega(\log T)$, $O(\log^{3/2} T \cdot \Lambda)$	$\Omega(n^2)$
high-degree nodes	$\Omega(\log T)$, $O(\log^{3/2} T \cdot \Lambda))$	$\Omega(\log T)$, $O(D \log^{3/2} T \cdot \Lambda)$	$\Omega(\log T)$	$\Omega(n)$
degree histogram	$\Omega(\log T)$, $O(D \log^{3/2} T \cdot \Lambda))$	$\Omega(\log T)$, $O(D^2 \log^{3/2} T \cdot \Lambda)$	$\Omega(\log T)$	$\Omega(n)$
triangle count	$\Omega(\log T)$, $O(D \log^{3/2} T \cdot \Lambda))$	$\Omega(\log T)$, $O(D^2 \log^{3/2} T \cdot \Lambda)$	$\Omega(\log T)$	$\Omega(n^3)$
k-star count	$\Omega(\log T)$, $O(D^k \log^{3/2} T \cdot \Lambda)$	$\Omega(\log T)$, $O(D^k \log^{3/2} T \cdot \Lambda)$	$\Omega(\log T)$	$\Omega(n^{k+1})$

Non-local problems. For non-local problems we present an algorithm that, by allowing a small multiplicative error, can obtain differentially private partially dynamic algorithms for a broad class of problems that includes the aforementioned problems. Table 2 lists our results for some common graph problems. The algorithm achieves the following performance.

▶ **Theorem 2** (see Theorem 16). *Let $\varepsilon, \beta, \delta, r > 0$ and let f be a function with range $[1, r]$ that is monotone on all input sequences and has sensitivity ρ. There exists an ε-differentially private dynamic algorithm with multiplicative error $(1 + \beta)$, additive error $O(\rho \log(r) \log(T)/\log(1 + \delta))$ and failure probability δ that computes f.*

Note that for partially dynamic graph algorithms it holds that $T = O(n^2)$. Thus for local problems the bounds presented in Table 1 are superior to the bounds in Table 2.

Lower bounds. We complement these upper bounds by also giving some lower bounds on the additive error of any differentially private dynamic graph algorithm. For the problems in Table 1 we show lower bounds of $\Omega(W \log T)$, resp. $\Omega(\log T)$. Note that these lower bounds apply to the partially dynamic as well as to the fully dynamic setting.

The above notion of differential privacy is also known as *event-level* differential privacy, where two graph sequences differ in at most one "event", i.e., one update operation. A more challenging notion is *user-level* differential privacy. Two graph sequences are edge-adjacent *on user-level* if they differ in *any* number of updates for a single edge (as opposed to *one* update for a single edge in the case of the former *event-level* adjacency). Note that requiring

Table 2 Private algorithms with failure probability δ with additional multiplicative error of $(1+\beta)$ for arbitrary $\beta > 0$. We use $\Lambda = 1/(\varepsilon\delta\log(1+\beta))$, D for the maximum degree, W for the maximum edge weight and n for the maximum number of nodes of any graph in the input sequence.

Graph function	partially dynamic, event-level	
	edge-adjacency	node-adjacency
minimum cut	$O(W\log(nW)\log(T)\cdot\Lambda)$	$O(DW\log(nW)\log(T)\cdot\Lambda)$
densest subgraph	$O(\log(n)\log(T)\cdot\Lambda)$	$O(\log(n)\log(T)\cdot\Lambda)$
minimum s,t-cut	$O(W\log(nW)\log(T)\cdot\Lambda)$	$O(DW\log(nW)\log(T)\cdot\Lambda)$
maximum matching	$O(W\log(nW)\log(T)\cdot\Lambda)$	$O(W\log(nW)\log(T)\cdot\Lambda)$

user-level edge-differential privacy is a more stringent requirement on the algorithm than event-level edge-differential privacy.[2] We show strong lower bounds for edge-differentially private algorithms on user-level for a broad class of dynamic graph problems.

▶ **Theorem 3** (informal, see Theorem 19). *Let f be a function on graphs, and let G_1, G_2 be arbitrary graphs. There exists a $T \geq 1$ so that any ε-edge-differentially private dynamic algorithm on user-level that computes f must have additive error $\Omega(|f(G_1) - f(G_2)|)$ on input sequences of length at least T.*

This theorem leads to the lower bounds for fully dynamic algorithms stated in Table 1.

Technical contribution. **Local problems.** Our algorithms for local problems (Theorem 1) incorporate a counting scheme by Chan et al. [6] and the *difference sequence technique* by Song et al. [29]. The difference sequence technique addresses the problem that two adjacent graphs might differ on all outputs starting from the point in the update sequence where their inputs differ. More formally, let $f_{\mathcal{G}}(t)$ be the output of the algorithm after operation t in the graph sequence $\mathcal{G}$. Then the *continuous global sensitivity* $\sum_{t=1}^{T} |f_{\mathcal{G}}(t) - f_{\mathcal{G}'}((t)|$ might be $\Theta(\rho T)$. Using the "standard" Laplacian mechanism for such a large sensitivity would, thus, lead to an additive error linear in T. The idea of [29] is to use instead the *difference sequence of f* defined as $\Delta f(t) = f(t) - f(t-1)$, as they observed that for various local graph properties the continuous global sensitivity of the difference sequence, i.e., $\sum_{t=1}^{T} |\Delta f_{\mathcal{G}}(t) - \Delta f_{\mathcal{G}'}((t)|$ can be bounded by a function independent of T. However, their resulting partially-dynamic algorithms still have an additive error linear in $\sqrt{T}$. We show how to combine the continuous global sensitivity of the difference sequence with the binary counting scheme of Chan et al. [6] to achieve partially-dynamic algorithms with additive error linear in $\log^{3/2} T$.

Furthermore we show that the approach based on the continuous global sensitivity of the difference sequence fails, if the presence or absence of a node or edge can significantly change the target function's value for all of the subsequent graphs. In particular, we show that for several graph problems like minimum cut and maximum matching changes in the function value between adjacent graph sequences can occur at every time step even for partially dynamic sequences, resulting in a continuous global sensitivity of the difference sequence that is linear in T. This implies that this technique cannot be used to achieve differentially private dynamic algorithms for these problems.

Non-local problems. We leverage the fact that the sparse vector technique [13] provides negative answers to threshold queries with little effect on the additive error to approximate monotone functions f on graphs under continual observation (e.g., the minimum cut value

[2] Node-adjacency on user-level is defined accordingly but not studied in this paper.

in an incremental graph) with multiplicative error $(1+\beta)$: If r is the maximum value of f, we choose thresholds $(1+\beta), \ldots, (1+\beta)^{\log_{1+\beta}(r)}$ for the queries. This results in at most $\log_{1+\beta}(r)$ positive answers, which affect the additive error linearly, while the at most T negative answers affect the additive error only logarithmically instead of linearly.

Lower bounds. Dwork et al. [12] had given a lower bound for counting in binary streams. We reduce this problem to partially dynamic graph problems on the event-level to achieve the event-level lower bounds.

For the user-level lower bounds we assume by contradiction that an ϵ-differentially private dynamic algorithm $\mathcal{A}$ with "small" additive error exists and construct an exponential number of graph sequences that are all user-level "edge-close" to a simple graph sequence $\mathcal{G}'$. Furthermore any two such graph sequences have at least one position with two very different graphs such that $\mathcal{A}$ (due to its small additive error) must return two different outputs at this position, which leads to two different output sequences if $\mathcal{A}$ answers within its error bound. Let O_i be the set of accurate output sequences of $\mathcal{A}$ on one of the graph sequences G_i. By the previous condition $O_i \cap O_j = \emptyset$ if $i \neq j$. As G_i is "edge-close" to $\mathcal{G}'$, there is a relatively large probability (depending on the degree of "closeness") that O_i is output when $\mathcal{A}$ runs on $\mathcal{G}'$. This holds for all i. However, since $O_i \cap O_j = \emptyset$ if $i \neq j$ and we have constructed exponentially many graph sequences G_i, the sum of these probabilities over all i adds up to a value larger than 1, which gives a contradiction. The proof is based on ideas of a lower bound proof for databases in [12].

All missing proofs can be found in the full version of the paper at `http://arxiv.org/abs/2106.14756`.

Related Work. Differential privacy, developed in [9, 11], is the de facto gold standard of privacy definitions and several lines of research have since been investigated [2, 25, 20, 10, 16, 4]. In particular, differentially private algorithms for the release of various graph statistics such as subgraph counts [19, 3, 7, 21, 32], degree distributions [18, 8, 33], minimum spanning tree [26], spectral properties [31, 1], cut problems [16, 1, 14], and parameter estimation for special classes of graphs [23] have been proposed. Dwork et al. [12] and Chan et al. [6] extended the analysis of differentially private algorithms to the regime of continual observation, i.e., to input that evolves over time. Since many data sets in applications are evolving data sets, this has lead to results for several problems motivated by practice [5, 22, 27, 15, 30]. Only one prior work analyzes evolving graphs: Song et al. [29] study problems in incremental bounded-degree graphs that are functions of local neighborhoods. Our results improve all bounds for undirected graphs initially established in [29] by a factor of $\sqrt{T}/\log^{3/2} T$ in the additive error.

2 Preliminaries

2.1 Graphs and Graph Sequences

We consider undirected graphs $G = (V, E)$, which change dynamically. Graphs may be edge-weighted, in which case $G = (V, E, w)$, where $w : E \to \mathbb{N}$. The evolution of a graph is described by a *graph sequence* $\mathcal{G} = (G_1, G_2, \ldots)$, where $G_t = (V_t, E_t)$ is derived from G_{t-1} by applying updates, i.e., inserting or deleting nodes or edges. We denote by $|\mathcal{G}|$ the length of $\mathcal{G}$, i.e., the number of graphs in the sequence. At time t we delete a set of nodes ∂V_t^- along with the corresponding edges ∂E_t^- and insert a set of nodes ∂V_t^+ and edges ∂E_t^+. More formally, $V_t = (V_{t-1} \setminus \partial V_t^-) \cup \partial V_t^+$ and $E_t = (E_{t-1} \setminus \partial E_t^-) \cup \partial E_t^+$, with initial node and edge sets V_0, E_0, which may be non-empty. If a node v is deleted, then

all incident edges are deleted at the same time, i.e., if $v \in \partial V_t^-$, then $(u,v) \in \partial E_t^-$ for all $(u,v) \in E_{t-1}$. Both endpoints of an edge inserted at time t need to be in the graph at time t, i.e., $\partial E_t^+ \subseteq ((V_{t-1} \setminus \partial V_t^-) \cup \partial V_t^+) \times ((V_{t-1} \setminus \partial V_t^-) \cup \partial V_t^+)$. The tuple $(\partial V_t^+, \partial V_t^-, \partial E_t^+, \partial E_t^-)$ is the *update* at time t. For any graph G and any update u, let $G \oplus u$ be the graph that results from applying u on G.

A graph sequence is *incremental* if $\partial E_t^- = \partial V_t^- = \emptyset$ at all time steps t. A graph sequence is *decremental* if $\partial E_t^+ = \partial V_t^+ = \emptyset$ at all time steps t. Incremental and decremental graph sequences are called *partially dynamic*. Graph sequences that are neither incremental nor decremental are *fully dynamic*.

Our goal is to continually release the value of a *graph function* f which takes a graph as input and outputs a real number. In other words, given a graph sequence $\mathcal{G} = (G_1, G_2, \dots)$ we want to compute the sequence $f(\mathcal{G}) = (f(G_1), f(G_2), \dots)$. We write $f(t)$ for $f(G_t)$. Our algorithms will compute an update to the value of f at each time step, i.e., we compute $\Delta f(t) = f(t) - f(t-1)$. We call the sequence Δf the *difference sequence of* f.

Given a graph function g the *continuous global sensitivity* $\mathrm{GS}(g)$ of g is defined as the maximum value of $||g(S) - g(S')||_1$ over all adjacent graph sequences S, S'. We will define adjacency of graph sequences below. In our case, we are often interested in the continuous global sensitivity of the difference sequence of a graph function f, which is given by the maximum value of $\sum_{t=1}^{T} |\Delta f_{\mathcal{G}}(t) - \Delta f_{\mathcal{G}'}(t)|$, where $\Delta f_{\mathcal{G}}$ and $\Delta f_{\mathcal{G}'}$ are the difference sequences of f corresponding to adjacent graph sequences $\mathcal{G}$ and $\mathcal{G}'$.

Two graphs are *edge-adjacent* if they differ in one edge. We also define global sensitivity of functions applied to a single graph. Let g be a graph function. Its *static global sensitivity* $\mathrm{GS}_{\mathrm{static}}(g)$ is the maximum value of $|g(G) - g(G')|$ over all edge-adjacent graphs G, G'.

2.2 Differential Privacy

The range of an algorithm $\mathcal{A}$, $\mathrm{Range}(\mathcal{A})$, is the set of all possible output values of $\mathcal{A}$. We denote the Laplace distribution with mean μ and scale b by $\mathrm{Lap}(\mu, b)$. If $\mu = 0$, we write $\mathrm{Lap}(b)$.

▶ **Definition 4** (ε-differential privacy). *A randomized algorithm $\mathcal{A}$ is* ε-differentially private *if for any two* adjacent *databases B, B' and any $S \subseteq \mathrm{Range}(\mathcal{A})$ we have* $\Pr[\mathcal{A}(B) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(B') \in S]$. *The parameter ε is called the* privacy loss *of $\mathcal{A}$.*

To apply Definition 4 to graph sequences we now define adjacency for graph sequences. First, we define edge-adjacency, which is useful if the data to be protected is associated with the edges in the graph sequence. Then, we define node-adjacency, which provides stronger privacy guarantees.

▶ **Definition 5** (Edge-adjacency). *Let $\mathcal{G}$, $\mathcal{G}'$ be graph sequences as defined above with associated sequences of updates (∂V_t^-), (∂V_t^+), (∂E_t^-), (∂E_t^+) and $(\partial V_t^{-\prime})$, $(\partial V_t^{+\prime})$, $(\partial E_t^{-\prime})$, $(\partial E_t^{+\prime})$. Let $\partial V_t^- = \partial V_t^{-\prime}$ and $\partial V_t^+ = \partial V_t^{+\prime}$ for all t. Let the initial node and edge sets for $\mathcal{G}$ and $\mathcal{G}'$ be $V_0 = V_0'$ and $E_0 = E_0'$. Assume w.l.o.g. that $\partial E_t^{-\prime} \subseteq \partial E_t^-$ and $\partial E_t^{+\prime} \subseteq \partial E_t^+$ for all t. The graph sequences $\mathcal{G}$ and $\mathcal{G}'$ are* adjacent on e^* *if $|\mathcal{G}| = |\mathcal{G}'|$, there exists an edge e^* and one of the following statements holds:*

1. $\partial E_t^- = \partial E_t^{-\prime} \forall\, t$ *and* $\exists t^*$ *such that* $\partial E_{t^*}^+ \setminus \partial E_{t^*}^{+\prime} = \{e^*\}$ *and* $\partial E_t^+ = \partial E_t^{+\prime} \forall\, t \neq t^*$;
2. $\partial E_t^+ = \partial E_t^{+\prime} \forall\, t$ *and* $\exists t^*$ *such that* $\partial E_{t^*}^- \setminus \partial E_{t^*}^{-\prime} = \{e^*\}$ *and* $\partial E_t^- = \partial E_t^{-\prime} \forall\, t \neq t^*$;

Remark. If $\mathcal{G}$ and $\mathcal{G}'$ are edge-adjacent, then for any index i the graphs at index i in the two sequences are edge-adjacent.

Two edge-adjacent graph sequences differ in either the insertion or the deletion of a single edge e^*. There are several special cases that fit into this definition. For example, we may have $\partial V_t^- = \partial V_t^{-\prime} = \partial V_t^+ = \partial V_{t'}^+ = \emptyset$, so only edge updates would be allowed. Similarly, we can use the definition in the incremental setting by assuming $\partial V_t^- = \partial V_t^{-\prime} = \partial E_t^- = \partial E_t^{-\prime} = \emptyset$.

The definition of node-adjacency is similar to that of edge-adjacency, but poses additional constraints on the edge update sets.

▶ **Definition 6** (Node-adjacency). *Let $\mathcal{G}$, $\mathcal{G}'$ be graph sequences as defined above with associated sequences of updates (∂V_t^-), (∂V_t^+), (∂E_t^-), (∂E_t^+) and $(\partial V_t^{-\prime})$, $(\partial V_t^{+\prime})$, $(\partial E_t^{-\prime})$, $(\partial E_t^{+\prime})$. Assume w.l.o.g. that $\partial V_t^{-\prime} \subseteq \partial V_t^-$ and $\partial V_t^{+\prime} \subseteq \partial V_t^+$ for all t. The graph sequences $\mathcal{G}$ and $\mathcal{G}'$ are* adjacent on v^* *if $|\mathcal{G}| = |\mathcal{G}'|$, there exists a node v^* and one of the following statements holds:*
1. *$\partial V_t^- = \partial V_t^{-\prime}\, \forall t$ and $\exists t^*$ such that $\partial V_t^+ \setminus \partial V_t^{+\prime} = \{v^*\}$ and $\partial V_t^+ = \partial V_t^{+\prime}\, \forall t \neq t^*$;*
2. *$\partial V_t^+ = \partial V_t^{+\prime}\, \forall t$ and $\exists t^*$ such that $\partial V_t^- \setminus \partial V_t^{-\prime} = \{v^*\}$ and $\partial V_t^- = \partial V_t^{-\prime}\, \forall t \neq t^*$;*

Additionally, all edges in ∂E_t^+ and ∂E_t^- are incident to at least one node in ∂V_t^+ and ∂V_t^-, respectively. Lastly, we require that $\partial E_t^{+\prime}$ ($\partial E_t^{-\prime}$) is the maximal subset of ∂E_t^+ (∂E_t^-) that does not contain edges incident to v^*.

We define the following notions of differential privacy based on these definitions of adjacency.

▶ **Definition 7.** *An algorithm is* ε-edge-differentially private *(on event-level) if it is ε-differentially private when considering edge-adjacency. An algorithm is* ε-node-differentially private *(on event-level) if it is ε-differentially private when considering node-adjacency.*

When explicitly stated, we consider a stronger version of ε-differential privacy, which provides adjacency on user-level. While adjacency on event-level only allows two graph sequences to differ in a single update, user-level adjacency allows any number of updates to differ as long as they affect the same edge (for edge-adjacency) or node (for node-adjacency), respectively.

▶ **Definition 8.** *Let $\mathcal{G} = (G_1, \ldots), \mathcal{G}' = (G'_1, \ldots)$ be graph sequences. The two sequences are edge-adjacent* on user-level *if there exists an edge e^* and a sequence of graph sequences $\mathcal{S} = (\mathcal{G}_1, \ldots, \mathcal{G}_\ell)$ so that $\mathcal{G}_1 = \mathcal{G}$,$\mathcal{G}_\ell = \mathcal{G}'$ and, for any $i \in [\ell - 1]$, $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$ are edge-adjacent on e^*. An algorithm is ε-edge-differentially private on user-level if it is ε-differentially private when considering edge-adjacency on user-level.*

2.3 Counting Mechanisms

Some of our algorithms for releasing differentially private estimates of functions on graph sequences rely on algorithms for counting in streams.

A *stream* $\sigma = \sigma(1)\sigma(2)\cdots$ is a string of *items* $\sigma(i) \in \{L_1, \ldots, L_2\} \subseteq \mathbb{Z}$, where the i-th item is associated with the i-th *time step*. A *binary stream* has $L_1 = 0$ and $L_2 = 1$. We denote the length of a stream, i.e., the number of time steps in the stream, by $|\sigma|$. Stream σ and σ' are adjacent if $|\sigma| = |\sigma'|$ and if there exists one and only one t^* such that $\sigma(t^*) \neq \sigma'(t^*)$ and $\sigma(t) = \sigma'(t)$ for all $t \neq t^*$.

A *counting mechanism* $\mathcal{A}(\sigma)$ takes a stream σ and outputs a real number for every time step. For all time steps t, $\mathcal{A}$'s output at time t is independent of all $\sigma(i)$ for $i > t$. At each time t a counting mechanism should estimate the count $c(t) = \sum_{i=1}^{t} \sigma(i)$.

Following Chan et al. [6] we describe our mechanisms in terms of *p-sums*, which are partial sums of the stream over a time interval. For a p-sum p we denote the beginning and end of the time interval by $\text{start}(p)$ and $\text{end}(p)$, respectively. With this notation the value of p is $\sum_{t=\text{start}(p)}^{\text{end}(p)} \sigma(t)$. To preserve privacy we add noise to p-sums and obtain *noisy p-sums*: given a p-sum p, a noisy p-sum is $\hat{p} = p + \gamma$, where γ is drawn from a Laplace-distribution.

2.4 Sparse Vector Technique

The *sparse vector technique* (SVT) was introduced by Dwork et al. [13] and was subsequently improved [17, 28]. SVT can be used to save privacy budget whenever a sequence of threshold queries $f_1, \ldots, f_T$ is evaluated on a database, but only $c \ll T$ queries are expected to exceed the threshold. Here, a threshold query asks whether a function f_i evaluates to a value above some threshold t_i on the input database. Using SVT, only queries that are answered positively reduce the privacy budget. We use the following variant of SVT, which is due to Lyu et al.

▶ **Lemma 9** ([24]). *Let D be a database, $\epsilon, \rho, c > 0$ and let $(f_1, t_1), \ldots$ be a sequence of mappings f_i from input databases to $\mathbb{R}$ and thresholds $t_i \in \mathbb{R}$, which may be generated adaptively one after another so that $\rho \geq \max_i \text{GS}_{\text{static}}(f_i)$. Algorithm 1 is ϵ-private.*

Algorithm 1 SVT algorithm [24].

Function `InitializeSvt`(D, ρ, ϵ, c)
  $\epsilon_1 \leftarrow \epsilon/2, \zeta \leftarrow \text{Lap}(\rho/\epsilon_1)$, $\epsilon_2 \leftarrow \epsilon - \epsilon_1, \mathit{count} \leftarrow 0$
Function `ProcessSvtQuery`(f_i, t_i)
  $\nu_i \leftarrow \text{Lap}(2c\rho/\epsilon_2)$
  if $\mathit{count} \geq c$ **then**
    return *abort*
  if $f_i(D) + \nu_i \geq t_i + \zeta$ **then**
    $\mathit{count} \leftarrow \mathit{count} + 1$, **return** $\top$
  else
    return $\bot$

3 Mechanisms Based on Continuous Global Sensitivity

Some of our mechanisms for privately estimating graph functions are based on mechanisms for counting in streams. In both settings, we compute the sum of a sequence of numbers and we will show that the mechanisms for counting can be transferred to the graph setting. However, there are differences in the analysis. In counting, the input streams differ at only one time step. This allows us to bound the difference in the true value between adjacent inputs and leads to low error. In the graph setting, the sequence of numbers can vary at many time steps. Here however, we use properties of the counting mechanisms to show that the total difference for this sequence can still be bounded, which results in the same error as in the counting setting.

We first generalize the counting mechanisms by Chan et al. [6] to streams of integers with bounded absolute value, and then transfer them to estimating graph functions.

3.1 Non-Binary Counting

We generalize the counting mechanisms of Chan et al. [6] to streams of numbers in $\{-L, \dots, L\}$, for some constant L. We view these algorithms as releasing noisy p-sums from which the count can be estimated. The generic algorithm is outlined in Algorithm 2 on page 9.

The algorithm releases a vector of noisy p-sums over T time steps, such that at every time step the noisy p-sums needed to estimate the count up to this time are available. Each of the noisy p-sums is computed exactly once. See the proof of Corollary 13 for an example on how to use p-sums.

In order to achieve the desired privacy loss the mechanisms need to meet the following requirements. Let $\mathcal{A}$ be a counting mechanism. We define $\text{Range}(\mathcal{A}) = \mathbb{R}^k$, where k is the total number of p-sums used by $\mathcal{A}$ and every item of the vector output by $\mathcal{A}$ is a p-sum. We assume that the time intervals represented by the p-sums in the output of $\mathcal{A}$ are deterministic and only depend on the length T of the input stream. For example, consider any two streams σ and σ' of length T. The ℓ-th element of $\mathcal{A}(\sigma)$ and $\mathcal{A}(\sigma')$ will be p-sums of the same time interval $[\text{start}(\ell), \text{end}(\ell)]$. We further assume that the p-sums are computed independently from each other in the following way: $\mathcal{A}$ computes the true p-sum and then adds noise from $\text{Lap}(z \cdot \varepsilon^{-1})$, where z is a sensitivity parameter. The next lemma is stated informally in [6].

Algorithm 2 Generic counting mechanism.

Input: privacy loss ε, stream σ of items $\{L_1, \dots, L_2\}$ with $|\sigma| = T$
Output: vector of noisy p-sums $a \in \mathbb{R}^k$, released over T time steps
Initialization: Determine which p-sums to compute based on T
At each time step $t \in \{1, \dots, T\}$:
    Compute new p-sums $p_i, \dots, p_j$ for t
    For $\ell = i, \dots, j$:
        $\hat{p}_\ell = p_\ell + \gamma_\ell, \quad \gamma_\ell \sim \text{Lap}((L_2 - L_1)\varepsilon^{-1})$
    Release new noisy p-sums $\hat{p}_i, \dots, \hat{p}_j$

▶ **Lemma 10** (Observation 1 from [6]). *Let $\mathcal{A}$ be a counting mechanism as described in Algorithm 2 with $L_1 = 0$ and $L_2 = 1$ that releases k noisy p-sums, such that the count at any time step can be computed as the sum of at most y p-sums and every item is part of at most x p-sums. Then, $\mathcal{A}$ is $(x \cdot \varepsilon)$-differentially private, and the error is $O(\varepsilon^{-1}\sqrt{y}\log\frac{1}{\delta})$ with probability at least $1 - \delta$ at each time step.*

To extend the counting mechanisms by Chan et al. to non-binary streams of values in $\{-L, \dots, L\}$, we only need to account for the increased sensitivity in the scale of the Laplace distribution. By Lemma 11, we can use the mechanisms of Chan et al. [6] to compute the sum of a stream of numbers in $\{-L, \dots, L\}$, but gain a factor $2L$ in the error.

▶ **Lemma 11** (Extension of Lemma 10). *Let $\mathcal{A}$ be a mechanism as in Algorithm 2 (see page 9) with $L_1 = -L$ and $L_2 = L$ that releases k noisy p-sums, such that the count at any time step can be computed as the sum of at most y p-sums, and every item is part of at most x p-sums. Furthermore, $\mathcal{A}$ adds noise $\text{Lap}(2L/\varepsilon)$ to every p-sum. Then, $\mathcal{A}$ is $(x \cdot \varepsilon)$-differentially private, and the error is $O(L\varepsilon^{-1}\sqrt{y}\log\frac{1}{\delta})$ with probability at least $1 - \delta$ at each time step.*

Algorithm 3 Generic graph sequence mechanism.

Input: privacy loss ε, contin. global sensitivity Γ, graph sequence $\mathcal{G} = (G_1, \ldots, G_T)$
Output: vector of noisy p-sums $a \in \mathbb{R}^k$, released over T time steps
Initialization: Determine which p-sums to compute based on T
At each time step $t \in \{1, \ldots, T\}$:
    Compute $f(t)$ and $\Delta f(t) = f(t) - f(t-1)$, $f(0) := 0$
    Compute new p-sums $p_i, \ldots, p_j$ for the sequence Δf
    For $\ell = i, \ldots, j$:
        $\hat{p}_\ell = p_\ell + \gamma_\ell, \quad \gamma_\ell \sim \mathrm{Lap}(\Gamma\varepsilon^{-1})$
    Release new noisy p-sums $\hat{p}_i, \ldots, \hat{p}_j$

3.2 Graph Functions via Counting Mechanisms

We adapt the counting mechanisms to continually release graph functions by following the approach by Song et al. [29]. Algorithm 3 outlines the generic algorithm. It is similar to Algorithm 2, with the difference that the stream of numbers to be summed is computed from a graph sequence $\mathcal{G}$. The algorithm is independent of the notion of adjacency of graph sequences, however the additive error is linear in the continuous global sensitivity of the difference sequence Δf.

In counting binary streams we considered adjacent inputs that differ at exactly one time step. In the graph setting however, the stream of numbers that we sum, i.e., the difference sequence Δf, can differ in multiple time steps between two adjacent graph sequences. We illustrate this with a simple example. Let f be the function that counts the number of edges in a graph and consider two node-adjacent incremental graph sequences $\mathcal{G}$, $\mathcal{G}'$. $\mathcal{G}$ contains an additional node v^*, that is not present in $\mathcal{G}'$. Whenever a neighbor is added to v^* in $\mathcal{G}$, the number of edges in $\mathcal{G}$ increases by more than the number of edges in $\mathcal{G}'$. Thus, every time a neighbor to v^* is inserted, the difference sequence of f will differ between $\mathcal{G}$ and $\mathcal{G}'$.

To generalize this, consider two adjacent graph sequences $\mathcal{G}$, $\mathcal{G}'$ that differ in an update at time t'. Let $\Delta f_{\mathcal{G}}$ and $\Delta f_{\mathcal{G}'}$ be the difference sequences used to compute the graph function f on $\mathcal{G}$ and $\mathcal{G}'$. As discussed above we may have $\Delta f_{\mathcal{G}}(t) \neq \Delta f_{\mathcal{G}'}(t)$ for all $t \geq t'$. Thus, more than x p-sums can be different, which complicates the proof of the privacy loss. However, we observe that the set P of p-sums with differing values can be partitioned into x sets $P_1, \ldots, P_x$, where the p-sums in each P_i cover disjoint time intervals. By using a bound on the continuous global sensitivity of the difference sequence Δf, this will lead to a privacy loss of $x \cdot \varepsilon$. The following lemma formalizes our result.

▶ **Lemma 12** (Lemma 11 for graph sequences). *Let f be a graph function whose difference sequence has continuous global sensitivity Γ. Let $0 < \delta < 1$ and $\varepsilon > 0$. Let $\mathcal{A}$ be a mechanism to estimate f as in Algorithm 3 that releases k noisy p-sums and satisfies the following conditions:*

1. *at any time step the value of a graph function f can be estimated as the sum of at most y noisy p-sums,*
2. *$\mathcal{A}$ adds independent noise from $\mathrm{Lap}(\Gamma/\varepsilon)$ to every p-sum,*
3. *the set P of p-sums computed by the algorithm can be partitioned into at most x subsets $P_1, \ldots, P_x$, such that in each partition P_x all p-sums cover disjoint time intervals. That is, for all $P_i \in \{P_1, \ldots, P_x\}$ and all $j, k \in P_i$, $j \neq k$, it holds that (1) $\mathrm{start}(j) \neq \mathrm{start}(k)$ and (2) $\mathrm{start}(j) < \mathrm{start}(k) \implies \mathrm{end}(j) < \mathrm{start}(k)$.*

Then, $\mathcal{A}$ is $(x \cdot \varepsilon)$-differentially private, and the error is $O(\Gamma\varepsilon^{-1}\sqrt{y}\log\frac{1}{\delta})$ with probability at least $1 - \delta$ at each time step.

Table 3 Global sensitivity of difference sequences.

Graph Function f	Continuous Global Sensitivity of Δf			
	Partially Dynamic		Fully Dynamic	
	node-adjacency	edge-adjacency	node-adj.	edge-adj.
edge count[a]	D	1	$\geq T$	2
high-degree nodes[a]	$2D+1$	4	$\geq T$	$\geq T$
degree histogram[a]	$4D^2+2D+1$	$8D$	$\geq 2T$	$\geq 2T$
triangle count[a]	$\binom{D}{2}$	D	$\geq T$	$\geq T$
k-star count[a]	$D\binom{D-1}{k-1}+\binom{D}{k}$	$2\cdot\left(\binom{D}{k}-\binom{D-1}{k}\right)$	$\geq T$	$\geq T$
minimum spanning tree	$2DW$	$2W-2$	$\geq T$	$\geq T$
minimum cut	$\geq T$	$\geq T$	$\geq T$	$\geq T$
maximum matching	$\geq T$	$\geq T$	$\geq T$	$\geq T$

[a]Bounds for partially dynamic node-adjacency from Song et al. [29]

We can compute the p-sums in Algorithm 3 as in the binary mechanism [6] to release ε-differentially private estimates of graph functions.

▶ **Corollary 13** (Binary mechanism). *Let f be a graph function whose difference sequence has continuous global sensitivity Γ. Let $0<\delta<1$ and $\varepsilon>0$. For each $T\in\mathbb{N}$ there exists an ε-differentially private algorithm to estimate f on a graph sequence which has error $O(\Gamma\varepsilon^{-1}\cdot\log^{3/2}T\cdot\log\delta^{-1})$ with probability at least $1-\delta$.*

3.3 Bounds on Continuous Global Sensitivity

Song et al. [29] give bounds on the continuous global sensitivity of the difference sequence for several graph functions in the incremental setting in terms of the maximum degree D. Table 3 summarizes the results on the continuous global sensitivity of difference sequences for a variety of problems in the partially dynamic and fully dynamic setting, both for edge- and node-adjacency. Note that bounds on the continuous global sensitivity of the difference sequence for incremental graph sequences hold equally for decremental graph sequences as for every incremental graph sequence there exists an equivalent decremental graph sequence which deletes nodes and edges in the reverse order.

In the partially dynamic setting, the continuous global sensitivity based approach works well for graph functions that can be expressed as the sum of local functions on the neighborhood of nodes. For non-local problems the approach is less successful. For the weight of a minimum spanning tree the continuous global sensitivity of the difference sequence is independent of the length of the graph sequence. However, for minimum cut and maximum matching this is not the case. In the fully-dynamic setting the approach seems not to be useful. Here, we can show that even for estimating the number of edges the continuous global sensitivity of the difference sequence scales linearly with T under node-adjacency. When considering edge-adjacency we only have low sensitivity for the edge count.

Using Corollary 13 we obtain ε-differentially private mechanisms with additive error that scales with $\log^{3/2}T$, compared to the factor $\sqrt{T}$ in [29]. Note that we recover their algorithm when using the Simple Mechanism II by Chan et al. [6] to sum the difference sequence.

The difference sequence approach can be employed to privately estimate the weight of a minimum spanning tree in partially dynamic graph sequences. If the edge weight is bounded by W, then the continuous global sensitivity of the difference sequence is $O(W)$ and $O(DW)$ under edge-adjacency and node-adjacency, respectively. Using Corollary 13 we obtain the algorithms outlined in Theorems 14 and 15.

▶ **Theorem 14.** *There exists an ε-edge-differentially private algorithm that outputs the weight of a minimum spanning tree on an incremental graph sequence $\mathcal{G}$ with edge-weights from the set $\{1, \ldots, W\}$. At every time step, the algorithm has error $O(W\varepsilon^{-1} \cdot \log^{3/2} T \cdot \log \delta^{-1})$ with probability at least $1 - \delta$, where T is the length of the graph sequence.*

▶ **Theorem 15.** *There exists an ε-node-differentially private algorithm that outputs the weight of a minimum spanning tree on an incremental graph sequence $\mathcal{G}$ with edge-weights from the set $\{1, \ldots, W\}$. At every time step, the algorithm has error $O(DW\varepsilon^{-1} \cdot \log^{3/2} T \cdot \log \delta^{-1})$ with probability at least $1 - \delta$, where T is the length of the graph sequence and D is the maximum degree.*

4 Upper Bound for Monotone Functions

In Section 3.3, we show that privately releasing the difference sequence of a graph sequence does not lead to good error guarantees for partially dynamic problems like minimum cut. Intuitively, the reason is that even for neighboring graph sequences $\mathcal{G}, \mathcal{G}'$, the differences of the difference sequence can be non-zero for all graphs G_i, G'_i. In other words, the difference of objective values for the graphs G_i and G'_i can constantly fluctuate during continual updates. However, the difference of objective values, regardless of fluctuations, is *always small*. We show that, by allowing an arbitrarily small *multiplicative* error, we can leverage this fact for a broad class of partially dynamic problems. In particular, we prove that there exist ε-differentially private algorithms for all dynamic problems that are non-decreasing (or non-increasing) on all valid input sequences. This includes, e.g., minimum cut, maximum matching and densest subgraph on partially dynamic inputs. See Algorithm 4 for the details and Table 2 on page 4 for explicit upper bounds for applications. We state the result for monotonically increasing functions, but it is straightforward to adapt the algorithm to monotonically decreasing functions.

Algorithm 4 Multiplicative error algorithm for monotone functions.

```
1 Function Initialize(D, ρ, ϵ, r, β)
2     k_0 ← 0
3     InitializeSvt (D, ρ, ϵ, log_{1+β}(r))                    // see Algorithm 1
4 Function Process(f_i)
5     k_i ← k_{i−1}
6     while ProcessSvtQuery(f_i, (1 + β)^{k_i}) = ⊤ do         // see Algorithm 1
7         k_i ← k_i + 1
8     return (1 + β)^{k_i}
```

▶ **Theorem 16.** *Let $r > 0$ and let f be any monotonically increasing function on dynamic inputs (e.g., graphs) with range $[1, r]$ and static global sensitivity $\rho := \mathrm{GS}_{\mathrm{static}}(f)$. Let $\beta \in (0, 1), \delta > 0$ and let $\alpha = 16 \log_{1+\beta}(r)\rho \cdot \ln(2T/\delta)/\varepsilon$. There exists an ε-differentially private algorithm for computing f with multiplicative error $(1 + \beta)$, additive error α and failure probability δ.*

5 Lower Bounds for Event-Level Privacy

We can show lower bounds for the error of edge- and node-differentially private algorithms in the partially dynamic setting. We derive the bounds by reducing differentially private counting in binary streams to these problems and apply a lower bound of Dwork et al. [12], which we restate here.

▶ **Theorem 17** (Lower bound for counting in binary streams [12]). *Any differentially private event-level algorithm for counting over T rounds must have error $\Omega(\log T)$ (even with $\varepsilon = 1$).*

We obtain a lower bound of $\Omega(W \log T)$ for minimum cut, maximum weighted matching and minimum spanning tree. The same approach yields a lower bound of $\Omega(\log T)$ for the subgraph-counting problems, counting the number of high-degree nodes and the degree histogram. See Table 1 on page 3. Note that any lower bound for the incremental setting can be transferred to the decremental setting, using the same reductions but proceeding in reverse.

6 Lower Bound for User-Level Privacy

We show that for several fundamental problems on dynamic graphs like minimum spanning tree and minimum cut, a differentially private algorithm with edge-adjacency on user-level must have an additive error that is linear in the maximum function value. Technically, we define the *spread* of a graph function as the maximum difference of the function's value on any two graphs. Then, we show that any algorithm must have an error that is linear in the graph function's spread. We write this section in terms of edge-adjacency but the corresponding result for node-adjacency carries over.

▶ **Definition 18.** *Let G_1 and G_2 be a pair of graphs. We define $\tau(G_1, G_2)$ to be an update sequence $u_1, \ldots, u_\ell$ of minimum length that transforms G_1 into G_2. We denote the graph sequence that results from applying $\tau(G_1, G_2)$ to G_1 by $\mathrm{T}(G_1, G_2)$.*

Let $s, \ell : \mathbb{N} \to \{2i \mid i \in \mathbb{N}\}$ be functions. A graph function f has spread $(s(n), \ell(n))$ on inputs of size n if, for every n, there exist two graphs G_1, G_2 of size n so that $|f(G_1) - f(G_2)| \geq s(n)$ and $|\tau(G_1, G_2)| = \ell(n)$.

See Table 1 on page 3 for the resulting lower bounds.

▶ **Theorem 19.** *Let $\epsilon, \delta > 0$ and let f be a graph function with spread (s, ℓ) that spares an edge e. For streams of length T on graphs of size n, where $T > \log(e^{4\epsilon\ell}/(1-\delta)) \in O(\epsilon\ell + \log(1/(1-\delta)))$, every ϵ-differentially private dynamic algorithm with user-level edge-adjacency that computes f with probability at least $1 - \delta$ must have error $\Omega(s(n))$.*

▶ **Fact 20.** *Minimum spanning tree has spread $(\Theta(nW), \Theta(n))$. Minimum cut has spread $(\Theta(nW), \Theta(n^2))$. Maximal matching has spread $(\Theta(n), \Theta(n))$. Maximum cardinality matching has spread $(\Theta(n), \Theta(n))$. Maximum weight matching has spread $(\Theta(nW), \Theta(n))$.*

References

1 Raman Arora and Jalaj Upadhyay. On differentially private graph sparsification and applications. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 13399–13410. Curran Associates, Inc., 2019.

2 Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In *Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '07, pages 273–282, New York, NY, USA, 2007. Association for Computing Machinery. doi:10.1145/1265530.1265569.

3 Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 87–96, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2422436.2422449.

4 BlumAvrim, LigettKatrina, and RothAaron. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)*, May 2013. doi:10.1145/2450142.2450148.

5 T. H. Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially Private Continual Monitoring of Heavy Hitters from Distributed Streams. In Simone Fischer-Hübner and Matthew Wright, editors, *Privacy Enhancing Technologies*, Lecture Notes in Computer Science, pages 140–159, Berlin, Heidelberg, 2012. Springer. doi:10.1007/978-3-642-31680-7_8.

6 T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3), 2011. doi:10.1145/2043621.2043626.

7 Shixi Chen and Shuigeng Zhou. Recursive mechanism: Towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 653–664, New York, NY, USA, June 2013. Association for Computing Machinery. doi:10.1145/2463676.2465304.

8 Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing Graph Degree Distribution with Node Differential Privacy. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 123–138, New York, NY, USA, June 2016. Association for Computing Machinery. doi:10.1145/2882903.2926745.

9 Cynthia Dwork. Differential Privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 1–12, Berlin, Heidelberg, 2006. Springer. doi:10.1007/11787006_1.

10 Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, pages 371–380, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1536414.1536466.

11 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, Lecture Notes in Computer Science, pages 265–284, Berlin, Heidelberg, 2006. Springer. doi:10.1007/11681878_14.

12 Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 715–724, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1806689.1806787.

13 Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil Vadhan. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, pages 381–390, New York, NY, USA, May 2009. Association for Computing Machinery. doi:10.1145/1536414.1536467.

14 Marek Eliás, Michael Kapralov, Janardhan Kulkarni, and Yin Tat Lee. Differentially Private Release of Synthetic Graphs. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 560–578. Society for Industrial and Applied Mathematics, December 2019. doi:10.1137/1.9781611975994.34.

15 M. A. Erdogdu and N. Fawaz. Privacy-utility trade-off under continual observation. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 1801–1805, June 2015. doi:10.1109/ISIT.2015.7282766.

16 Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially Private Combinatorial Optimization. In *Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms*, Proceedings, pages 1106–1125. Society for Industrial and Applied Mathematics, January 2010. doi:10.1137/1.9781611973075.90.

17 Moritz Hardt and Guy N. Rothblum. A Multiplicative Weights Mechanism for Privacy-Preserving Data Analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 61–70, October 2010. doi:10.1109/FOCS.2010.85.

18 M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate Estimation of the Degree Distribution of Private Networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178, 2009. doi:10.1109/ICDM.2009.11.

19 Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157, August 2011. doi:10.14778/3402707.3402749.

20 S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What Can We Learn Privately? In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 531–540, 2008. doi:10.1109/FOCS.2008.27.

21 Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing Graphs with Node Differential Privacy. In Amit Sahai, editor, *Theory of Cryptography*, Lecture Notes in Computer Science, pages 457–476, Berlin, Heidelberg, 2013. Springer. doi:10.1007/978-3-642-36594-2_26.

22 Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12):1155–1166, 2014. doi:10.14778/2732977.2732989.

23 Wentian Lu and Gerome Miklau. Exponential random graph estimation under differential privacy. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 921–930, New York, NY, USA, August 2014. Association for Computing Machinery. doi:10.1145/2623330.2623683.

24 Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *Proceedings of the VLDB Endowment*, 10(6), 2017.

25 F. McSherry and K. Talwar. Mechanism Design via Differential Privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103, October 2007. doi:10.1109/FOCS.2007.66.

26 Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 75–84, New York, NY, USA, June 2007. Association for Computing Machinery. doi:10.1145/1250790.1250803.

27 J. Le Ny and G. J. Pappas. Differentially Private Filtering. *IEEE Transactions on Automatic Control*, 59(2):341–354, 2014. doi:10.1109/TAC.2013.2283096.

28 Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, pages 765–774, New York, NY, USA, June 2010. Association for Computing Machinery. doi:10.1145/1806689.1806794.

29 Shuang Song, Susan Little, Sanjay Mehta, Staal Vinterbo, and Kamalika Chaudhuri. Differentially private continual release of graph statistics, 2018. arXiv:1809.02575.

30 Q. Wang, Y. Zhang, X. Lu, Z. Wang, Z. Qin, and K. Ren. Real-Time and Spatio-Temporal Crowd-Sourced Social Network Data Publishing with Differential Privacy. *IEEE Transactions on Dependable and Secure Computing*, 15(4):591–606, 2018. doi:10.1109/TDSC.2016.2599873.

31 Yue Wang, Xintao Wu, and Leting Wu. Differential Privacy Preserving Spectral Graph Analysis. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, pages 329–340, Berlin, Heidelberg, 2013. Springer. doi:10.1007/978-3-642-37456-2_28.

32 Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Private Release of Graph Statistics using Ladder Functions. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 731–745, New York, NY, USA, May 2015. Association for Computing Machinery. doi:10.1145/2723372.2737785.

33 Sen Zhang, Weiwei Ni, and Nan Fu. Differentially Private Graph Publishing with Degree Distribution Preservation. *Computers & Security*, page 102285, 2021. doi:10.1016/j.cose.2021.102285.