Fully Dynamic Algorithms for Graph Databases with Edge Differential Privacy

SOFYA RASKHODNIKOVA, Boston University, USA

TERESA ANNA STEINER, University of Southern Denmark, Denmark

We study differentially private algorithms for analyzing graph databases in the challenging setting of continual release with fully dynamic updates, where edges are inserted and deleted over time, and the algorithm is required to update the solution at every time step. Previous work has presented differentially private algorithms for many graph problems that can handle insertions only or deletions only (called *partially dynamic algorithms*) and obtained some hardness results for the fully dynamic setting. The only algorithms in the latter setting were for the edge count, given by Fichtenberger, Henzinger, and Ost (ESA '21), and for releasing the values of all graph cuts, given by Fichtenberger, Henzinger, and Upadhyay (ICML '23). We provide the first differentially private and fully dynamic graph algorithms for several other fundamental graph statistics (including the triangle count, the number of connected components, the size of the maximum matching, and the degree histogram), analyze their error, and show strong lower bounds on the error for all algorithms in this setting. Previously, only lower bounds for *purely* differentially private algorithms were known; our lower bounds give an exponential improvement in terms of the dependence on the number of time steps, while applying to algorithms satisfying *pure* as well as *approximate* differential privacy.

We study two variants of *edge differential privacy* for fully dynamic graph algorithms: *event-level* and *item-level*. Under the former notion, two graph database update sequences are considered neighboring if, roughly speaking, they differ in at most one update; under the latter notion, they can differ only in updates pertaining to one edge. Differential privacy requires that for any two neighboring inputs, the output distributions of the algorithm are close. We give upper and lower bounds on the error of both—event-level and item-level—fully dynamic algorithms for several fundamental graph problems. No fully dynamic algorithms that are private at the item-level (the more stringent of the two notions) were known before. In the case of item-level privacy, for several problems, our algorithms match our lower bounds.

CCS Concepts: • **Security and privacy**; • **Theory of computation** → **Dynamic graph algorithms**;

Additional Key Words and Phrases: differential privacy, dynamic graph databases, lower bounds

1 Introduction

Graph databases provide important tools for understanding networks and relationships. Specifically, graphs can be used to model complex relational databases, and analyzing those has applications in recommender systems, analysis of social networks, market analysis, etc. [48]. However, some graph databases contain sensitive personal information, and it is an ongoing effort within the database community to develop methods for protecting sensitive data when results of statistical analyses performed on them are published. Differential privacy [25] has emerged as the standard

---

Authors' Contact Information: Sofya Raskhodnikova, Boston University, Boston, USA, sofya@bu.edu; Teresa Anna Steiner, University of Southern Denmark, Odense, Denmark, steiner@imada.sdu.dk.

for rigorous privacy guarantees. It has been adapted to graph data and widely investigated in the *static setting*, also referred to as the *batch setting*, where the input graph database does not change (see [57] for a survey of differentially private analysis of graphs in this setting). An additional challenge in studying graph databases that capture information about people is that they evolve over time. Differential privacy was first studied in the setting of *continual release* (also called *continual observation*), where data changes over time and published statistic must be continually updated, by Dwork et al. [27] and Chan et al. [12]. This difficult setting was adapted to graph data and investigated by Song et al. [62], Fichtenberger et al. [33], and Jain et al. [43]. In this setting, the algorithm receives a sequence of database updates, one per time step, where each update is either an insertion or a deletion of an edge. *Fully dynamic* algorithms handle both insertions and deletions, whereas *partially dynamic* algorithms handle insertions only or deletions only. In this work, we study fully dynamic graph algorithms in the continual release model.

Differential privacy (DP), intuitively, guarantees that, for any two neighboring databases, the output distributions of the algorithm are roughly the same. We consider *edge DP,* introduced in [54], that uses the notion of *edge-neighboring graphs*. Two graphs are edge-neighboring if they differ in one edge. (There is also a stronger notion of *node DP*, first studied in [4, 13, 47].) Differential privacy is defined with two parameters, ε and δ; see Definitions A.2 and 2.4. When $\delta = 0$, it is referred to as *pure DP*; the setting when $\delta > 0$ is called *approximate DP*.

The continual release model comes with two natural definitions of neighboring sequences and two corresponding variants of differential privacy: *event-level* and *item-level*, and we study both of them. Update sequences are *event-neighboring* if they differ in one update; they are *item-neighboring* if they differ on an arbitrary number of updates pertaining to one item. For edge DP, we adapt these definitions as follows. Two graph sequences are considered *event-level edge-neighboring* if one can be obtained from the other by replacing either one update with a no-op or two consecutive updates on the same edge with no-ops.[1] Two graph sequences are *item-level edge-neighboring* if they are the same on all updates, except updates on one edge. Event-level privacy is less stringent than item-level privacy. Note that there is no distinction between event-level and item-level edge DP for partially dynamic algorithms, because each edge can be updated at most once. But the picture is more nuanced for fully dynamic algorithms.

Our goal is to investigate the best error achievable by fully dynamic, edge differentially private (edge-DP) graph algorithms. We call an algorithm (α, β)-accurate if, with probability at least $1 - \beta$, it has additive error at most α at every time step. For algorithms that output real-valued vectors, the additive error is measured in terms of L_∞, i.e., as the maximum over all coordinates of the vector. We express our error bounds in terms of the number of nodes in the graph, denoted by N; the number of time steps, called the *time horizon* and denoted by T; and sometimes also in terms of the bound D on the maximum degree.

The systematic investigation by Fichtenberger et al. [33] provided partially dynamic private algorithms with additive error polylog T for numerous graph problems. The partially dynamic setting under node DP was further investigated by Jain et al. [43]. The only fully dynamic graph algorithms in previous work were for the edge count, given by Fichtenberger et al. [33], and for releasing the values of all graph cuts, given by Fichtenberger et al. [34]. Both algorithms satisfy event-level, edge DP; the former algorithm has dependence $O(\log^{3/2} T)$ on the time horizon T in the

---

[1]We consider two updates on the same edge e consecutive if the updates between them do not involve e. Consecutive updates on the same edge must be of different types: an insertion and a deletion (in either order). We allow up to *two* updates to be replaced, as opposed to one in the classical continual release setting, for a technical reason: since an edge can be inserted only when it is absent and deleted only when it is present, there exist update sequences for which changing any one update results in an invalid graph sequence. This happens, for example, if the updates alternate between inserting and deleting the same edge, ending in an insertion. As a result, our definition is more natural in the dynamic setting.

	Upper bounds	Lower bounds	Lower bounds out-det	[33] $\delta = 0$
$f_\triangle$	$\tilde{O}(\min(T, \sqrt{TN}, \sqrt[3]{T}N, N^3))$ Theorem 3.8	$\Omega(\min(T, \sqrt[3]{T}N^{2/3}, N^2))$ Theorem 3.1	same as lower bounds	
f_{CC} $\ldots$	$\tilde{O}(\min(\sqrt[3]{T}, N))$ Table 2	$\Omega(\min(\sqrt[4]{T}, \sqrt{N}))$ Corollary 6.8	$\tilde{\Omega}(\min(\sqrt[3]{T}, N))$ full version [59]	$\Omega(\log T)$
$f_{\deg}$	$O(\log T \log(TN))$ Lemma 4.2	$\Omega(\log T)$ Remark 1	same as lower bounds	—

Table 1. Bounds on error α for (α, β)-accurate, **event-level** (ε, δ)-edge-DP fully dynamic graph algorithms (suppressing factors polynomial in $\frac{1}{\varepsilon} \log \frac{1}{\delta\beta}$). We use $\tilde{O}(X)$ and $\tilde{\Omega}(X)$ to hide polylog X factors. The bounds under f_{CC}... hold for $f_{CC}, f_{MM}, f_{\geq \tau}, f_{\text{degHist}}$.

	Approximate DP ($\delta > 0$)		Pure DP ($\delta = 0$)		
	Upper bounds	Lower bounds	Upper bounds	Lower bounds	[33]
$f_\triangle$	$\tilde{O}(\min(T^{\frac{4}{3}}, \sqrt[3]{T}N, N^3))$ Corollary 5.3	$\tilde{\Omega}(\min(T^{\frac{7}{6}}, \sqrt[3]{T}N, N^3))$ Theorem 3.11	$\tilde{O}(\min(T^{\frac{3}{2}}, \sqrt{TN}, N^3))$ Corollary 5.3	$\tilde{\Omega}(\min(T^{\frac{5}{4}}, \sqrt{TN}, N^3))$ Theorem 3.11	$\Omega(N^3)$
f_{CC} $\ldots$	$\tilde{O}(\min(\sqrt[3]{T}, N))$ Corollary 5.2	$\tilde{\Omega}(\min(\sqrt[3]{T}, N))$ Corollary 6.11	$\tilde{O}(\min(\sqrt{T}, N))$ Corollary 5.2	$\tilde{\Omega}(\min(\sqrt{T}, N))$ Corollary 6.11	$\Omega(N)$
$f_{\deg}$		as for f_{CC} Remark 1		as for f_{CC} Remark 1	—
f_{edges}	$\tilde{O}(\min(\sqrt[3]{T}, N^2))$ Corollary 5.2	$\Omega(\min(\sqrt[3]{T}, N^2))$ Theorem 6.12	$\tilde{O}(\min(\sqrt{T}, N^2))$ Corollary 5.2	$\Omega(\min(\sqrt{T}, N^2))$ Theorem 6.12	$\Omega(N^2)$

Table 2. Bounds on error α for (α, β)-accurate, **item-level** (ε, δ)-edge-DP fully dynamic graph algorithms (suppressing factors polynomial in $\frac{1}{\varepsilon} \log \frac{1}{\delta\beta}$. We use $\tilde{O}(X)$ and $\tilde{\Omega}(X)$ to hide factors polylogarithmical in X). The bounds under f_{CC}.. hold for $f_{CC}, f_{MM}, f_{\geq \tau}, f_{\text{degHist}}$. Bounds from [33] hold only for sufficiently large T.

additive error, whereas the latter algorithm has dependence $O(N^{3/2}\sqrt{\log N}\log^{3/2} T)$ on T and the number of nodes N. Fichtenberger et al. [33] also obtained $\Omega(\log T)$ lower bounds for many graph problems in this setting and lower bounds in terms of N (that hold for large T) in the item-level DP setting. Their lower bounds are only for *pure DP*, that is, when $\delta = 0$. Fichtenberger et al. [34] show a lower bound on the accuracy of continually releasing the value of the min cut with event-level, edge DP, but in a different model: they have parallel edges and allow multiple edges to be changed at every time step, both of which make the problem harder and lead to a lower bound on the error that is larger than our event-level upper bound for the value of min cut.

1.1 Our results

Our results on event-level and item-level edge-DP are summarized in Tables 1 and 2, respectively.

We give the *first fully dynamic,* differentially private algorithms for several fundamental graph statistics, including triangle count ($f_\triangle$), the number of connected components (f_{CC}), the size of the maximum matching (f_{MM}), the number of nodes of degree at least τ ($f_{\geq \tau}$), the degree histogram (f_{degHist}), and the degree list ($f_{\deg}$) with edge DP at both event level and item level. For the edge count (f_{edges}), we give the first item-level edge-DP algorithm. (See Definition 2.7 for problem definitions.) Our event-level algorithm for $f_{\deg}$ has additive error α polylogarithmic in T and N. All other algorithms have error polynomial in these parameters.

Event-level. We demonstrate that $f_\triangle$, f_{CC}, f_{MM}, $f_{\geq\tau}$, and f_{degHist} are fundamentally different from EdgeCount and DegreeList by showing that every event-level edge-DP algorithm for these five problems must have additive error polynomial in T and N (as opposed to polylogarithmic in these parameters), thus providing an exponential improvement on the previously known lower bounds of $\Omega(\log T)$ by [33] in terms of the dependence on the time horizon T. Our lower bounds apply for general $\delta \in [0, 1)$, that is, even for approximate DP, whereas the lower bounds in [33] hold only for pure DP. Specifically, we show that when N is sufficiently large, the best additive error of event-level edge-DP algorithms must be $\Theta(T)$ for $f_\triangle$ and between $O(T^{1/3})$ and $\Omega(T^{1/4})$ for $f_{CC}, f_{MM}, f_{\geq\tau}, f_{\text{degHist}}$.

Event-level output-determined. While there remains an intriguing gap between upper and lower bounds for these problems, we are able to eliminate it for $f_{CC}, f_{MM}, f_{\geq\tau}, f_{\text{degHist}}$ for a special class of algorithms, called *output-determined (out-det)*, first considered in [39], where the authors showed a lower bound for output-determined algorithms for counting the number of distinct elements in a stream. Output-determined algorithms have roughly the same output distributions on inputs with the same output sequences. All known algorithms for these four problems (all of which come from our work) are output-determined. Our lower bounds for output-determined algorithms demonstrate that we would need a different approach to improve the error. The proof and details for output-determined lower bounds are given in the full version [59] of this paper.

Item-level. We also give item-level lower bounds on the additive error for fully dynamic graph algorithms. For all problems listed in the table, except for the triangle count, our item-level lower bounds match the error of our algorithms up to polylogarithmic factors. Our bounds for triangle count under item-level (ε, δ)-edge-DP are tight (up to polylogarithmic factors) for $N \leq T^{5/6}$, and under item-level ε-edge-DP for $N \leq T^{3/4}$. Further, we show that when N is sufficiently large, the additive error of item-level (ε, δ)-edge-DP algorithms must be between $O(T^{4/3})$ and $\Omega(T^{7/6})$ for $f_\triangle$ and $\tilde{\Theta}(T^{1/3})$ for $f_{CC}, f_{MM}, f_{\geq\tau}, f_{\text{degHist}}, f_{\text{edges}}$, and f_{deg} when $\delta > 0$. When $\delta = 0$, it is between $T^{3/2}$ and $T^{5/4}$ for $f_\triangle$ and $\Theta(\sqrt{T})$ for the remaining problems.

Discussion and Bounded Degree. Putting our results together with the $O(\log^{3/2} T)$ event-level upper bound for f_{edges} by [33], we see that f_{edges} and f_{deg} become much more difficult in the item-level setting: the error grows from polylog T to $T^{1/2}$ or $T^{1/3}$ (depending whether we consider pure or approximate DP). For the remaining problems, while $f_{CC}, f_{MM}, f_{\geq\tau}, f_{\text{degHist}}$ behave similarly in terms of the additive error of fully dynamic algorithms, the triangle count stands out. We further investigate it in the event-level setting by considering error bounds in terms of the maximum degree D and time horizon T. We prove the upper bound of $\tilde{O}(\min(\sqrt{TD}, \sqrt[3]{T}D))$ and the lower bound of $\Omega(\min(\sqrt[3]{T}D^{2/3}, \sqrt[4]{T}D))$ for this problem. We stress that our algorithms are differentially private even when the degree bound is violated, but the accuracy guarantees expressed in terms of D rely on the input graph being of degree at most D at every time step. Subroutines that are differentially private only when the input graphs have maximum degree at most D are called *D-restricted*. As shown by Jain et al. [43], such a subroutine can exhibit blatant failures of differential privacy on graphs that do not satisfy the promise if they are used as stand-alone algorithms.

1.2 Our Techniques

We prove the first superlogarithmic lower bounds for dynamic differentially private algorithms for graph problems. Our starting point for proving lower bounds is the sequential embedding technique proposed by Jain et al. [42] and further developed by Jain et al. [41]. The main idea is to reduce from a problem in the batch setting that returns multiple outputs about the same individuals and use different time steps to extract answers that correspond to different outputs. In particular, [42] used a reduction from the 1-way marginals problem in the batch setting and then applied lower bounds

of [37] and [11] for releasing all 1-way marginals; [41] used a reduction from the Inner Product problem in the batch setting and applied the lower bounds for this problem from [17, 20, 26, 51].

Lower bounds via Submatrix Queries. To prove our event-level lower bound for TriangleCount (stated in Theorem 3.1), we reduce from the Submatrix Query problem, which has not been used in the context of continual release before. It was defined by Eden et al. [30] for studying the local model of differential privacy, where each party holds their own private data and interacts with the rest of the world using differentially private algorithms. In the submatrix query problem, the input database is an $n \times n$ matrix Y of random bits, each belonging to a different individual. Each submatrix query is specified by two vectors $a, b \in \{0, 1\}^n$. The answer to the query (a, b) is $a^T Y b = \sum_{i=1}^{n} \sum_{j=1}^{n} a[i] Y[i, j] b[j]$. A submatrix query is a type of a *linear query*. The answer to a linear query is a dot product of the secret database (viewed as a vector; in our case, of length n^2) and a query vector of the same length (in our case, a vectorization of the outer product of vectors a and b). Dinur and Nissim [20], in the paper that laid foundations for establishing the field of differential privacy, show that answering many random linear queries too accurately leads to blatant privacy violations. Eden et al. [30] extended their attack to submatrix queries. Implicitly, they showed differentially private algorithms for answering $\Omega(n^2)$ submatix queries on $n \times n$ database must have additive error $\Omega(n)$. We use this lower bound together with our reduction from Submatrix Queries to triangle counting with event-level edge-DP.

Our reduction transforms a database Y and a set Q of submatrix queries into a dynamic graph sequence $\mathcal{S}$. It ensures that the answers to different submatrix queries in Q correspond to triangle counts in $\mathcal{S}$ at different time steps. Moreover, the transformations of neighboring databases Y and Y' (with the same query set Q) return event-level, edge-neighboring graph sequences. Our reduction encodes the secret database Y as edges of a bipartite graph. Then it crucially uses insertions and deletions by first inserting and then deleting edges that encode each query. The main advantage of using submatrix queries instead of arbitrary linear queries in our reduction is that they can be encoded more efficiently, i.e., with fewer edges. This creates shorter dynamic graph sequences, leading to stronger lower bounds in terms of the time horizon T.

General lower bound framework. We provide a general lower bound framework for fully dynamic graph algorithms, which we employ to prove nearly all of our lower bounds (with the notable exception of event-level triangle counting, for which we use the submatrix queries method described above, in order to get a better bound). The framework is based on the existence of a small graph gadget with two special edges such that the value of the target function (e.g., f_{MM}) remains the same when only one of the special edges is removed, but changes by a specific amount when both special edges are removed. We call such a gadget *2-edge distinguishing for f* (see Figure 2). We show how to use a 2-edge distinguishing gadget for f to obtain a reduction from the Inner Product problem (see Definition 6.5) in the batch setting to the continual release of f in the fully dynamic, event-level setting. We also show how to use a slightly simpler gadget for f to obtain a reduction from 1-way Marginals (see Definition 3.12) in the batch setting to the continual release of f in the fully dynamic, item-level setting. Finally, 2-edge distinguishing gadgets are used in the general reduction from 1-way Marginals to get event-level lower bounds in the output-determined setting. We then apply our general framework to obtain numerous lower bounds for specific problems.

Transformation from D-restricted to general algorithms in the event-level setting. One of the most common methods for obtaining DP graph algorithms is to start by designing an algorithm that is tailored to a specific graph family, typically graphs of degree at most D. The reason for this is that some graph statistics have much lower sensitivity on bounded degree graphs. E.g., if one edge in an N-node graph is added or removed, the triangle count can change by $N - 2$; however, in a graph of degree at most D, it can change by at most $D - 1$. In the batch setting, many techniques (notably,

based on graph projections [4, 16, 47] and Lipschitz extensions [4, 13, 16, 44, 47, 57, 58]) have been developed for transforming algorithms tailored to bounded-degree graphs to algorithms that are private on *all graphs* without noticeably increasing their error on bounded-degree graphs. For dynamic graph algorithms, D-restricted algorithms were studied in [33, 62]. For the insertions-only setting, Jain et al. [43] present a general projection that transforms a D-restricted algorithm into an algorithm which is differentially private on the universe of *all* input graph sequences, with similar error guarantees depending on the maximum degree of the input sequence (up to log factors).

We obtain a similar result in the fully dynamic setting for event-level edge DP: i.e., if for all D there exists an event-level D-restricted edge-DP algorithm for estimating a function on all dynamic graph sequences of maximum degree D, with an error bound depending on D, then there exists an event-level edge-DP algorithm on *all* dynamic graph sequences, with an error bound depending on the maximum degree of the sequence. The error bounds are the same as for the D-restricted case, up to factors logarithmic in TN and small dependence on the privacy parameters and the failure probability β.

One of the tools in our transformation is our event-level algorithm for degree list, which is based on the continual histograms algorithm from [34]. We use the degree-list algorithm to keep a running estimate of the maximum degree. We run the corresponding degree restricted mechanism until the maximum degree increases significantly. We then re-initialize a new restricted mechanism with a higher degree bound.

Fully dynamic algorithm for triangle count. Our event-level algorithm for triangle count uses our transformation from D-restricted to general algorithms. To get a D-restricted algorithm, we use a binary-tree-based mechanism for counting on the difference sequence for $f_\triangle$ (i.e., a sequence that records the change in $f_\triangle$ between time steps) and then add noise using the Gaussian mechanism. This strategy is used for other graph problems in [33] with Laplacian noise instead of Gaussian noise. To bound the error of our mechanism, we give a careful analysis of the L_2-sensitivity of the counts stored in the binary tree.

1.3 Additional Related Work

Prior work. The two concurrent works [12, 27] that initiated the investigation of the continual release model, also proposed the binary-tree mechanism for computing sums of bits, a.k.a. continual counting. The problem of continual counting was further studied in [14, 21, 28, 34, 40]. The binary-tree mechanism has been extended to work for sums of real values [55], weighted sums [7], counting distinct elements [7, 32, 35, 39, 41] and, most relevantly, partially dynamic algorithms for graph statistics [33].

The first lower bound in the continual release model was an $\Omega(\log T)$ bound on the additive error of continual counting, shown by [27]. The first lower bounds that depended polynomially on T were proved by Jain et al. [42] and applied to the problems of releasing the value and index of the attribute with the highest sum, given a database that stores whether each individual possesses a given attribute. They pioneered the sequential embedding technique that reduces multiple instances of a static problem to a dynamic problem. Like in that paper, we also reduce from the 1-way marginals problem to obtain some of our lower bounds. However, our lower bound for the triangle count is proved by a reduction from a different problem, and our reductions use the specific structure of the graph problems we consider.

First edge-DP algorithms appeared in [54]; they handled the cost of the minimum spanning tree and triangle count. Since then differentially private graph algorithms have been designed for a variety of tasks, including estimating subgraph counts [4, 13, 19, 45, 47, 49], degree list [46], degree and triangle distributions [16, 38, 49, 58], the number of connected components [44], spectral

properties [2, 64], parameters in stochastic block models [8, 9, 61]; training of graph neural networks [15]; generating synthetic graphs [46, 68], and many more [1, 3, 5, 18, 36, 50, 52, 53, 56, 60, 63, 65–67].

Concurrent work. Recently and independently, [31] showed similar lower bounds for fully dynamic algorithms achieving event-level, ε-edge-DP via a reduction from InnerProduct; however, the focus of their paper is not on the fully dynamic model, but on obtaining better additive error in the insertions-only model. In the fully dynamic model, they prove $\Omega\left(\min\left(\sqrt{\frac{N}{\varepsilon}}, \frac{T^{1/4}}{\varepsilon^{3/4}}, N, T \right) \right)$ lower bounds for f_{MM}, f_{CC}, and $f_{\triangle}$. Their lower bounds match ours from Corollary 6.8 for f_{MM} and f_{CC} in terms of the dependence on N and T, but also have an explicit dependence on ε in them for the special case of pure differential privacy (ours are stated for constant ε and more general (ε, δ)-DP). However, our lower bound for $f_{\triangle}$ from Theorem 3.1 is better by a polynomial factor in T and N.

1.4 Discussion and Open Questions

Our transformation from D-restricted to general edge-DP algorithms works in the event-level setting. An interesting open question is if one can design such a transformation for the item-level setting (with a small error blowup).

Our error bounds for f_{CC}, f_{MM}, $f_{\geq \tau}$, and f_{degHist} exhibit similar dependence on T as the bounds in [41] on counting the number of distinct elements in turnstile streams in the continual release model. Unlike in our setting, where an edge can be added only if it is present and deleted only if it is absent, turnstile streams might have an element added or deleted no matter what the current count for this element is. Only elements with positive counts are included in the current distinct element count. Interestingly, [39] show that when elements can only be added when absent and deleted when present (i.e., in the so-called "likes" model), the additive error drops down to polylogartimic in T. We conjecture that improving the bounds for counting distinct elements in the event-level setting would lead to better bounds for graph problems as well.

1.5 Organization of Technical Sections

We start by giving basic notation, problem and model definitions in Section 2. In Section 3, we collect all our results on triangle counting that require specialized techniques. Section 4 presents the general transformation from degree-restricted to general event-level algorithms. Section 5 presents algorithms based on recomputation at regular intervals. It also applies these tools to get most upper bounds on the error presented in Tables 1 and 2. Section 6 presents our lower bound framework and most lower bounds on the error from the tables. Appendix A collects some privacy background. Appendices B, C, D, and E contain some omitted proofs from Sections 3, 4, 5, and 6, respectively.

2 Preliminaries

We use log to denote the logarithm base 2 and ln to denote the natural logarithm. Let $[T]$ denote the set $\{1, 2, \ldots, T\}$.

2.1 Graph Sequences

A *(simple)* graph on a set of nodes V is a pair (V, E), where E is a subset of $\{\{i, j\} : i, j \in V \text{ and } i \neq j\}$. For a set of nodes V, let $\mathcal{G}(V)$ denote the set of all simple graphs on V. The symmetric difference of two sets S and S', denoted $\Delta(S, S')$, is $(S \setminus S') \cup (S' \setminus S)$.

Definition 2.1 (Edge-neighboring). Graphs $G = (V, E)$ and $H = (V, E')$ are *edge-neighboring*, denoted by $G \sim H$, if E and E' differ in at most one element, i.e., $|\Delta(E, E')| \leq 1$.

Definition 2.2 (Dynamic graph sequence). Let V be a set of nodes and $T \in \mathbb{N}$. For all $t \in \{0\} \cup [T]$, let G_t be a graph (V, E_t) with some edge set $E_t \subseteq V \times V$ and $E_0 = \emptyset$. Then $\mathcal{S} = (G_1, \ldots, G_T)$ is a *dynamic graph sequence of length T* if $|\Delta(E_t, E_{t-1})| \leq 1$ for all $t \in [T]$. That is, every two consecutive graphs in the sequence differ by at most one edge insertion or deletion. We say $\mathcal{S}$ has *maximum degree D* if G_t has maximum degree D for all $t \in [T]$.

To denote the edge difference sets, for all $t \in [T]$, we use $dE_t^- = E_{t-1} \setminus E_t$ and $dE_t^+ = E_t \setminus E_{t-1}$, as well as $dE_t = dE_t^- \cup dE_t^+ = \Delta(E_t, E_{t-1})$.

Continual release graph algorithms. Let V be a set of nodes and $T, k \in \mathbb{N}$. A *continual release graph algorithm* $\mathcal{A} : \mathcal{G}(V)^T \to (\mathbb{R}^k)^T$ receives as input[2] a dynamic graph sequence $\mathcal{S} = (G_1, \ldots, G_T)$ and, at every time step $t \in [T]$, produces an output $a_t = \mathcal{A}(G_1, \ldots, G_t) \in \mathbb{R}^k$.

Definition 2.3 (Item-level and event-level neighbors). Let V be a set of nodes and $T \in \mathbb{N}$. Let $\mathcal{S}$ and $\mathcal{S}'$ be dynamic graph sequences of length T. Let dE_t^+, dE_t^-, and dE_t be the edge difference sets as in Definition 2.2, and $dE_t^{+\prime}$, $dE_t^{-\prime}$, and dE_t' be the corresponding sets for $\mathcal{S}'$. We say $\mathcal{S}$ and $\mathcal{S}'$ are *item-level, edge-neighboring* if they differ in updates pertaining to at most one edge. That is, there is an edge $e^* \in V^2$ such that $dE_t^+ \setminus \{e^*\} = dE_t^{+\prime} \setminus \{e^*\}$ and $dE_t^- \setminus \{e^*\} = dE_t^{-\prime}$ for all $t \in [T]$.

We say $\mathcal{S}$ and $\mathcal{S}'$ are *event-level, edge-neighboring* if they differ in at most one edge insertion (or deletion) and the next deletion (or insertion) of the same edge.[1] That is, there exist an edge $e^* \in V^2$ and a time interval $[t_1, t_2]$ with $t_1 \in [1, T]$ and $t_2 \in [2, T+1]$ such that[3]

(1) $dE_t^+ = dE_t^{+\prime}$ and $dE_t^- = dE_t^{-\prime}$ for all $t \in [T] \setminus \{t_1, t_2\}$.
(2) $e^* \notin dE_t$ for all $t \in (t_1, t_2)$.
(3) a) $dE_{t_1}^+ = dE_{t_2}^- = \{e^*\}$ and $dE_{t_1}' = dE_{t_2}' = \emptyset$ or b) $dE_{t_1}^- = dE_{t_2}^+ = \{e^*\}$ and $dE_{t_1}' = dE_{t_2}' = \emptyset$,

or if the symmetric properties are true with the roles of $\mathcal{S}$ and $\mathcal{S}'$ switched.

Note that if $\mathcal{S} = (G_1, \ldots, G_T)$ and $\mathcal{S}' = (G_1', \ldots, G_T')$ are event-level edge-neighboring, then they are also item-level edge-neighboring; if they are item-level edge-neighboring, then the graphs G_t and G_t' are edge-neighboring for every $t \in [T]$.

Definition 2.4 (Item-level and event-level edge-DP). Let V be a set of nodes and $T, k \in \mathbb{N}$. Let $\mathcal{A}$ be a continual release graph algorithm $\mathcal{A} : \mathcal{G}(V)^T \to (\mathbb{R}^k)^T$. Let $\varepsilon > 0$ and $\delta \geq 0$. The algorithm $\mathcal{A}$ is item-level (respectively, event-level), (ε, δ)-edge-DP, if for all item-level (respectively, event-level) edge-neighboring dynamic graph sequences $\mathcal{S} \in \mathcal{G}(V)^T$ and $\mathcal{S}' \in \mathcal{G}(V)^T$ and all Out $\subseteq (\mathbb{R}^k)^T$:

$$\Pr\left[(\mathcal{A}(G_1, \ldots, G_t)_{t \leq T} \in \text{Out}\right] \leq e^\varepsilon \Pr\left[(\mathcal{A}(G_1, \ldots, G_t)_{t \leq T} \in \text{Out}\right] + \delta. \tag{1}$$

We use ε-edge-DP as a shorthand for $(\varepsilon, 0)$-edge-DP.

In many works on DP graph algorithms, the first step is to design algorithms that require a promise on the degree of the graph for the privacy guarantee. Jain et al. [43] call such algorithms *D-restricted*.

Definition 2.5 (D-restricted differential privacy). Let V be a set of nodes and $k, D, T \in \mathbb{N}$. Let $\mathcal{A}$ be a continual release graph algorithm $\mathcal{G}(V)^T \to (\mathbb{R}^k)^T$. Let $\varepsilon > 0$ and $\delta \geq 0$. The algorithm $\mathcal{A}$ is *D-restricted, event-level, (ε, δ)-edge differentially private (D-restricted (ε, δ)-edge-DP)*, if for all event-level edge-neighboring dynamic graph sequences $\mathcal{S}, \mathcal{S}' \in \mathcal{G}(V)^T$ of maximum degree D and every Out $\subseteq (\mathbb{R}^k)^T$:

$$\Pr\left[(\mathcal{A}(G_1, \ldots, G_t)_{t \leq T} \in \text{Out}\right] \leq e^\varepsilon \Pr\left[(\mathcal{A}(G_1, \ldots, G_t)_{t \leq T} \in \text{Out}\right] + \delta.$$

---

[2] We assume T is given to the algorithm. By a standard reduction [12], all our error bounds hold if T is not known in advance, up to polylogarithmic factors in T.
[3] To represent the case when e^* is not updated after time step t_1, we set t_2 to $T + 1$. In that case, the requirements for this time step in Item 3 should be ignored.

The definition of D-restricted item-level is analogous.

(α, β)-accuracy for graph functions. Let $f : \mathcal{G}(V) \to \mathbb{R}^k$. Let $\mathcal{S} = (G_1, \ldots, G_T)$ with $G_t \in \mathcal{G}(V)$ be a dynamic graph sequence. A continual release graph algorithm $\mathcal{A}$ is *(α, β)-accurate* for f on $\mathcal{S}$ if $\Pr\left[\max_{t \in [T]} \|\mathcal{A}(G_1, \ldots, G_t) - f(G_t)\|_\infty > \alpha\right] \le \beta$.

Definition 2.6 (Difference sequence). Let $f : \mathcal{G}(V) \to \mathbb{R}^k$. Let $\mathcal{S} = (G_1, \ldots, G_T)$ be a dynamic graph sequence on vertex set V and $G_0 = (V, \emptyset)$. The *difference sequence for f* is $(df(\mathcal{S}, t))_{t \in [T]}$, where $df(\mathcal{S}, t) = f(G_t) - f(G_{t-1})$ for $t \in [T]$.

2.2 Problem Definitions

We consider the following functions on (simple) graphs. We use $\deg(v)$ to denote the degree of a node v.

Definition 2.7 (Graph functions). Let $N \in \mathbb{N}$ and $G = (V, E)$ be a graph with $|V| = N$.
- EdgeCount: The function f_{edges} maps a graph G to the number of edges $|E|$.
- TriangleCount: The function $f_\triangle$ maps a graph G to the number of triangles in G: i.e., $f_\triangle((V, E)) = |\{\{x, v, u\} \in V^3 : (x, u), (u, v), (v, x) \in E\}|$.
- HighDegree(τ): For each $\tau \in \mathbb{N}_{\ge 0}$, the function $f_{\ge \tau}$ maps a graph G to the number of nodes in G of degree at least τ, i.e., $f_{\ge \tau}(G) = |\{v \in V : \deg(v) \ge \tau\}|$.
- DegreeList: The function f_{deg} maps a graph G to the list of node degrees: $f_{\text{deg}}(G) = (\deg(v))_{v \in V}$.
- DegreeHist: The function f_{degHist} maps a graph G to the histogram $(h_0, h_1, \ldots, h_{N-1})$ of node degrees, where $h_j = |\{v \in V : \deg(v) = j\}|$ for all $j = \{0, \ldots, N-1\}$.
- MaximumMatching: The function f_{MM} maps a graph G to the size of the maximum matching in G.
- ConnectedComponents: The function f_{CC} maps a graph G to the number of connected components in G.

We refer to Appendix A for additional privacy background.

3 Fully Dynamic Private Triangle Counting

This section presents algorithms and lower bounds for TriangleCount with event-level privacy and a lower bound for item-level privacy.

3.1 Lower Bounds for Event-Level TriangleCount via Submatrix Queries

In this section, we present the following theorem on the lower bounds on the error of fully dynamic, event-level, edge-DP algorithms for TriangleCount.

THEOREM 3.1 (LOWER BOUNDS FOR EVENT-LEVEL TriangleCount). *Let $T, N, D \in \mathbb{N}$ be sufficiently large. Then every event-level $(0.1, 0.02)$-edge-DP algorithm which is $(\alpha, 1/6)$-accurate for* Triangle-Count *on all dynamic graph sequences with N nodes and length T satisfies $\alpha = \Omega(\min(T, \sqrt[3]{TN^2}, N^2))$.*
If the accuracy condition holds instead for all dynamic graph sequences with maximum degree D and length T then $\alpha = \Omega\left(\min\left(T^{1/3}D^{2/3}, T^{1/4}D\right)\right)$.

As discussed in Section 1.2, our reduction relies on submatrix queries, defined next.

Definition 3.2 (Submatrix Query Problem). In the submatrix query problem, the data universe is $\mathcal{U} = \{0, 1\}$. A database $Y \in \mathcal{U}^M$, where $M = n^2$ for some $n \in \mathbb{N}$, can be seen as an $n \times n$ binary matrix, where one data bit corresponds to one entry in the matrix. A submatrix query on $\mathcal{U}^M$ is defined by two vectors $a, b \in \{0, 1\}^n$ and maps $Y \in \mathcal{U}^M$ to $a^T Y b = \sum_{i=1}^n \sum_{j=1}^n a[i] Y[i, j] b[j]$. Submatrix$(n, k)$ is the problem of answering k submatrix queries of the form (a, b), where $a, b \in \{0, 1\}^n$.

Example 3.3. Let $Y = (1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1)$. We have $M = 9$ and $n = 3$. To answer a submatrix query a, b, where $a = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ and $b = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$, we compute $a^T Y b = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = 3$.

Definition 3.4 (Accuracy for batch algorithms). Let $k, n \in \mathbb{N}$. Let $\mathcal{U}$ be a universe and $f : \mathcal{U}^n \to \mathbb{R}^k$ a function. A randomized algorithm $\mathcal{A}$ is (α, β)-accurate for f, if for all databases $y \in \mathcal{U}^n$ and $z = f(y)$, it outputs $a_1, \ldots, a_k$ such that $\Pr[\max_{i \in [k]} |a_i - z_i| > \alpha] \leq \beta$, where the probability is taken over the random coin flips of the algorithm.

The following lower bound is implicitly proved in [30] (Claim 3.5, Lemmas 3.8-3.10).

LEMMA 3.5 (SUBMATRIX QUERY LOWER BOUND). *There exist constants $c_1 \geq 1$ and $c_2 > 0$ such that, for all large enough $n \in \mathbb{N}$, every $(0.1, 0.02)$-differentially private and $(\alpha, 1/6)$-accurate algorithm for* Submatrix$(n, c_1 n^2)$ *satisfies $\alpha > c_2 n$.*

LEMMA 3.6 (REDUCTION FROM SUBMATRIX QUERIES TO TriangleCount). *Let $k, n \in \mathbb{N}$. Let $Y \in \{0, 1\}^{n \times n}$ be a database and $Q = \left((a^{(m)}, b^{(m)}) \right)_{m \in [k]}$, where each $a^{(m)}, b^{(m)} \in \{0, 1\}^n$, be a sequence of k submatrix queries. Then for all $w \in \mathbb{N}$, there exists a transformation from (Y, Q) to a dynamic graph sequence $(G_1, \ldots, G_T)$ of N-node graphs, where $N = 2n + w$ and $T = n^2 + 4knw$, such that:*

- *For neighboring databases $Y, Y' \in \{0, 1\}^{n \times n}$ and a query sequence Q, the transformations of (Y, Q) and (Y', Q) give graph sequences which are event-level, edge-neighboring;*
- *For all $m \in [k]$, the triangle count $f_\triangle(G_{t_m}) = w \cdot (a^{(m)})^T Y b^{(m)}$, where $t_m = n^2 + 2nw(2m - 1)$.*

PROOF. Given a database $Y \in \{0, 1\}^{n \times n}$, we construct a graph sequence on $2n + w$ nodes $x_1, \ldots, x_n$, $v_1, \ldots, v_n$, and $z_1, \ldots, z_w$, starting from graph G_0 with no edges. We begin with an initialization phase of n^2 time steps: For all $i, j \in [n]$, we insert the edge (x_i, v_j) at time $ni + j$ if $Y[i, j] = 1$; otherwise, we do not perform an update at time $ni + j$.

Next, for each $m \in [k]$, we use $4nw$ time steps to process query $(a^{(m)}, b^{(m)})$ as follows. Let $s_m = n^2 + 4nw(m - 1)$ be the time step right before we start processing this query. For all $i \in [n]$, if $a^{(m)}[i] = 1$, then, for all $\ell \in [w]$, we insert the edge (x_i, z_ℓ) at time $s_m + (i - 1)w + \ell$ and delete it at time $s_m + 2nw + (i - 1)w + \ell$. Analogously, for all $j \in [n]$, if $b^{(m)}[j] = 1$, then, for all $\ell \in [w]$, we insert the edge (v_j, z_ℓ) at time $s_m + nw + (j - 1)w + \ell$ and delete it at time $s_m + 3nw + (j - 1)w + \ell$. See Figure 1a.

Correctness. If Y and Y' differ in coordinate $(i, j) \in [n] \times [n]$, then the corresponding constructed graph sequences differ only in the presence or absence of the initial insertion of edge (x_i, v_j) at time step $in + j$. Thus, they are event-level, edge-neighboring. For each $m \in [k]$, at time step $t_m = n^2 + 4nw(m - 1) + 2nw = s_m + 2nw$, the constructed graph has all edges (x_i, z_ℓ) corresponding to the 1-entries of $a^{(m)}$ and all edges (v_j, z_ℓ) corresponding to the 1-entries of $b^{(m)}$. Further, there exists an edge (x_i, v_j) if and only if $Y[i, j] = 1$. Thus, the number of triangles at time t_m is equal to w times the number of pairs (i, j) satisfying $a^{(m)}[i] = Y[i, j] = b^{(m)}[j] = 1$. That is, $f_\triangle(G_{t_m}) = w \cdot a^{(m)} Y b^{(m)}$.

Analysis. The initialization phase uses n^2 time steps. Processing of each query uses $4nw$ time steps. Thus, $T = n^2 + 4knw$.   □

LEMMA 3.7 (REDUCTION FROM SUBMATRIX QUERIES TO TriangleCount, BOUNDED DEGREE). *Let $k, n \in \mathbb{N}$. Let $Y \in \{0, 1\}^{n \times n}$ be an input database and $Q = \left((a^{(m)}, b^{(m)}) \right)_{m \in [k]}$, where each $a^{(m)}, b^{(m)} \in \{0, 1\}^n$, be a sequence of k submatrix queries. Then for all $w, B \in \mathbb{N}$, where B divides n, there exists a transformation from (Y, Q) to a dynamic graph sequence $(G_1, \ldots, G_T)$ of N-node graphs of maximum degree $D = 2B + w$, where $N = \frac{2n^2}{B} + \frac{n^2 w}{B^2}$ and $T = n^2 + \frac{4k wn^2}{B}$, such that:*

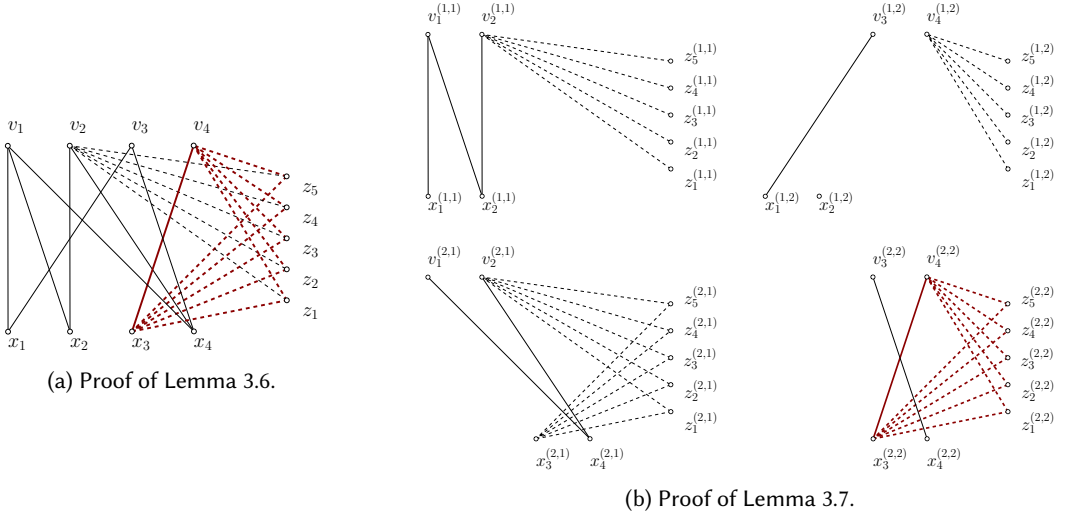(a) Proof of Lemma 3.6.

(b) Proof of Lemma 3.7.

Fig. 1. An example of the construction in the proofs of Lemma 3.6 and Lemma 3.7 for $n = 4$, $B = 2$ and $w = 5$. The input matrix is $Y = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$; the dashed lines show the edges inserted and then deleted for the query (a, b) with $a^T = (0\ 0\ 1\ 0)$ and $b^T = (0\ 1\ 0\ 1)$.

- For neighboring databases $Y, Y' \in \{0, 1\}^{n \times n}$ and a query sequence Q, the transformations of (Y, Q) and (Y', Q) give graph sequences which are event-level, edge-neighboring;
- For all $m \in [k]$, the triangle count $f_\triangle(G_{t_m}) = w \cdot a^{(m)} Y b^{(m)}$, where $t_m = n^2 + (m-1)\frac{4n^2 w}{B} + \frac{2n^2 w}{B}$.

The proof of Lemma 3.7 (deferred to Appendix B) is similar to the proof of Lemma 3.6. The main idea is to split the matrix Y into submatrices of size $B \times B$, such that the maximum degree of the graph can be reduced. Figure 1b shows an example of the construction in the proof of Lemma 3.7.

PROOF OF THEOREM 3.1. Let c_1 be as in Lemma 3.5. First, we prove the lower bound in terms of T and N. Set $w = \left\lfloor \min\left(\frac{N}{3}, \frac{T-1}{4}\right) \right\rfloor$ and $n = \left\lfloor \min\left(\frac{N}{3}, \left(\frac{T}{4wc_1+1}\right)^{1/3}\right) \right\rfloor$. For answering $k = c_1 n^2$ queries, the construction in Lemma 3.6 has $2n + w \leq N$ nodes and $n^2 + 4c_1 n^3 w \leq (4c_1 w + 1)n^3 \leq T$ time steps. We add $N - (2n + w)$ dummy nodes and $T - (n^2 + 4c_1 n^3)$ time steps without updates in order to get a dynamic graph sequence of N nodes and T time steps. Let $\mathcal{A}$ be an event-level $(0.1, 0.01)$-edge DP algorithm for TriangleCount which is $(\alpha, \frac{1}{6})$-accurate for this graph sequence. Then Lemma 3.6 gives an algorithm which, with probability at least $\frac{5}{6}$, has additive error at most $\frac{\alpha}{w}$ on all submatrix queries. Thus, by Lemma 3.5, we have $\frac{\alpha}{w} > c_2 n$ and therefore $\alpha > c_2 nw$, where c_2 is as in Lemma 3.5. Consider three cases:

(1) $T \leq N$: Then $w = \Theta(T)$ and $n = \Theta(1)$, and the lower bound is $\Omega(nw) = \Omega(T)$.
(2) $T \in (N, N^4)$: Then $w = \Theta(N)$ and $n = \Theta\left(\frac{T}{N}\right)^{1/3}$, and the lower bound is $\Omega(nw) = \Omega(T^{1/3}N^{2/3})$.
(3) $T \geq N^4$: Then $N \leq T$ and $N \leq \left(\frac{T}{N}\right)^{1/3}$, so $w = n = \Theta(N)$, and the lower bound is $\Omega(nw) = \Omega(N^2)$.

In all three cases, the lower bound is $\Omega(nw) = \Omega(\min(T, T^{1/3}N^{2/3}, N^2))$.

Now we prove the lower bound in terms of T and D. Set $w = \lfloor D/3 \rfloor$ and $n' = \left\lfloor \min\left(\sqrt[4]{\frac{T}{5c_1}}, \sqrt[3]{\frac{3T}{5Dc_1}}\right)\right\rfloor$. If $n' \le \lfloor D/3 \rfloor$, let $n = n'$, else, let n be an integer in $[n' - \lfloor D/3 \rfloor, n']$ such that $\lfloor D/3 \rfloor$ divides n. Let $B = \min(n, \lfloor D/3 \rfloor)$. For answering $k = c_1 n^2$ queries, the construction from Lemma 3.7 gives a graphs sequence of maximum degree D and length $n^2 + \frac{4c_1 n^4 w}{B} \le \frac{5c_1 n^4 w}{B} \le T$. We add extra dummy time steps to get a graph sequence of length T. Let $\mathcal{A}$ be an event-level $(0.1, 0.02)$-edge DP algorithm for TriangleCount which is $(\alpha, 1/6)$-accurate for this graph sequence. Then Lemma 3.7 gives a $(0.1, 0.01)$-DP algorithm with additive error at most $\frac{\alpha}{w}$ on all submatrix queries. Then Lemma 3.5 implies $\alpha > c_2 w n$, where c_2 is as in Lemma 3.5. Thus, $\alpha = \Omega\left(\min\left(T^{1/4}D, T^{1/3}D^{2/3}\right)\right)$. $\qquad\square$

3.2 Event-Level Private Algorithm for Triangle Count

In this section, we present our fully dynamic event-level, edge-DP algorithm for triangle counting. Our algorithmic results on this problem are summarized in the following theorem.

THEOREM 3.8 (UPPER BOUND FOR EVENT-LEVEL TriangleCount). *For all $\varepsilon, \delta, \beta \in (0, 1)$ and $T, N \in \mathbb{N}$, there exists an event-level (ε, δ)-edge-DP algorithm which is (α, β)-accurate for* TriangleCount *on all dynamic graph sequences with N nodes and length T, where*

$$\alpha = O\left(\min\left(\frac{\sqrt{TN}}{\varepsilon}(\ln\frac{1}{\delta})\log^{3/2}\frac{T}{\beta}, N\left(\frac{T}{\varepsilon^2}(\ln\frac{1}{\delta})\ln\frac{T}{\beta}\right)^{1/3}, N^3\right)\right).$$

Further, for all dynamic graph sequences with degree at most D, for $D \in \mathbb{N}$ and $D \le N$, the algorithm satisfies $\alpha = O\left(\min\left(\sqrt{TD}, DT^{1/3}, ND^2\right) \cdot \text{poly}\left(\frac{1}{\varepsilon}\log\frac{TN}{\beta\delta}\right)\right)$.

The minimum expressions in the bounds in Theorem 3.8 are the result of selecting the best algorithm for each setting of parameters. Later (in Corollary 5.3), we will present item-level (and thus also event-level) algorithms for triangle counting. The algorithm in this section, whose performance is summarized in Theorem 3.9, has better error than the item-level edge-DP algorithm for the same problem in Corollary 5.3 when the number of nodes is large: specifically, $N \ge T^{1/3}$.

THEOREM 3.9 (EVENT-LEVEL DP TRIANGLE COUNT). *For all $\varepsilon, \delta, \beta \in (0, 1)$ and $T, N, D \in \mathbb{N}$, there exists an event-level (ε, δ)-edge-DP algorithm which is (α, β)-accurate for* TriangleCount *on all dynamic graph sequences with N nodes, maximum degree D and length T, where $\alpha = O(\sqrt{TD} \cdot \text{poly}(\frac{1}{\varepsilon}\log\frac{TN}{\beta\delta}))$.*

Theorem 3.9 follows from Lemma 3.10 that gives a D-restricted version of the algorithm and Theorem 4.1 that gives a transformation from D-restricted to general algorithms. The D-restricted version of the algorithm is based on running a binary tree mechanism on the difference sequence of $f_\triangle$ (see Definition 2.6), combined with a careful analysis of the L_2-sensitivity of the partial sums used by the mechanism. The full proof is given in Appendix B.

LEMMA 3.10 (D-RESTRICTED PRIVATE TRIANGLE COUNT). *Let $\varepsilon, \delta, \beta \in (0, 1)$ and $T, D \in \mathbb{N}$. There exists an event-level D-restricted (ε, δ)-edge-DP algorithm which is (α, β)-accurate for* TriangleCount *on all dynamic graph sequences of maximum degree D and length T, where $\alpha = O\left(\frac{1}{\varepsilon}\sqrt{TD\ln\frac{1}{\delta}}\log^{3/2}\frac{T}{\beta}\right)$.*

3.3 Lower Bound for Item-Level TriangleCount

In this section, we present the following theorem that gives a lower bound on triangle counting with item-level edge-DP.

THEOREM 3.11 (ITEM-LEVEL LOWER BOUND FOR TRIANGLE COUNT). *For all $\varepsilon \in (0, 1]$, $\delta \in [0, 1)$, and sufficiently large $T, N \in \mathbb{N}$, every item-level, (ε, δ)-edge DP algorithm for $f_\triangle$ which is $(\alpha, \frac{1}{3})$-accurate on all dynamic graph sequences with N nodes and length T satisfies*

(1) *if $\delta > 0$ and $\delta = o(\frac{1}{T})$, then $\alpha = \Omega\left(\min\left(\frac{T^{7/6}}{(\varepsilon \log T)^{1/3}}, \frac{NT^{1/3}}{\varepsilon^{2/3} \log^{2/3} T}, N^3\right)\right)$.*

(2) *if $\delta = 0$, then $\alpha = \Omega\left(\min\left(\frac{T^{5/4}}{\varepsilon^{1/4}}, N\sqrt{\frac{T}{\varepsilon}}, N^3\right)\right)$.*

To prove the theorem, we give a reduction from the *1-way marginals problem* in the batch model, defined next.

Definition 3.12 (1-way marginals). Let $d, n \in \mathbb{N}$ and let $\mathcal{U} = \{0, 1\}^d$. In the Marginals(n, d) problem, the input is a database $Y = (Y_1, \ldots, Y_n) \in \mathcal{U}^n$, and the goal is to compute the vector $(q_1(Y), \ldots, q_d(Y))$, where $q_j(Y) = \frac{1}{n} \sum_{i=1}^{n} Y_i[j]$ for all $j \in [d]$.

A randomized algorithm $\mathcal{A}$ is (α, β)-accurate for Marginals(n, d), if for all databases $Y \in \{\{0, 1\}^d\}^n$, it outputs $a_1, \ldots, a_k$ such that $\Pr[\max_{j \in [k]} |a_j - q_j(Y)| > \alpha] \leq \beta$, where the probability is taken over the random coin flips of the algorithm.

The following lower bound is taken from [42], which in turn is based on [11, 37].

LEMMA 3.13 (LOWER BOUND FOR MARGINALS [11, 37, 42]). *For all $\varepsilon \in (0, 1]$, $\delta \in [0, 1)$, $\alpha \in (0, 1)$, and $d, n \in \mathbb{N}$, every (ε, δ)-DP algorithm that is $(\alpha, \frac{1}{3})$-accurate for Marginals(n, d) satisfies:*

- *if $\delta > 0$ and $\delta = o\left(\frac{1}{n}\right)$, then $\alpha = \Omega\left(\frac{\sqrt{d}}{n\varepsilon \log d}\right)$.*

- *if $\delta = 0$, then $\alpha = \Omega\left(\frac{d}{n\varepsilon}\right)$.*

The reduction from the 1-way marginals problem to item-level DP triangle count is given in Lemma B.1 in the appendix. Due to space constraints and its similarity to the proof of Theorem 6.9, the complete proof of Theorem 3.11 is deferred to the full version [59].

4 Transformation from Degree-Restricted to Private

In this section, we give our general transformation in the event-level setting from a D-restricted edge-DP algorithm with an error bound that depends on D to an edge-DP algorithm for *all* dynamic graph sequences which has an error bound depending on the maximum degree of the sequence. The error bounds are the same as for the D-restricted case, up to polynomial factors in $\frac{1}{\varepsilon} \log(\frac{TN}{\beta\delta})$.

THEOREM 4.1. *Let $T, N, k \in \mathbb{N}$ and f be a function $\mathcal{G}(V) \to \mathbb{R}^k$. Let $\alpha : \mathbb{N} \times (0, 1) \to \mathbb{R}^+$ be a function of the degree D and the error probability β, such that $\alpha(D, \beta)$ is nondecreasing in D and nonincreasing in β. If for all $\beta \in (0, 1)$, $\varepsilon > 0$, $D \in \mathbb{N}$, and $\delta \in [0, 1)$, there is an event-level D-restricted (ε, δ)-edge-DP algorithm $\mathcal{A}_D$ which is $(\alpha(D, \beta), \beta)$-accurate for f on all dynamic graph sequences with N nodes, maximum degree D and length T, then for all $D' \in \mathbb{N}$, $\varepsilon' > 0$, $\delta' \in (0, 1)$ and $\beta' \in (0, 1)$, there exists an event-level (ε', δ')-edge DP algorithm for f which is $\left(\alpha\left(D'', \min\left(\frac{\delta'}{2+2e^{\varepsilon'}}, \frac{\beta'}{2+\log N}\right)\right), \beta'\right)$-accurate on all dynamic graph sequences with N nodes, maximum degree D', and length T, where $D'' = O(D' + \frac{\log T \cdot \log N}{\varepsilon'} \log \frac{TN \log N}{\beta'\delta'})$. The value D' does not need to be given to the algorithm.*

As one of the main tools, we first give an algorithm for accurately estimating the degree list of a dynamic graph sequence. It is based on using a continual counting algorithm on the difference sequence, similar to strategies in [33]. The proof of the lemma is deferred to the full version [59].

LEMMA 4.2 (ALGORITHM FOR DegreeList). *For all $T, N, D \in \mathbb{N}$, $\varepsilon > 0$, and $\beta \in (0, 1)$, there exists an event-level ε-edge-DP algorithm for DegreeList which is (α, β)-accurate on all dynamic N-node graph sequences of length T, where $\alpha = O\left(\frac{\log T}{\varepsilon} \log \frac{TN}{\beta}\right)$.*

The main idea in the proof of Theorem 4.1 is to keep a running estimate of the maximum degree, and run the corresponding degree restricted mechanism, until the maximum degree increases significantly. We then re-initialize a new restricted mechanism with a higher degree bound. The full proof is given in Appendix C.

5 Algorithms Based on Recomputing at Regular Intervals

In this section, we collect error bounds which follow from the *recomputing strategy*, i.e., recomputing the answer every fixed number of time steps and returning the most recently computed answer at ever time step. First, we give Theorem 5.1 on item-level algorithms for functions with known sensitivity. It generalizes Theorems E.2 and E.5 in [42] (stated there for functions of sensitivity 1) to higher-dimensional functions and adapts them for graph functions. The proof is given in Appendix D.

THEOREM 5.1 (RELEASING LOW SENSITIVITY FUNCTIONS). *Let V be a set of nodes, $r > 0$ be a range parameter, and $k \in \mathbb{N}$ be the dimension. Let f be a function $f : \mathcal{G}(V) \to [0, r]^k$ of L_p-sensitivity Δ_p (w.r.t. the edge-neighboring metric in static graphs) for all $p \in \{1, 2\}$. For all $\varepsilon > 0, \delta \in [0, 1)$, $\beta \in (0, 1)$, there exists an item-level (ε, δ)-edge-DP algorithm which is (α, β)-accurate for estimating f on all dynamic graph sequences of length $T \in \mathbb{N}$, satisfying the following accuracy guarantee:*

(1) *if $\delta = 0$, then $\alpha = O\left(\min\left(\Delta_1 \cdot \sqrt{\frac{T}{\varepsilon} \ln \frac{Tk}{\beta}}, r\right)\right)$;*

(2) *if $\delta > 0$, then $\alpha = O\left(\min\left(\Delta_2 \cdot \sqrt[3]{\frac{T}{\varepsilon^2}(\ln \frac{Tk}{\beta}) \ln \frac{1}{\delta}}, r\right)\right)$.*

Theorem 5.1 immediately yields the following corollary for functions with constant sensitivity, including EdgeCount, HighDegree, MaximumMatching, MinCut, ConnectedComponents, DegreeHist, DegreeList.

COROLLARY 5.2 (ITEM-LEVEL ALGORITHMS). *Let C be the class of functions of constant sensitivity from graphs to $[0, N]$ (that includes $f_{\geq \tau} \forall \tau \in \mathbb{N}$, f_{MM}, f_{MC}, and f_{CC}) and C_N be the class of functions from graphs to real-valued vectors of length $O(N)$ with entries in $[0, N]$ (that includes DegreeHist and DegreeList). For all $\varepsilon > 0, \delta \in [0, 1), \beta \in (0, 1), \tau \in \mathbb{N}$ and $T \in \mathbb{N}$ and $f \in C \cup C_N$, there exists an item-level (ε, δ)-edge-DP algorithm which is (α, β)-accurate for f on all dynamic graph sequences of length T such that $\alpha = O(\min(\alpha', N^2))$ for f_{edges} and $\alpha = O(\min(\alpha', N))$ for all $f \in C \cup C_N$, where*

(1) *if $\delta = 0$, then $\alpha' = \sqrt{\frac{T}{\varepsilon} \ln \frac{T}{\beta}}$ for f_{edges} and all $f \in C$, and $\alpha' = \sqrt{\frac{T}{\varepsilon} \ln \frac{TN}{\beta}}$ for all $f \in C_N$.*

(2) *if $\delta > 0$, then $\alpha' = \sqrt[3]{\frac{T}{\varepsilon^2}(\ln \frac{T}{\beta}) \ln \frac{1}{\delta}}$ for f_{edges} and all $f \in C$, and $\alpha' = \sqrt[3]{\frac{T}{\varepsilon^2}(\ln \frac{TN}{\beta}) \ln \frac{1}{\delta}}$ for all $f \in C_N$.*

By Theorem 5.1, there exists a D-restricted item-level (and therefore also event-level), ε-edge-DP algorithm for fully dynamic triangle count which is (α, β)-accurate with $\alpha = O\left(D\sqrt{\frac{T}{\varepsilon} \ln \frac{T}{\beta}}\right)$, and a D-restricted, event-level, (ε, δ)-edge-DP algorithm for fully dynamic triangle count which is (α, β)-accurate with $\alpha = O\left(D \cdot \sqrt[3]{\frac{T}{\varepsilon^2}(\ln \frac{T}{\beta}) \ln \frac{1}{\delta}}\right)$. (Observe that when D is set to N, every D-restricted algorithm becomes edge-DP because the promise on the degree becomes vacuous.) Together with Theorem 4.1, we get the following corollary.

COROLLARY 5.3 (TRIANGLE COUNT, RECOMPUTING BOUND). *Let $\varepsilon > 0$ and $\delta, \beta \in (0, 1)$. Let $T, N, D \in \mathbb{N}$. There exist an event-level (ε, δ)-edge-DP algorithm which is (α, β)-accurate for TriangleCount on all dynamic graph sequences with N nodes, maximum degree D and length T, where $\alpha = O\left(T^{1/3}D \cdot \text{poly}\left(\frac{1}{\varepsilon} \log \frac{TN}{\delta\beta}\right)\right)$. There exist an item-level (ε, δ)-edge-DP algorithm which is (α, β)-accurate for TriangleCount on all dynamic graph sequences with N nodes and length T, where*
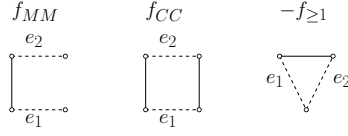
Fig. 2. 2-edge distinguishing gadgets for f_{MM}, f_{CC} and $-f_{\geq 1}$ of constant size and weight. Note that for f_{MM}, we have $e_1, e_2 \notin E'$, and for f_{CC} and $-f_{\geq 1}$, we have $e_1, e_2 \in E'$.

(1) if $\delta = 0$, then $\alpha = O\left(\min\left(\min(T, N) \cdot \sqrt{\frac{T}{\varepsilon} \ln \frac{T}{\beta}}, N^3\right)\right)$.

(2) if $\delta > 0$, then $\alpha = O\left(\min\left(\min(T, N) \cdot \sqrt[3]{\frac{T}{\varepsilon^2}(\ln \frac{T}{\beta}) \ln \frac{1}{\delta}}, N^3\right)\right)$.

6 A Lower Bound Framework for DP Fully Dynamic Algorithms

We start this section by defining terminology and gadgets used in our lower bound framework. In Section 6.1 (respectively, Section 6.2), we present a general theorem for proving event-level (respectively, item-level) lower bounds for fully dynamic graph algorithms.

Let $\mathcal{G}$ denote the set of all graphs, and let $f : \mathcal{G} \to \mathbb{R}$ be a real-valued graph function. Our framework applies to *additive graph functions*, which have a *distinguishing gadget*, defined next.

Definition 6.1 (Additive graph function). A function f is *additive* if for each graph G consisting of the connected components $C_1, \ldots, C_m$, we have $f(G) = \sum_{i \in [m]} f(C_i)$.

Definition 6.2 (Distinguishing gadget). Let $H = (V', E')$ be a graph and $e_1, e_2 \in V' \times V'$. Set $n_g = |V'|$ and $m_g = |E'|$. The pair (H, e_1) is called a *1-edge distinguishing gadget* for f of size (n_g, m_g) and weight w if either

- $e_1 \notin E'$ and $f((V', E' \cup \{e_1\})) = f(H) + w$ or
- $e_1 \in E'$ and $f((V', E' \setminus \{e_1\})) = f(H) + w$.

The triple (H, e_1, e_2) is called a *2-edge distinguishing gadget* for f of size (n_g, m_g) and weight w, if either

- $e_1, e_2 \notin E'$ and $f((V', E' \cup \{e_1, e_2\})) = f(H) + w$ and $f((V', E' \cup \{e_1\})) = f((V', E' \cup \{e_2\})) = f(H)$ or
- $e_1, e_2 \in E'$ and $f((V', E' \setminus \{e_1, e_2\})) = f(H) + w$ and $f((V', E' \setminus \{e_1\})) = f((V', E' \setminus \{e_2\})) = f(H)$.

If there exists a distinguishing gadget for f, we say that f has a distinguishing gadget. Examples of distinguishing gadgets are illustrated in Figure 2.

LEMMA 6.3. *If f has a 2-edge distinguishing gadget of size (n_g, m_g) and weight w, then f also has a 1-edge distinguishing gadget of size $(n_g, m_g + 1)$ or $(n_g, m_g - 1)$ and weight w.*

6.1 Framework for Event-Level Lower Bounds

In this section, we state and prove Theorem 6.4 that encapsulates our lower bound framework for event-level DP. We also apply our framework to specific problems, stating the resulting lower bounds in Corollary 6.8.

THEOREM 6.4 (EVENT-LEVEL LOWER BOUND FOR 2-EDGE DISTINGUISHING). *Let $f : \mathcal{G}(V) \to \mathbb{R}$ be an additive function with a 2-edge distinguishing gadget of size (n_g, m_g) and weight w for some $n_g, m_g, w \in \mathbb{N}$. Then, for all sufficiently large $T, N \in \mathbb{N}$, every event-level, $(1, 1/3)$-edge DP algorithm which is $(\alpha, 0.01)$-accurate for f on all dynamic graph sequences with N nodes and length T has error $\alpha = \Omega\left(w \cdot \min\left(T^{1/4}, \sqrt{\frac{T}{m_g}}, \sqrt{\frac{N}{n_g}}\right)\right)$.*

We prove Theorem 6.4 via a reduction to the inner product problem, defined next.

Definition 6.5 (Inner product). In the inner product problem, the data universe is $\mathcal{U} = \{0, 1\}$. An inner product query on $\mathcal{U}^n$ is defined by a vector $q \in \{0, 1\}^n$ and maps $y \in \mathcal{U}^n$ to $q \cdot y$. The problem InnerProd(n, k) is the problem of answering k inner product queries $q^{(1)}, \ldots, q^{(k)} \in \{0, 1\}^n$.

The following formulation of the lower bound is based on [41], which is in turn based on [17, 20, 26, 51].

LEMMA 6.6 (INNER PRODUCT LOWER BOUND). *There exist constants $c_1 \geq 1$ and $c_2 > 0$ such that, for all large enough $n \in \mathbb{N}$, every $(1, 1/3)$-DP and $(\alpha, 0.01)$-accurate algorithm for InnerProd$(n, c_1 n)$ satisfies $\alpha > c_2 \sqrt{n}$.*

LEMMA 6.7 (REDUCTION FROM INNER PRODUCT FOR EVENT-LEVEL). *Let $n, k \in \mathbb{N}$. Let $y \in \{0, 1\}^n$ and $Q = (q^{(i)})_{i \in [k]}$, where each $q^{(i)} \in \{0, 1\}^n$, be a sequence of k inner product queries. Then for all $n_g, m_g, w \in \mathbb{N}$ and each additive function $f : \mathcal{G}(V) \to \mathbb{R}$ with a 2-edge distinguishing gadget of size (n_g, m_g) and weight w, there exists a transformation from (y, Q) to a dynamic graph sequence $(G_1, \ldots, G_T)$ of N-node graphs, where $T = (m_g + 2k)n$ and $N = n_g n$, such that*

- *For neighboring $y, y' \in \{0, 1\}^n$ and the same query sequence Q, the transformations of (y, Q) and (y', Q) give event-level, edge-neighboring graph sequences;*
- *Let $t_0 = (m_g + 1)n$. For all $\ell \in [k]$ and $t_\ell = (m_g + 2\ell)n$, we have $f(G_{t_\ell}) - f(G_{t_0}) = wy \cdot q^{(\ell)}$.*

PROOF. Let $H = (V', E')$ and (H, e_1, e_2) be a 2-edge distinguishing gadget of size (n_g, m_g) of weight w for f. W.l.o.g. assume $e_1, e_2 \notin E'$. We define a graph sequence on $n \cdot n_g$ nodes as follows: In an *initialization phase*, we build n copies of H, denoted $H^{(1)}, \ldots, H^{(n)}$, using $m_g n$ time steps. For a vertex pair e in H and $j \in [n]$, let $e^{(j)}$ denote the copy of e in $H^{(j)}$. For every $j \in [n]$, if $y[j] = 1$, insert edge $e_1^{(j)}$ at time step $m_g n + j$ (otherwise, do nothing at that time step).

For each query vector $q^{(\ell)}$, where $\ell \in [k]$, we first have an *insertion phase*, where for all $j \in [n]$, if $q^{(\ell)}[j] = 1$, then we insert edge $e_2^{(j)}$ into $H^{(j)}$ at time $t_0 + 2(\ell - 1)n + j$; otherwise, we do nothing at that time step. For all but the last query, the insertion phase is followed by the *deletion phase*, where the edges that were inserted in the insertion phase are deleted over n time steps.

Correctness. If $y, y' \in \{0, 1\}^n$ are neighboring input vectors differing in coordinate i^*, then the resulting graph sequences for (y, Q) and (y', Q) differ only in the absence or presence of the initial insertion of $e_1^{(i^*)}$. Thus, they are event-level, edge-neighboring. The insertion phase for the ℓth query ends at time step $t_\ell = (m_g + 2\ell)n$. At that time step, for all $i \in [n]$, the copy $H^{(i)}$ contains both $e_1^{(i)}$ and $e_2^{(i)}$ if and only if $y[i] = q[i] = 1$. Thus, $f(G_{t_\ell}) = f(G_{t_0}) + w \cdot y \cdot q$.

Analysis. The initialization phase uses $(m_g + 1)n$ time steps. Each of the k insertion phases and $k - 1$ deletion phases uses n time steps. Thus, $T = (m_g + 2k)n$. There are $n \cdot n_g$ nodes in the graph. □

PROOF OF THEOREM 6.4. Let c_1, c_2 be as in Lemma 6.6. Given T and N, set $n = \min\left(\sqrt{\frac{T}{4c_1}}, \frac{T}{2m_g}, \frac{N}{n_g}\right)$. Then for each $y \in \{0, 1\}^n$ and k-query sequence Q, where $k = c_1 n$, the transformation in Lemma 6.7 gives a sequence with at most $2c_1 n^2 + m_g n \leq T$ time steps and at most N nodes. We can add dummy nodes and time steps to get a sequence of exactly T time steps and N nodes. If we can estimate $f(G_t)$ up to error α at all time steps with probability at least $1 - 0.01$, then with the same probability, we can estimate $y \cdot q^{(m)} = \frac{f(G_{t_w}) - f(G_{t_0})}{w}$ up to error $\frac{2\alpha}{w}$ for all $m \in [k]$. Thus, $\alpha \geq w \cdot \sqrt{n} \cdot \frac{c_2}{2} = \Omega\left(w \cdot \min\left(T^{1/4}, \sqrt{\frac{T}{m_g}}, \sqrt{\frac{N}{n_g}}\right)\right)$. □

6.1.1 Event-Level Lower Bounds for Specific Problems. Theorem 6.4 yields the following corollary for MaximumMatching, ConnectedComponents, HighDegree(τ), and DegreeHist.

COROLLARY 6.8 (EVENT-LEVEL LOWER BOUNDS). *Let $\tau \in \mathbb{N}$ and $f \in \{f_{MM}, f_{CC}, f_{\geq \tau}, f_{\text{degHist}}\}$. Then, for all sufficiently large $T, N \in \mathbb{N}$, every event-level, $(1, 1/3)$-edge DP algorithm which is $(\alpha, 0.01)$-accurate for f on all dynamic graph sequences with N nodes and length T has error $\alpha = \Omega\left(\min\left(T^{1/4}, \sqrt{N}\right)\right)$.*

PROOF. Using the gadgets illustrated in Figure 2 in Theorem 6.4 gives the stated bound for MaximumMatching, ConnectedComponents, and HighDegree(1). The lower bound for DegreeHist is inherited from the lower bound on HighDegree(1): one of the entries in the degree histogram of a graph $G = (V, E)$ is the number of isolated nodes, which is $|V| - f_{\geq 1}(G)$.

Finally, the lower bound for $f_{\geq \tau}$ for $\tau \geq 2$ is inherited from the lower bound for $f_{\geq 1}$. Fix $\tau \geq 2$. We can reduce from fully dynamic $f_{\geq 1}$ to fully dynamic $f_{\geq \tau}$ as follows. Let $\mathcal{S} = (G_1, \ldots, G_T)$ be a dynamic graph sequence on vertex set V of size $N \geq 2$. We construct a new graph sequence $\mathcal{S}_\tau$ on nodes $V \cup U$, where U is disjoint from V and $|U| = \tau - 1$. We initialize $\mathcal{S}_\tau$ by inserting a complete bipartite graph on $U \times V$ and a complete graph on U. This takes $t_0 = (\tau - 1)(N + \frac{\tau - 2}{2})$ time steps. Then we proceed with the same updates on V as in $\mathcal{S}$. The resulting sequence $\mathcal{S}_\tau$ is on graphs with $N_\tau = N + \tau$ nodes and has length $T_\tau = t_0 + T$. Let $\mathcal{S}_\tau = (H_1, \ldots, H_{T_\tau})$.

In H_{t_0} (at the end of initialization), all vertices from U have degree $\tau + |V| - 2 \geq \tau$, and all vertices from V have degree $\tau - 1$. For all $t \in [T]$, we have $f_{\geq 1}(G_t) = f_{\geq \tau}(H_{t_0 + T}) - (\tau - 1)$. This transformation preserves event-level (as well as item-level) edge-neighboring relationship. Since $N_\tau = \Theta(N)$ and $T_\tau = \Theta(T)$, the additive error lower bound for $f_{\geq 1}$ applies to $f_{\geq \tau}$. □

6.2 Framework for Item-Level Lower Bounds

In this section, we state Theorem 6.9 that encapsulates our lower bound framework for item-level DP. We also apply our framework to specific problems in Section 6.2.1.

THEOREM 6.9 (ITEM-LEVEL LOWER BOUND FOR 1-EDGE DISTINGUISHING). *Let $f : \mathcal{G}(V) \to \mathbb{R}$ be an additive function with a 1-edge distinguishing gadget of size (n_g, m_g) and weight w for some $n_g, m_g, w \in \mathbb{N}$. Then, for all $\varepsilon \in (0, 1]$, $\delta \in [0, 1)$, and sufficiently large $T, N \in \mathbb{N}$, every item-level, (ε, δ)-edge DP algorithm which is $(\alpha, \frac{1}{3})$-accurate for f on all dynamic graph sequences with N nodes and length T satisfies*

(1) *if $\delta > 0$ and $\delta = o(\frac{1}{T})$, then $\alpha = \Omega\left(w \cdot \min\left(\frac{T^{1/3}}{\varepsilon^{2/3} \log^{2/3} T}, \frac{T}{m_g}, \frac{N}{n_g}\right)\right)$.*

(2) *if $\delta = 0$, then $\alpha = \Omega\left(w \cdot \min\left(\sqrt{\frac{T}{\varepsilon}}, \frac{T}{m_g}, \frac{N}{n_g}\right)\right)$.*

We prove Theorem 6.9 by reducing from the Marginals problem (Definition 3.12).

LEMMA 6.10 (REDUCTION FROM MARGINALS FOR ITEM-LEVEL). *Let $n, d \in \mathbb{N}$ and $Y \in \{\{0, 1\}^d\}^n$. Then for all $n_g, m_g, w \in \mathbb{N}$ and each additive function $f : \mathcal{G}(V) \to \mathbb{R}$ with a 1-edge distinguishing gadget of size (n_g, m_g) and weight w, there exists a transformation from Y to a dynamic graph sequence $(G_1, \ldots, G_T)$ of N-node graphs, where $T = (m_g + 2d)n$ and $N = n_g n$, such that*

- *The trasformations of neighboring $Y, Y' \in \{\{0, 1\}^d\}^n$ give item-level, edge-neighboring graph sequences;*
- *Let $t_0 = (m_g + 1)n$. For all $j \in [d]$ and $t_j = (m_g + 2j)n$, we have $f(G_{t_j}) - f(G_0) = w \sum_{i=1}^{n} Y_i[j]$.*

The proofs of Lemma 6.10 and Theorem 6.9 (deferred to Appendix E) are similar to the proofs of Lemma 6.7 and Theorem 6.4.

6.2.1 Item-Level Lower Bounds for Specific Problems. Using the 2-edge distinguishing gadgets from Figure 2 together with Lemma 6.3 on converting 2-edge distinguishing gadgets to 1-edge distinguishing, we immediately get the following corollary for MaximumMatching, ConnectedComponents,

and HighDegree(1) from Theorem 6.9. By the same reasoning as in the proof of Corollary 6.8, this lower bound also applies to DegreeHist and to HighDegree(τ) for $\tau \geq 2$.

COROLLARY 6.11 (ITEM-LEVEL LOWER BOUNDS). *Let $\tau \in \mathbb{N}$ and $f \in \{f_{MM}, f_{CC}, f_{\geq \tau}, f_{degHist}\}$. Then, for all $\varepsilon \in (0,1]$, $\delta \in [0,1)$, and sufficiently large $T, N, D \in \mathbb{N}$, every item-level, (ε, δ)-edge DP algorithm which is $(\alpha, \frac{1}{3})$-accurate for f on all dynamic graph sequences with N nodes and length T satisfies*

(1) *if $\delta > 0$ and $\delta = o(\frac{1}{T})$, then $\alpha = \Omega\left(\min\left(\frac{T^{1/3}}{\varepsilon^{2/3} \log^{2/3} T}, T, N\right)\right)$.*

(2) *if $\delta = 0$, then $\alpha = \Omega\left(\min\left(\sqrt{\frac{T}{\varepsilon}}, T, N\right)\right)$.*

Next, we prove the item-level lower bound for EdgeCount. It cannot be obtained as a direct corollary of Theorem 6.9 because the gadgets we use for this problem overlap, but otherwise it follows a similar proof strategy and yields a similar-looking (albeit not exactly the same) bound.

THEOREM 6.12 (ITEM-LEVEL LOWER BOUND FOR EDGE COUNT). *For all $\varepsilon \in (0,1]$, $\delta \in [0,1)$, and sufficiently large $T, N \in \mathbb{N}$, every item-level, (ε, δ)-edge DP algorithm for f_{edges} which is $(\alpha, \frac{1}{3})$-accurate on all dynamic graph sequences with N nodes and length T satisfies*

(1) *if $\delta > 0$ and $\delta = o(\frac{1}{T})$, then $\alpha = \Omega\left(\min\left(\frac{T^{1/3}}{\varepsilon^{2/3} \log^{2/3} T}, N^2\right)\right)$.*

(2) *if $\delta = 0$, then $\alpha = \Omega\left(\min\left(\sqrt{\frac{T}{\varepsilon}}, N^2\right)\right)$.*

As in the general framework of Theorem 6.9, we reduce from Marginals (recall Definition 3.12).

LEMMA 6.13 (REDUCTION FROM MARGINALS TO EdgeCount). *Let $n, d \in \mathbb{N}$ and $Y \in \{\{0,1\}^d\}^n$. Then there exists a transformation from Y to a dynamic graph sequence $(G_1, \ldots, G_T)$ of N-node graphs, where $T = 2nd$ and $N = \lceil \sqrt{n} \rceil$, such that*

- *The transformations of neighboring $Y, Y' \in \{\{0,1\}^d\}^n$ give item-level, edge-neighboring graph sequences;*
- *For all $j \in [d]$, we have $f_{edges}(G_{t_j}) = \sum_{i=1}^n Y_i[j]$, where $t_j = (2j-1)n$.*

PROOF. We define a set V of $\lceil \sqrt{n} \rceil$ nodes. V has at least n node pairs. We give them an arbitrary order $e_1, \ldots, e_{\lceil \sqrt{n} \rceil^2}$. In particular, there exist the node pairs $e_1, \ldots, e_n$.

For all $j \in [d]$ and all $i \in [n]$, if $Y_i[j] = 1$, then we insert e_i at time $2(j-1)n + i$; otherwise, we do nothing at that time step. For all $j \in [d-1]$ and all $i \in [n]$, we delete e_i at time $(2j-1)n + i$.

Correctness. If Y and Y' are neighboring input databases differing in row i^*, then the resulting graph sequences for Y and Y' differ only in insertions and deletions related to e_i. Thus, they are item-level, edge-neighboring. At time step $t_j = (2j-1)n$, the graph G_{t_j} includes an edge e_i if and only if $Y_i[j] = 1$. Thus, the total edge count is equal to $\sum_{i=1}^n Y_i[j]$.   □

The proof of Theorem 6.12 (deferred to the full version [59]) is similar to the proof of Theorem 6.9.

REMARK 1. DegreeList *inherits lower bounds from* EdgeCount, *except that N gets replaced with $\sqrt{N}$. This holds both for event-level and item-level edge DP. This is because there is a direct reduction from fully dynamic f_{edges} to fully dynamic* DegreeList *that converts each dynamic graph sequences $\mathcal{S}$ on N nodes of length T to a dynamic graph sequences $\mathcal{S}'$ on $\binom{N}{2} + 1$ nodes of the same length. Given a vertex set V of $\mathcal{S}$, the vertex set V' of $\mathcal{S}'$ is defined as $\{u\} \cup \{v_e : e \in V \times V\}$. Then each update on an edge e in $\mathcal{S}$ is replaced with the same time of update on the edge (u, v_e) in $\mathcal{S}'$. At every time step, the degree of u in $\mathcal{S}'$ is equal to the edge count in $\mathcal{S}$. Moreover, this reduction preserves event-level and item-level neighbor relationships.*

Acknowledgments

References

[1] Faraz Ahmed, Alex X. Liu, and Rong Jin. 2020. Publishing social network graph eigenspectrum with privacy guarantees. *IEEE Transactions on Network Science and Engineering* 7, 2 (2020), 892–906. https://doi.org/10.1109/TNSE.2019.2901716

[2] Raman Arora and Jalaj Upadhyay. 2019. On Differentially Private Graph Sparsification and Applications. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 13378–13389. https://proceedings.neurips.cc/paper/2019/hash/e44e875c12109e4fa3716c05008048b2-Abstract.html

[3] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2012. The Johnson-Lindenstrauss transform itself preserves differential privacy. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*. 410–419. https://doi.org/10.1109/FOCS.2012.67

[4] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2013. Differentially private data analysis of social networks via restricted sensitivity. In *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, Robert D. Kleinberg (Ed.). ACM, 87–96. https://doi.org/10.1145/2422436.2422449

[5] Jeremiah Blocki, Elena Grigorescu, and Tamalika Mukherjee. 2022. Privately Estimating Graph Parameters in Sublinear Time. In *Proceedings, International Colloquium on Automata, Languages and Programming (ICALP)*, Vol. 229. 26:1–26:19. https://doi.org/10.4230/LIPIcs.ICALP.2022.26

[6] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. 2005. Practical privacy: the SuLQ framework. In *Proceedings of the Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 13-15, 2005, Baltimore, Maryland, USA*, Chen Li (Ed.). ACM, 128–138. https://doi.org/10.1145/1065167.1065184

[7] Jean Bolot, Nadia Fawaz, S. Muthukrishnan, Aleksandar Nikolov, and Nina Taft. 2013. Private Decayed Predicate Sums on Streams. In *International Conference on Database Theory (ICDT)*. 284–295.

[8] Christian Borgs, Jennifer T. Chayes, and Adam D. Smith. 2015. Private Graphon Estimation for Sparse Graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*. 1369–1377.

[9] Christian Borgs, Jennifer T. Chayes, Adam D. Smith, and Ilias Zadik. 2018. Revealing Network Structure, Confidentially: Improved Rates for Node-Private Graphon Estimation. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*. 533–543. https://doi.org/10.1109/FOCS.2018.00057

[10] Mark Bun and Thomas Steinke. 2016. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. In *Proceedings, Theory of Cryptography Conference (TCC)*, Vol. 9985. 635–658.

[11] Mark Bun, Jonathan Ullman, and Salil Vadhan. 2018. Fingerprinting Codes and the Price of Approximate Differential Privacy. *SIAM J. Comput.* 47, 5 (2018), 1888–1938.

[12] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. 2011. Private and Continual Release of Statistics. *ACM Trans. Inf. Syst. Secur.* 14, 3 (2011), 26:1–26:24.

[13] Shixi Chen and Shuigeng Zhou. 2013. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, Kenneth A. Ross, Divesh Srivastava, and Dimitris Papadias (Eds.). ACM, 653–664. https://doi.org/10.1145/2463676.2465304

[14] Edith Cohen, Xin Lyu, Jelani Nelson, Tamás Sarlós, and Uri Stemmer. 2024. Lower Bounds for Differential Privacy Under Continual Observation and Online Threshold Queries. Cryptology ePrint Archive, Paper 2024/373. https://eprint.iacr.org/2024/373 https://eprint.iacr.org/2024/373.

[15] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. 2021. Node-Level Differentially Private Graph Neural Networks. arXiv:2111.15521

[16] Wei-Yen Day, Ninghui Li, and Min Lyu. 2016. Publishing Graph Degree Distribution with Node Differential Privacy. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, Fatma Özcan, Georgia Koutrika, and Sam Madden (Eds.). ACM, 123–138. https://doi.org/10.1145/2882903.2926745

[17] Anindya De. 2012. Lower Bounds in Differential Privacy. In *Proceedings, Theory of Cryptography Conference (TCC)*, Vol. 7194. 321–338.

[18] Laxman Dhulipala, Quanquan C. Liu, Sofya Raskhodnikova, Jessica Shi, Julian Shun, and Shangdi Yu. 2022. Differential privacy from locally adjustable graph algorithms: k-Core decomposition, low out-degree ordering, and densest subgraphs. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*. 754–765. https://doi.org/10.1109/FOCS54457.2022.00077

[19] Xiaofeng Ding, Xiaodong Zhang, Zhifeng Bao, and Hai Jin. 2018. Privacy-Preserving Triangle Counting in Large Graphs. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 1283–1292. https://doi.org/10.1145/3269206.3271736

[20] Irit Dinur and Kobbi Nissim. 2003. Revealing information while preserving privacy. In *Proc. 22nd SIGACT-SIGMOD-SIGART*. 202–210. https://doi.org/10.1145/773153.773173

[21] Wei Dong, Qiyao Luo, and Ke Yi. 2023. Continual Observation under User-level Differential Privacy. In *44th SP*. 2190–2207. https://doi.org/10.1109/SP46215.2023.10179466

[22] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4004)*, Serge Vaudenay (Ed.). Springer, 486–503. https://doi.org/10.1007/11761679_29

[23] Cynthia Dwork and Jing Lei. 2009. Differential privacy and robust statistics. In *Proc. 41st STOC*. 371–380. https://doi.org/10.1145/1536414.1536466

[24] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proc. 3rd TCC*, Vol. 3876. Springer, 265–284.

[25] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2016. Calibrating Noise to Sensitivity in Private Data Analysis. *J. Priv. Confidentiality* 7, 3 (2016), 17–51.

[26] Cynthia Dwork, Frank McSherry, and Kunal Talwar. 2007. The price of privacy and the limits of LP decoding. In *Proc. 39th STOC*. 85–94. https://doi.org/10.1145/1250790.1250804

[27] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. 2010. Differential privacy under continual observation. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*. 715–724.

[28] Cynthia Dwork, Moni Naor, Omer Reingold, and Guy N. Rothblum. 2015. Pure Differential Privacy for Rectangle Queries via Private Partitions. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 9453)*, Tetsu Iwata and Jung Hee Cheon (Eds.). Springer, 735–751. https://doi.org/10.1007/978-3-662-48800-3_30

[29] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. 2010. Boosting and Differential Privacy. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*. 51–60.

[30] Talya Eden, Quanquan C. Liu, Sofya Raskhodnikova, and Adam D. Smith. 2023. Triangle Counting with Local Edge Differential Privacy. In *Proc. 50th ICALP*. 52:1–52:21. https://doi.org/10.4230/LIPICS.ICALP.2023.52

[31] Alessandro Epasto, Quanquan C. Liu, Tamalika Mukherjee, and Felix Zhou. 2024. The Power of Graph Sparsification in the Continual Release Model. arXiv:2407.17619 [cs.DS] https://arxiv.org/abs/2407.17619

[32] Alessandro Epasto, Jieming Mao, Andres Muñoz Medina, Vahab Mirrokni, Sergei Vassilvitskii, and Peilin Zhong. 2023. Differentially Private Continual Releases of Streaming Frequency Moment Estimations. In *Proceedings, Innovations in Theoretical Computer Science (ITCS)*, Vol. 251. 48:1–48:24.

[33] Hendrik Fichtenberger, Monika Henzinger, and Lara Ost. 2021. Differentially Private Algorithms for Graphs Under Continual Observation. In *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference) (LIPIcs, Vol. 204)*, Petra Mutzel, Rasmus Pagh, and Grzegorz Herman (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 42:1–42:16. https://doi.org/10.4230/LIPIcs.ESA.2021.42

[34] Hendrik Fichtenberger, Monika Henzinger, and Jalaj Upadhyay. 2023. Constant Matters: Fine-grained Error Bound on Differentially Private Continual Observation. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 10072–10092. https://proceedings.mlr.press/v202/fichtenberger23a.html

[35] Badih Ghazi, Ravi Kumar, Jelani Nelson, and Pasin Manurangsi. 2023. Private Counting of Distinct and k-Occurring Items in Time Windows. In *Proceedings, Innovations in Theoretical Computer Science (ITCS)*, Vol. 251. 55:1–55:24.

[36] Anupam Gupta, Aaron Roth, and Jonathan Ullman. 2012. Iterative constructions and private data release. In *Proceedings, Theory of Cryptography Conference (TCC)*, Vol. 7194. 339–356. https://doi.org/10.1007/978-3-642-28914-9_19

[37] Moritz Hardt and Kunal Talwar. 2010. On the Geometry of Differential Privacy. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*. 705–714.

[38] Michael Hay, Chao Li, Gerome Miklau, and David D. Jensen. 2009. Accurate Estimation of the Degree Distribution of Private Networks. In *Proceedings, SIAM International Conference on Data Mining (ICML)*. 169–178. https://doi.org/10.1109/ICDM.2009.11

[39] Monika Henzinger, A. R. Sricharan, and Teresa Anna Steiner. 2024. Private Counting of Distinct Elements in the Turnstile Model and Extensions. In *Proc. 28th RANDOM*.

[40] Monika Henzinger, Jalaj Upadhyay, and Sarvagya Upadhyay. 2023. Almost Tight Error Bounds on Differentially Private Continual Counting. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, Nikhil Bansal and Viswanath Nagarajan (Eds.). SIAM, 5003–5039. https://doi.org/10.1137/1.9781611977554.CH183

[41] Palak Jain, Iden Kalemaj, Sofya Raskhodnikova, Satchit Sivakumar, and Adam D. Smith. 2023. Counting Distinct Elements in the Turnstile Model with Differential Privacy under Continual Observation. In *Proc. 36th NeurIPS*. http://papers.nips.cc/paper_files/paper/2023/hash/0ef1afa0daa888d695dcd5e9513bafa3-Abstract-Conference.html

[42] Palak Jain, Sofya Raskhodnikova, Satchit Sivakumar, and Adam D. Smith. 2023. The Price of Differential Privacy under Continual Observation. In *Proc. 40th ICML*. 14654–14678. https://proceedings.mlr.press/v202/jain23b.html

[43] Palak Jain, Adam Smith, and Connor Wagaman. 2024. Time-Aware Projections: Truly Node-Private Graph Statistics under Continual Observation. *CoRR* abs/2403.04630 (2024). https://doi.org/10.48550/ARXIV.2403.04630 arXiv:2403.04630

[44] Iden Kalemaj, Sofya Raskhodnikova, Adam D. Smith, and Charalampos E. Tsourakakis. 2023. Node-Differentially Private Estimation of the Number of Connected Components. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2023, Seattle, WA, USA, June 18-23, 2023*, Floris Geerts, Hung Q. Ngo, and Stavros Sintos (Eds.). ACM, 183–194. https://doi.org/10.1145/3584372.3588671

[45] Vishesh Karwa, Sofya Raskhodnikova, Adam D. Smith, and Grigory Yaroslavtsev. 2014. Private Analysis of Graph Structure. *ACM Trans. Database Syst.* 39, 3 (2014), 22:1–22:33. https://doi.org/10.1145/2611523

[46] Vishesh Karwa and Aleksandra B. Slavkovic. 2012. Differentially private graphical degree sequences and synthetic graphs. In *Privacy in Statistical Databases*, Vol. 7556. Springer, 273–285. https://doi.org/10.1007/978-3-642-33627-0_21

[47] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2013. Analyzing Graphs with Node Differential Privacy. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 7785)*, Amit Sahai (Ed.). Springer, 457–476. https://doi.org/10.1007/978-3-642-36594-2_26

[48] Jure Leskovec. 2023. Databases as Graphs: Predictive Queries for Declarative Machine Learning. In *Proc. 42nd PODS 2023*. 1. https://doi.org/10.1145/3584372.3589939

[49] Ganghong Liu, Xuebin Ma, and Wuyungerile Li. 2020. Publishing Node Strength Distribution With Node Differential Privacy. *IEEE Access* 8 (2020), 217642–217650. https://doi.org/10.1109/ACCESS.2020.3040077

[50] Wentian Lu and Gerome Miklau. 2014. Exponential random graph estimation under differential privacy. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 921–930. https://doi.org/10.1145/2623330.2623683

[51] Darakhshan Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. 2011. Pan-Private Algorithms via Statistics on Sketches. In *Proceedings, ACM Symposium on Principles of Database Systems (PODS)*. 37–48.

[52] Yvonne Mülle, Chris Clifton, and Klemens Böhm. 2015. Privacy-integrated graph clustering through differential privacy. In *Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference*, Vol. 1330. 247–254.

[53] Hiep H. Nguyen, Abdessamad Imine, and Michaël Rusinowitch. 2016. Detecting communities under differential privacy. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES@CCS)*. 83–93.

[54] Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, David S. Johnson and Uriel Feige (Eds.). ACM, 75–84.

[55] Victor Perrier, Hassan Jameel Asghar, and Dali Kaafar. 2019. Private Continual Release of Real-Valued Data Streams. In *Network and Distributed System Security Symposium (NDSS)*.

[56] Davide Proserpio, Sharon Goldberg, and Frank McSherry. 2014. Calibrating Data to Sensitivity in Private Data Analysis. *Proceedings of the VLDB Endowment* 7, 8 (2014), 637–648. https://doi.org/10.14778/2732296.2732300

[57] Sofya Raskhodnikova and Adam Smith. 2016. *Differentially Private Analysis of Graphs*. Springer New York, New York, NY, 543–547. https://doi.org/10.1007/978-1-4939-2864-4_549

[58] Sofya Raskhodnikova and Adam D. Smith. 2016. Lipschitz Extensions for Node-Private Graph Statistics and the Generalized Exponential Mechanism. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*. 495–504. https://doi.org/10.1109/FOCS.2016.60

[59] Sofya Raskhodnikova and Teresa Anna Steiner. 2024. Fully Dynamic Graph Algorithms with Edge Differential Privacy. arXiv:2409.17623 [cs.DS] https://arxiv.org/abs/2409.17623

[60] Leyla Roohi, Benjamin I. P. Rubinstein, and Vanessa Teague. 2019. Differentially-private two-party egocentric betweenness centrality. In *IEEE Conference on Computer Communications (INFOCOM)*. 2233–2241. https://doi.org/10.1109/INFOCOM.2019.8737405

[61] Adam Sealfon and Jonathan R. Ullman. 2021. Efficiently Estimating Erdos-Renyi Graphs with Node Differential Privacy. *J. Priv. Confidentiality* 11, 1 (2021). https://doi.org/10.29012/jpc.745

[62] Shuang Song, Susan Little, Sanjay Mehta, Staal A. Vinterbo, and Kamalika Chaudhuri. 2018. Differentially Private Continual Release of Graph Statistics. *CoRR* abs/1809.02575 (2018). arXiv:1809.02575

[63] Jalaj Upadhyay. 2013. Random projections, graph sparsification, and differential privacy. In *ASIACRYPT (1)*, Vol. 8269.
      276–295. https://doi.org/10.1007/978-3-642-42033-7_15
[64] Yue Wang, Xintao Wu, and Leting Wu. 2013. Differential privacy preserving spectral graph analysis. In *Advances in
      Knowledge Discovery and Data Mining Pacific-Asia Conference (PAKDD)*, Vol. 7819. 329–340. https://doi.org/10.1007/978-
      3-642-37456-2_28
[65] Yue Wang, Xintao Wu, Jun Zhu, and Yang Xiang. 2013. On learning cluster coefficient of private networks. *Social
      Network Analysis and Mining* 3, 4 (2013), 925–938. https://doi.org/10.1007/s13278-013-0127-7
[66] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2015. Private Release
      of Graph Statistics using Ladder Functions. In *Proceedings, ACM International Conference on Management of Data
      (SIGMOD)*. 731–745. https://doi.org/10.1145/2723372.2737785
[67] Sen Zhang and Weiwei Ni. 2019. Graph embedding matrix sharing with differential privacy. *IEEE Access* 7 (2019),
      89390–89399. https://doi.org/10.1109/ACCESS.2019.2927365
[68] Sen Zhang, Weiwei Ni, and Nan Fu. 2020. Community Preserved Social Graph Publishing with Node Differential
      Privacy. In *Proceedings, SIAM International Conference on Data Mining (ICML)*. 1400–1405. https://doi.org/10.1109/
      ICDM50108.2020.00184

A Privacy Background

Let $\mathcal{U}$ be a universe of data items and $n \in \mathbb{N}$. A *database over $\mathcal{U}$ of size n* is an n-tuple of elements from $\mathcal{U}$. We use $\mathcal{U}^*$ to denote the set of all databases (of all sizes) over $\mathcal{U}$.

Definition A.1 (Neighboring databases). Two databases $x, y \in \mathcal{U}^n$ are *neighboring*, denoted $x \sim y$, if there is an $i \in [n]$ such that $x[i] \neq y[i]$, and $x[j] = y[j]$ for all $j \in [n] \setminus \{i\}$.

Definition A.2 (Differential privacy [22, 24]). Let $\mathcal{A}$ be an algorithm which takes as input a database over $\mathcal{U}$. Let $\varepsilon > 0$ and $\delta \in [0, 1)$. Algorithm $\mathcal{A}$ is *(ε, δ)-differentially private ((ε, δ)-DP)* if for all neighboring databases $x, y \in \mathcal{U}^*$ and all $\text{Out} \subseteq \text{range}(\mathcal{A})$,

$$\Pr[\mathcal{A}(x) \in \text{Out}] \leq e^\varepsilon \Pr[\mathcal{A}(y) \in \text{Out}] + \delta.$$

If $\mathcal{A}$ is (ε, δ)-differentially private for $\delta = 0$, it is also called *ε-differentially private (ε-DP)*.

Definition A.3 (Sensitivity). Let $k \in \mathbb{N}$, and $f : \mathcal{U}^* \to \mathbb{R}^k$ be a function. Let $p \in \{1, 2\}$. The *L_p-sensitivity of f*, denoted by Δ_p, is defined as $\Delta_p = \max_{x \sim y} \|f(x) - f(y)\|_p$.

Definition A.4 (Important probability distributions). The *Laplace distribution* centered at 0 with scale b is the distribution with probability density function $f_{\text{Lap}(b)}(x) = \frac{1}{2b} \exp\left(\frac{-|x|}{b}\right)$. We use $Y \sim \text{Lap}(b)$ or just $\text{Lap}(b)$ to denote a random variable Y distributed according to $f_{\text{Lap}(b)}(x)$. The *normal distribution* centered at 0 with variance σ^2 is the distribution with the probability density function $f_{N(0,\sigma^2)}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$. We use $Y \sim N(0, \sigma^2)$ or sometimes just $N(0, \sigma^2)$ to denote a random variable Y distributed according to $f_{N(0,\sigma^2)}$.

LEMMA A.5 (LAPLACE MECHANISM [24]). *Let $k \in \mathbb{N}$ and $\varepsilon > 0$ and $f : \mathcal{U}^* \to \mathbb{R}^k$ be a function with L_1-sensitivity Δ_1. The Laplace mechanism is defined as $\mathcal{A}(x) = f(x) + (Y_1, \ldots, Y_k)$, where $Y_i \sim \text{Lap}(\Delta_1/\varepsilon)$ are independent random variables for all $i \in [k]$. The Laplace mechanism is ε-differentially private. Further, for every $x \in \mathcal{U}^*$ and every $\beta \in (0, 1)$, it satisfies $\|\mathcal{A}(x) - f(x)\|_\infty \leq \frac{\Delta_1}{\varepsilon} \ln \frac{k}{\beta}$ with probability at least $1 - \beta$.*

LEMMA A.6 (GAUSSIAN MECHANISM [6, 10]). *Let $k \in \mathbb{N}$ and $f : \mathcal{U}^* \to \mathbb{R}^k$ be a function with L_2-sensitivity Δ_2. Let $\varepsilon \in (0, 1)$, $\delta \in (0, 1)$, $c^2 > 2 \ln(1.25/\delta)$, and $\sigma \geq c\Delta_2/\varepsilon$. The Gaussian mechanism is defined as $\mathcal{A}(x) = f(x) + (Y_1, \ldots, Y_k)$, where $Y_i \sim N(0, \sigma^2)$ are independent random variables for all $i \in [k]$. The Gaussian mechanism is (ε, δ)-differentially private. Further, for every $x \in \mathcal{U}^*$, every $\beta \in (0, 1)$, and $\sigma = \sqrt{2\ln(2/\delta)}\Delta_2/\varepsilon$, it satisfies $\|\mathcal{A}(x) - f(x)\|_\infty \leq \frac{2\Delta_2}{\varepsilon} \sqrt{\ln(2/\delta)\ln(2k/\beta)}$ with probability at least $1 - \beta$.*

LEMMA A.7 (GAUSSIAN TAIL BOUND). *Let $Y \sim N(\mu, \sigma^2)$ and $t \geq 0$. Then $\Pr[|Y - \mu| \geq \sigma t] \leq 2e^{-t^2/2}$.*

LEMMA A.8 (SIMPLE COMPOSITION [22, 23, 29]). *Let $\varepsilon_1, \varepsilon_2 > 0$ and $\delta_1, \delta_2 \in [0, 1)$. Let $\mathcal{A}_1$ be an $(\varepsilon_1, \delta_1)$-differentially private algorithm $\mathcal{U}^* \to \text{range}(\mathcal{A}_1)$ and $\mathcal{A}_2$ an $(\varepsilon_2, \delta_2)$-differentially private algorithm $\mathcal{U}^* \times \text{range}(\mathcal{A}_1) \to \text{range}(\mathcal{A}_2)$. Then $\mathcal{A}_1 \circ \mathcal{A}_2$ is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-differentially private.*

B Omitted Proofs from Section 3

PROOF OF LEMMA 3.7. Given a database $Y \in \{0, 1\}^{n \times n}$ and $B, w \in \mathbb{N}$, we construct a graph sequence on $\frac{n^2}{B^2} \times (2B + w)$ nodes, starting from graph G_0 with no edges. At a high level, we divide the matrix Y into submatrices of size $B \times B$, and for each submatrix, we build a graph as in Lemma 3.6. Specifically, for all $p_1, p_2 \in \left[\frac{n}{B}\right]$, define $Y^{(p_1,p_2)}$ as the $B \times B$ matrix with $Y^{(p_1,p_2)}[i, j] = Y[(p_1 - 1)B + i, (p_2 - 1)B + j]$ for all $i, j \in [B]$. We create a *gadget* for each $(p_1, p_2) \in \left[\frac{n}{B}\right]^2$ by allocating $2n + w$ nodes for it. In the initialization phase, each gadget is updated by running the

initialization phase from the proof of Lemma 3.6 for $Y^{(p_1,p_2)}$ and w. The initialization phase uses $\left(\frac{n}{B}\right)^2 B^2 = n^2$ time steps.

Then, for each $m \in [k]$, to process query $(a^{(m)}, b^{(m)})$, we run the processing procedure from the proof of Lemma 3.6 for every $p_1, p_2 \in \left[\frac{n}{B}\right]$ for $(a^{(m)}[(p_1-1)B+1, p_1 B], b^{(m)}[(p_2-1)B+1, p_2 B])$ within the gadget for (p_1, p_2). That is, for all $p_1 \in \left[\frac{n}{B}\right]$, we use $Bw\frac{n}{B}$ time steps to insert the edges corresponding to $a^{(m)}[(p_1-1)B+1, p_1 B]$ into all gadgets for (p_1, p_2) for all $p_2 \in \left[\frac{n}{B}\right]$. Similarly, for all $p_2 \in \left[\frac{n}{B}\right]$, we insert the edges corresponding to $b^{(m)}[(p_2-1)B+1, p_2 B])$ into all gadgets for (p_1, p_2) for all $p_1 \in \left[\frac{n}{B}\right]$. This uses in total $2\frac{n^2 w}{B}$ time steps. Afterwards, we use $2\frac{n^2 w}{B}$ time steps to delete the edges corresponding to the mth query for all gadgets (as in the proof of Lemma 3.6).

Correctness. For every pair $(i, j) \in [n]^2$, there exists exactly one pair (p_1, p_2) such that $i \in [(p_1-1)B+1, p_1 B]$ and $j \in [(p_2-1)B+1, p_2 B]$. Thus, if Y and Y' differ in entry (i, j), then only the gadget for (p_1, p_2) will differ in the initial insertion of the corresponding edge. We finish inserting all edges corresponding to query $(a^{(m)}, b^{(m)})$ at time step $t_m = n^2 + (m-1)\frac{4n^2 w}{B} + 2n^2 wB$. For each pair $(p_1, p_2) \in \left[\frac{n}{B}\right] \times \left[\frac{n}{B}\right]$, the number of triangles in the gadget for (p_1, p_2) at time t_m is $wa^{(m)}[(p_1-1)B+1, p_1 B]Y^{(p_1,p_2)}b^{(m)}[(p_2-1)B+1, p_2 B])$, and $\sum_{p_1,p_2 \in \left[\frac{n}{B}\right]} wa^{(m)}[(p_1-1)B+1, p_1 B]Y^{(p_1,p_2)}b^{(m)}[(p_2-1)B+1, p_2 B]) = wa^{(m)}Y^{(p_1,p_2)}b^{(m)}$.

Analysis. The total number of nodes is $\frac{n^2}{B^2} \times (2B + w) = \frac{2n^2}{B} + \frac{n^2 w}{B^2}$. The degree of each G_t is at most $2B + w$. Initialization takes n^2 time steps, and each query uses $4 \cdot \frac{n^2}{B} \cdot w$ time steps. Thus, $T = n^2 + 4k \cdot \frac{n^2}{B} \cdot w$. $\qquad\square$

PROOF OF LEMMA 3.10. Let $\mathcal{S} = (G_t)_{t \in [T]}$ be a dynamic graph sequence of degree at most D. Recall that $f_\triangle(G)$ returns the number of triangles in graph G. Let $(df_\triangle(\mathcal{S}, t))_{t \in [T]}$ be the difference sequence for $f_\triangle$. (See Definition 2.6.)

Our general strategy is to run a counting mechanism similar to the binary tree mechanism by Dwork et al. [27] to compute a running estimate on $f_\triangle(G_t) = \sum_{t' \in [t]} df_\triangle(\mathcal{S}, t')$. However, we use Gaussian noise to compute the noisy counts for nodes in the binary tree. We show that when building the binary tree over $df_\triangle$, the L_2-sensitivity of the vector of the counts of the binary tree nodes is $O(\sqrt{TD \log T})$. We next describe this approach in detail.

Let $\mathcal{I}_\ell = \{[j \cdot 2^\ell + 1, (j+1)2^\ell], 0 \leq j \leq \lceil T/2^\ell \rceil - 1\}$ for all (integer) levels $\ell \in [0, \log T]$ and $\mathcal{I} = \bigcup_{0 \leq \ell \leq \lfloor \log T \rfloor} \mathcal{I}_\ell$. For an interval $[a, b]$ in $\mathcal{I}$, define $s_{[a,b]}(\mathcal{S}) = \sum_{a \leq t \leq b} df_\triangle(\mathcal{S}, t)$. We first obtain a private estimate of $s_{[a,b]}$ for all intervals $[a, b] \in \mathcal{I}$, by computing $\tilde{s}_{[a,b]} = s_{[a,b]} + Y_{[a,b]}$, where $Y_{[a,b]}$ is independently drawn from $N(0, \sigma^2)$ with $\sigma = 12\sqrt{TD\left(\ln \frac{2}{\delta}\right)} \log T$. For any interval $[1, t]$, there exists a set $I_t \subseteq \mathcal{I}$ such that (i) $\bigcup_{[a,b] \in I_t}[a, b] = [1, t]$, (ii) $\bigcap_{[a,b] \in I_t} = \emptyset$, and (iii) $|I_t| \leq \lfloor \log T \rfloor + 1$. We compute an estimate of $f_\triangle(G_t)$ as $\tilde{f}_\triangle(G_t) = \sum_{[a,b] \in I_t} \tilde{s}_{[a,b]} = \sum_{[a,b] \in I_t} s_{[a,b]} + Y_{[a,b]}$.

CLAIM 1. *The L_2-sensitivity of $(s_{[a,b]})_{[a,b] \in \mathcal{I}}$ is at most $6\sqrt{TD \log T}$.*

PROOF. Let $\mathcal{S} = (G_t)_{t \in [T]}$ and $\mathcal{S}' = (G'_1, \ldots, G'_T)$ be two event-level, edge-neighboring dynamic graph sequences of degree at most D. Let t_1 be the first time step where $\mathcal{S}$ and $\mathcal{S}'$ differ. W.l.o.g., suppose an edge $e^* = (u^*, v^*)$ is inserted into G_{t_1}, but $G'_{t_1} = G'_{t_1-1}$. Let t_2 be the time step where e^* is deleted from $\mathcal{S}$ but not $\mathcal{S}'$ (set $t_2 = T + 1$ if e^* is not deleted in $\mathcal{S}$ after t_1). Then $|df_\triangle(\mathcal{S}, t_1) - df_\triangle(\mathcal{S}', t_1)| \leq D - 1$ and, if $t_2 \leq T$ then $|df_\triangle(\mathcal{S}, t_2) - df_\triangle(\mathcal{S}', t_2)| \leq D - 1$, since e^* can be involved in at most $D - 1$ triangles. For all $t \in (t_1, t_2)$, if edge $e = (u, v)$ is inserted at step t, it can form at most one triangle with e^*. Thus, $|df_\triangle(\mathcal{S}, t) - df_\triangle(\mathcal{S}', t)| \leq 1$ for all $t \notin \{t_1, t_2\}$.

Now, fix a level $\ell \in \{0, \ldots, \lfloor \log T \rfloor\}$. For each interval $[a, b] \in \mathcal{I}_\ell$ (which has length 2^ℓ), we have $|s_{[a,b]}(\mathcal{S}) - s_{[a,b]}(\mathcal{S}')| \leq D + 2^\ell$ if $t_1 \in [a, b]$ or $t_2 \in [a, b]$, and $|s_{[a,b]}(\mathcal{S}) - s_{[a,b]}(\mathcal{S}')| \leq 2^\ell$, otherwise.

For all $t \in [T]$, the total triangle count can differ by at most $D - 1$ for G_t and G_t', that is, $|f_\triangle(G_t) - f_\triangle(G_t')| \leq D - 1$. For each interval $[a, b] \in \mathcal{I}$, we have $s_{[a,b]}(\mathcal{S}) = f_\triangle(G_b) - f_\triangle(G_a)$ and $|s_{[a,b]}(\mathcal{S}) - s_{[a,b]}(\mathcal{S}')| = |f_\triangle(G_b) - f_\triangle(G_a) - f_\triangle(G_b') + f_\triangle(G_a')| \leq |f_\triangle(G_b) - f_\triangle(G_b')| + |f_\triangle(G_a') - f_\triangle(G_a)| \leq 2D$.

There are at most two intervals $I \in \mathcal{I}_\ell$ such that $t_1 \in I$ or $t_2 \in I$. For such an interval I, we have $|s_I(\mathcal{S}) - s_I(\mathcal{S}')| \leq D + \min(D, 2^\ell)$. For all other intervals $[a, b] \in \mathcal{I}_\ell$, we have $|s_{[a,b]}(\mathcal{S}) - s_{[a,b]}(\mathcal{S}')| \leq \min(2D, 2^\ell)$. Since $\mathcal{I}_\ell$ contains at most $T/2^\ell$ intervals, the L_2-sensitivity of $(s_{[a,b]})_{[a,b] \in \mathcal{I}_\ell}$ is at most

$$2D + \sqrt{\frac{T}{2^\ell}} \min(2D, 2^\ell) \leq 2D + \sqrt{\frac{T}{\min(2D, 2^\ell)}} \min(2D, 2^\ell) \leq 2D + \sqrt{T \cdot \min(2D, 2^\ell)} \leq 2D + \sqrt{2TD},$$

which is at most $4\sqrt{TD}$, since $D \leq T$. Therefore, the L_2-sensitivity of $(s_{[a,b]})_{[a,b] \in \mathcal{I}}$ is at most $\sqrt{\sum_{\ell=0}^{\lfloor \log T \rfloor} 16TD} \leq 6\sqrt{TD \log T}$. $\qquad\square$

By Lemma A.6, the algorithm that returns $\tilde{s}_{[a,b]} = s_{[a,b]} + Y_{[a,b]}$, where every $Y_{[a,b]}$ is independently drawn from $N(0, \sigma^2)$, is (ε, δ)-DP. Recall that $\tilde{f}_\triangle(G_t) = \sum_{[a,b] \in I_t} \tilde{s}_{[a,b]} = \sum_{[a,b] \in I_t} s_{[a,b]} + Y_{[a,b]}$ and note that $\sum_{[a,b] \in I_t} s_{[a,b]} = f_\triangle(G_t)$. The random variable $Z_t = \sum_{[a,b] \in I_t} Y_{[a,b]}$ has distribution $N(0, \hat{\sigma}^2)$, where $\hat{\sigma}^2 \leq (\lfloor \log T \rfloor + 1)\sigma^2$. By Lemma A.7, for all $\beta \in (0, 1)$, we have that $\Pr[|Z_t| \geq \hat{\sigma}\sqrt{2\ln(T/\beta)}] \leq \beta/T$. By a union bound over the T time steps, with probability at least $1 - \beta$, estimates returned at all steps have error at most $O(\hat{\sigma}\sqrt{\log(T/\beta)})$. Thus, the algorithm is (α, β)-accurate, for $\alpha = O\left(\frac{1}{\varepsilon}\sqrt{TD\left(\ln \frac{1}{\delta}\right)\log T}\sqrt{\log T}\sqrt{\log(T/\beta)}\right) = O\left(\frac{1}{\varepsilon}\sqrt{TD \ln \frac{1}{\delta}}\log^{3/2}\frac{T}{\beta}\right)$. $\qquad\square$

LEMMA B.1 (REDUCTION FROM MARGINALS TO TRIANGLECOUNT). *Let $n, d \in \mathbb{N}$ and $Y \in \{\{0, 1\}^d\}^n$. Then for all $w \in \mathbb{N}$, there exists a transformation from Y to a dynamic graph sequence $(G_1, \ldots, G_T)$ of N-node graphs, where $T = 2\lceil\sqrt{n}\rceil w + 2nd$ and $N = 2\lceil\sqrt{n}\rceil + w$, such that*

- *The transformations of neighboring $Y, Y' \in \{\{0, 1\}^d\}^n$ give item-level, edge-neighboring graph sequences;*
- *Let $t_0 = 2\lceil\sqrt{n}\rceil w$. For all $j \in [d]$ and $t_j = t_0 + (2j - 1)n$, we have $f_\triangle(G_{t_j}) = w \sum_{i=1}^n Y_i[j]$.*

PROOF. We define a set $V = V_0 \cup V_1 \cup W$, where V_0, V_1 and W are pairwise disjoint, $|V_0| = |V_1| = \lceil\sqrt{n}\rceil$, and $|W| = w$. There are at least n node pairs $(v_0, v_1) \in V_0 \times V_1$. We give them an arbitrary order $e_1, \ldots, e_{\lceil\sqrt{n}\rceil^2}$. In particular, there exist the node pairs $e_1, \ldots, e_n$.

In the initialization phase, we insert edge (v, u) for all $v \in V_0 \cup V_1$ and all $u \in U$, in an arbitrary but fixed order. This takes $2\lceil\sqrt{n}\rceil w$ time steps. Then, for all $j \in [d]$ and $i \in [n]$, if $Y_i[j] = 1$ then we insert e_i at time $t_0 + 2(j - 1)n + i$ and delete e_i at time $t_0 + (2j - 1)n + i$; otherwise, we do nothing at these time steps. This takes $2nd$ time steps, and $T = 2\lceil\sqrt{n}\rceil w + 2nd$ for the whole construction.

Correctness. If Y and Y' are neighboring input databases differing in row i^*, then the resulting graph sequences for Y and Y' differ only in insertions and deletions related to e_{i^*}. Thus, they are item-level, edge-neighboring. The only triangles that can form have to include an edge from W to V_0, from V_0 to V_1, and from V_1 to W. Let $G_t = (V, E_t)$. Since every node in W is connected to every node $V_0 \cup V_1$, the number of triangles is exactly $w|\{e_i : e_i \in E_t\}|$. At time step $t_j = t_0 + (2j - 1)n$, the graph G_{t_j} includes an edge e_i if and only if $Y_i[j] = 1$. Thus, the total triangle count is equal to $w \sum_{i=1}^n Y_i[j]$. $\qquad\square$

C Omitted Proofs from Section 4

To prove Theorem 4.1, we first give Lemma C.1, which is the key lemma of this proof.

Lemma C.1 (Transformation from degree-restricted privacy). *Let $T, N, k \in \mathbb{N}$ and let f be a function $\mathcal{G}(V) \to \mathbb{R}^k$. Let $\alpha : \mathbb{N} \to \mathbb{R}^+$ be a nondecreasing function of the degree D. Let $\varepsilon > 0$ and $\delta \in [0, 1)$ and $\beta, \beta_s \in (0, 1)$. Assume that for every $D \in \mathbb{N}$, there is an event-level D-restricted (ε, δ)-edge-DP algorithm $\mathcal{A}_D$ which is $(\alpha(D), \beta)$-accurate for f on all dynamic graph sequences with N nodes, maximum degree D and length T. Then there exists an event-level $(\varepsilon(2+\log N), \delta(1+\log N)+\beta_s(1+e^\varepsilon))$-edge DP algorithm which is $(\alpha(D'), \beta(1 + \log N) + \beta_s)$-accurate for f on all dynamic graph sequences with N nodes, maximum degree D, and length T, where $D' = O(D + \frac{\log T}{\varepsilon} \log \frac{TN}{\beta_s})$ for all $D \in \mathbb{N}$. The value D does not need to be given to the algorithm.*

The proof requires the following definition:

Definition C.2 ((ε, δ)-indistinguishability). Let $\mathcal{U}$ and $\mathcal{Y}$ be two sets. Random variables Y and Y', mapping $\mathcal{U}$ to $\mathcal{Y}$, are (ε, δ)-*indistinguishable* if for all $\text{Out} \subseteq \mathcal{Y}$, we have $\Pr[Y \in \text{Out}] \le e^\varepsilon \Pr[Y' \in \text{Out}]+\delta$ and $\Pr[Y' \in \text{Out}] \le e^\varepsilon \Pr[Y \in \text{Out}]+\delta$. We denote that Y and Y' are (ε, δ)-indistinguishable by $Y \approx_{\varepsilon,\delta} Y'$.

That is, an algorithm $\mathcal{A}$ is (ε, δ)-DP if and only if $\mathcal{A}(x) \approx_{\varepsilon,\delta} \mathcal{A}(y)$ for all neighboring x and y.

Proof of Lemma C.1. We run the event-level ε-edge-DP algorithm for estimating the degree sequence from Lemma 4.2 with failure probability set to β_s and keep track of the maximum (noisy) degree $\tilde{d}_{\max}^{(t)}$ seen up until time t. By Lemma 4.2, this algorithm is (γ, β_s)-accurate with $\gamma = O(\frac{\log T}{\varepsilon} \log \frac{TN}{\beta_s})$. Let $\hat{N} = 2^{\lceil \log N \rceil} < 2N$. Note $\log \hat{N} = \lceil \log N \rceil < \log N + 1$. We set $D_j = \gamma + 2^j$ and $\tau_j = 2^j$ for all $j \in [\log \hat{N}]$. For all time steps t such that $\tau_{j-1} \le \tilde{d}_{\max}^{(t)} < \tau_j$, we run $\mathcal{A}_{D_j}$.

Accuracy. Let $\mathcal{S} = (G_1, \ldots, G_T)$ and $\hat{D}^{(t)}$ be the maximum degree of $(G_1, \ldots, G_t)$ for all $t \in [T]$. With probability at least $1 - \beta_s$, at all time steps $t \in [T]$, we have that $\hat{D}^{(t)} \in [\tilde{d}_{\max}^{(t)} - \gamma, \tilde{d}_{\max}^{(t)} + \gamma]$. Call that event C. Conditioning on C, then for $j \in [\log \hat{N}]$, at all time steps t where we run $\mathcal{A}_{D_j}$, we have $\tilde{d}_{\max}^{(t)} \in [\tau_{j-1}, \tau_j)$, and therefore $\hat{D}^{(t)} \in [\tau_{j-1} - \gamma, \tau_j + \gamma) = [D_j/2 - \gamma, D_j)$. Conditioning on C, with probability at least $1 - \beta$, for a fixed $j \in [\log \hat{N}]$, algorithm $\mathcal{A}_{D_j}$ has error at most $\alpha(D_j)$ for all time steps t where we run $\mathcal{A}_{D_j}$. Thus, by a union bound, for all $j \in [\log \hat{N}]$, we have that with probability at least $1 - \beta_s - \beta(1 + \log N)$, the error at every time step t where we run $\mathcal{A}(D_j)$ is bounded by $\alpha(D_j) = \alpha(O(\hat{D}^{(t)} + \gamma)) = \alpha(O(\hat{D}^{(T)} + \gamma))$.

Privacy. The algorithm implicitly partitions $[T]$ into intervals $I_1, \ldots, I_{\log \hat{N}}$, such that we run $\mathcal{A}_{D_j}$ on I_j for $j \in [\log \hat{N}]$. Call $\mathcal{A}_{\text{Int}}$ the algorithm which takes as input a graph sequence $\mathcal{S}$ and outputs the interval partition. Since it is post-processing of the algorithm from Lemma 4.2, $\mathcal{A}_{\text{Int}}$ is event-level ε-edge-DP. Let $\mathcal{S} = (G_1, \ldots, G_T)$ and $\mathcal{S}' = (G'_1, \ldots, G'_T)$ be two neighboring input sequences. Let $\mathcal{I} = (I_1, \ldots, I_{\log \hat{N}})$ be an interval partition such that for all $j \in [\log \hat{N}]$ and the interval $I_j = [t_j, t_{j+1})$ the maximum degrees of G_t and G'_t are bounded by D_j for all $t \in I_j$. We get $\mathcal{A}_{D_j}(G_{t_j}, \ldots, G_{t_{j+1}-1}) \approx_{\varepsilon,\delta} \mathcal{A}_{D_j}(G'_{t_j}, \ldots, G'_{t_{j+1}-1})$ for all $j \in [\log \hat{N}]$, by definition of $\mathcal{A}_{D_j}$. Thus, $\left(\mathcal{A}_{D_j}(G_{t_j}, \ldots, G_{t_{j+1}-1}) \right)_{j \in [\log \hat{N}]} \approx_{\varepsilon',\delta'} \left(\mathcal{A}_{D_j}(G'_{t_j}, \ldots, G'_{t_{j+1}-1}) \right)_{j \in [\log \hat{N}]}$, where $\varepsilon' = \varepsilon(1 + \log N)$ and $\delta' = \delta(1 + \log N)$ by Lemma A.8.

Let $\mathcal{A}_{\text{all}D}$ be the algorithm that takes as input a sequence $\mathcal{S}$ and an interval partition $\mathcal{I} = (I_1, \ldots, I_{\log \hat{N}})$, and runs $\mathcal{A}_{D_j}$ on $G_{t_j}, \ldots, G_{t_{j+1}-1}$ for each $I_j = [t_j, t_{j+1})$ and $j \in [\log \hat{N}]$. Our full algorithm can be seen as $\mathcal{A}(\mathcal{S}) := \mathcal{A}_{\text{all}D}(\mathcal{S}, \mathcal{A}_{\text{Int}}(\mathcal{S}))$. Let B be the set of all interval sequences $\mathcal{I} = (I_1, I_2, \ldots, I_{\log \hat{N}})$, where $I_j = [t_j, t_{j+1})$ for all $j \in [\log \hat{N}]$, such that the maximum degrees of $G_{t_j}, \ldots, G_{t_{j+1}-1}$ and $G'_{t_j}, \ldots, G'_{t_{j+1}-1}$ are bounded by D_j, for all $j \in [\log \hat{N}]$. Let H be the set of intervals where the degree bound is violated for $\mathcal{S}$, and H' be the set of intervals where

the degree bound is violated for $\mathcal{S}'$, such that $B = \overline{H \cup H'}$. By the properties of $\mathcal{A}_{\text{Int}}$, we have $\Pr[\mathcal{A}_{\text{Int}}(\mathcal{S}) \in H] \leq \beta_s$ and $\Pr[\mathcal{A}_{\text{Int}}(\mathcal{S}) \in H'] \leq e^\varepsilon \Pr[\mathcal{A}_{\text{Int}}(\mathcal{S}') \in H'] \leq e^\varepsilon \beta_s$. Thus, by a union bound, $\Pr[\mathcal{A}_{\text{Int}}(\mathcal{S}) \in H \cup H'] \leq \beta_s + e^\varepsilon \beta_s$ and therefore $\Pr[\mathcal{A}_{\text{Int}}(\mathcal{S}) \in B] \geq 1 - \beta_s - e^\varepsilon \beta_s$. For every $\text{Out} \in \text{range}(\mathcal{A})$,

$$\Pr[\mathcal{A}(\mathcal{S}) \in \text{Out}] = \Pr[\mathcal{A}_{\text{all}D}(\mathcal{S}, \mathcal{A}_{\text{Int}}(\mathcal{S})) \in \text{Out}]$$
$$\leq \Pr[\mathcal{A}_{\text{all}D}(\mathcal{S}, \mathcal{A}_{\text{Int}}(\mathcal{S})) \in \text{Out}|\mathcal{A}_{\text{Int}}(\mathcal{S}) \in B] \cdot \Pr[\mathcal{A}_{\text{Int}}(\mathcal{S}) \in B] + \beta_s(1 + e^\varepsilon).$$

We further bound the first summand:

$$\Pr[\mathcal{A}_{\text{all}D}(\mathcal{S}, \mathcal{A}_{\text{Int}}(\mathcal{S})) \in \text{Out}|\mathcal{A}_{\text{Int}}(\mathcal{S}) \in B] \cdot \Pr[\mathcal{A}_{\text{Int}}(\mathcal{S}) \in B]$$
$$= \sum_{I \in B} \Pr[\mathcal{A}_{\text{all}D}(\mathcal{S}, I) \in \text{Out}] \Pr[\mathcal{A}_{\text{Int}}(\mathcal{S}) = I]$$
$$\leq \sum_{I \in B} \left(e^{\varepsilon(\log N+1)} \Pr[\mathcal{A}_{\text{all}D}(\mathcal{S}', I) \in \text{Out}] + \delta(\log N + 1) \right) \Pr[\mathcal{A}_{\text{Int}}(\mathcal{S}) = I]$$
$$\leq \sum_{I \in B} e^{\varepsilon(\log N+2)} \Pr[\mathcal{A}_{\text{all}D}(\mathcal{S}', I) \in \text{Out}] \Pr[\mathcal{A}_{\text{Int}}(\mathcal{S}') = I] + \delta(\log N + 1)$$
$$= e^{\varepsilon(\log N+2)} \Pr[\mathcal{A}_{\text{all}D}(\mathcal{S}', \mathcal{A}_{\text{Int}}(\mathcal{S}')) \in \text{Out}|\mathcal{A}_{\text{Int}}(\mathcal{S}') \in B] \Pr[\mathcal{A}_{\text{Int}}(\mathcal{S}') \in B] + \delta(\log N + 1)$$
$$\leq e^{\varepsilon(\log N+2)} \Pr[\mathcal{A}(\mathcal{S}') \in \text{Out}] + \delta(\log N + 1),$$

where the first inequality follows since for each $I \in B$, we have that $\mathcal{A}_{\text{all}D}(\mathcal{S}, I) \approx_{\varepsilon', \delta'} \mathcal{A}_{\text{all}D}(\mathcal{S}', I)$ with $\varepsilon' = \varepsilon(\log N + 1)$ and $\delta' = \delta(\log N + 1))$, as argued above. The second inequality follows since $\mathcal{A}_{\text{Int}}$ is ε-DP. Together, this gives $\Pr[\mathcal{A}(\mathcal{S}) \in \text{Out}] \leq e^{\varepsilon(\log N+2)} \Pr[\mathcal{A}(\mathcal{S}') \in \text{Out}] + \delta(\log N + 1)] + \beta_s(1 + e^\varepsilon)$. □

PROOF OF THEOREM 4.1. Apply Lemma C.1 with $\varepsilon = \frac{\varepsilon'}{\log N+2}$, $\delta = \frac{\delta'}{2(1+\log N)}$, and $\beta = \beta_s = \min\left(\frac{\delta'}{2+2e^{\varepsilon'}}, \frac{\beta'}{2+\log N}\right)$. Note that $\frac{\delta'}{2+2e^{\varepsilon'}} \leq \frac{\delta'}{2+2e^\varepsilon}$. □

D Omitted Proofs form Section 5

PROOF OF THEOREM 5.1. The second term in the minimum expression (for both items) comes from the trivial algorithm that always outputs $f(G_0)$, where $G_0 = (V, \emptyset)$, without looking at the data. It remains to analyze the first term. For all $p \in \{1, 2\}$ and every function f with L_p-sensitivity Δ_p, the rescaled function $f' = f/\Delta_p$ has sensitivity 1. Therefore, it suffices to analyze the first term in the minimum for functions with $\Delta_p = 1$.

The algorithm for the first term uses an edge-DP algorithm to recompute the function value every B time steps, and does not update the output in between. That is, we divide the time line into *blocks* of the form $[1, B], [B + 1, 2B], \ldots, [(\lceil \frac{T}{B} \rceil - 1)B + 1, T]$. Denote $t_1, \ldots, t_{\lceil T/B \rceil}$ the last time step in each block, i.e., $t_i = i \cdot B$ for all $i \in [\lceil T/B \rceil - 1]$ and $t_{\lceil T/B \rceil} = T$. Since f has L_p-sensitivity 1, and all G_i, G_{i+1} are edge-neighboring, we have that $\|f(G_{t_{i+1}}) - f(G_{t_i})\|_\infty \leq B$ for all $i \in [\lceil T/B \rceil - 2]$. We first set the output to 0. At every time step t_i, we compute an estimate of $f(G_{t_i})$ using a differentially private mechanism, and within a block, we do not update the output. That is, for ε-differential privacy, we run the Laplace mechanism on $\lceil \frac{T}{B} \rceil$ inputs $G_{t_1}, \ldots, G_{t_{\lceil T/B \rceil}}$, and for every $t_i < t < t_{i+1}$, we output the same value as in the previous time step. Since for each of these inputs and every item-level edge-neighboring sequence $G'_1, \ldots, G'_T$, we have that G_{t_j} and G'_{t_j} are neighboring, the L_1-sensitivity of $(f(G_{t_1}), \ldots, f(G_{t_{\lceil T/B \rceil}}))$ is $\lceil \frac{T}{B} \rceil$. The value of α follows by balancing B with the error of the Laplace mechanism: By Lemma A.5, the Laplace mechanism has error at most $\lceil \frac{T}{B} \rceil \frac{1}{\varepsilon} \ln \frac{kT}{\beta}$. Setting $B = \Theta\left(\sqrt{\frac{T}{\varepsilon} \ln \frac{Tk}{\beta}}\right)$ gives the claimed bound α. Similarly, for (ε, δ)-differential privacy, we

run the Gaussian mechanism on $\lceil \frac{T}{B} \rceil$ inputs $G_{t_1}, \ldots, G_{t_{\lceil T/B \rceil}}$. Since for each of these inputs and every item-level neighboring sequence $G'_1, \ldots, G'_T$, we have that G_{t_j} and G'_{t_j} are neighboring, the L_2-sensitivity of $(f(G_{t_1}), \ldots, f(G_{t_{\lceil T/B \rceil}}))$ is $\sqrt{\lceil T/B \rceil}$. The value of α follows by balancing B with the error of the Gaussian mechanism: By Lemma A.6, the Gaussian mechanism has error at most $\sqrt{\lceil T/B \rceil} \frac{2}{\varepsilon} \sqrt{\ln(2/\delta) \ln(2Tk/\beta)}$. Setting $B = \Theta(\sqrt[3]{\frac{T}{\varepsilon^2} (\ln \frac{Tk}{\beta}) \ln \frac{1}{\delta}})$ gives the claimed bound α. $\qquad\square$

E Omitted Proofs from Section 6

Proof of Lemma 6.10. Let $H = (V', E')$ and (H, e_1) be a 1-edge distinguishing gadget of size (n_g, m_g) of weight w for f. W.l.o.g. assume $e_1 \notin E'$. We define a graph sequence on $n \cdot n_g$ nodes as follows: In an *initialization phase*, we build n copies of H, denoted $H^{(1)}, \ldots, H^{(n)}$, using $m_g n$ time steps. For a vertex pair e in H and $i \in [n]$, let $e^{(i)}$ denote the copy of e in $H^{(i)}$.

For all $j \in [d]$, we first have an *insertion phase*, where for all $i \in [n]$, if $Y_i[j] = 1$, then we insert $e_1^{(i)}$ into $H^{(i)}$ at time $m_g n + 2(j-1)n + i$; otherwise, we do nothing at that time step. For all but the last phase, the insertion phase is followed by a *deletion phase*, where the edges that were inserted in the insertion phase are deleted.

Correctness. If Y and Y' are neighboring input databases differing in the row $i^* \in [n]$, then the resulting graph sequences differ only in insertions and deletions related to $e_1^{(i^*)}$. Thus, they are item-level, edge-neighboring. The insertion phase for the jth query ends at time step $t_j = (m_g + 1)n + 2(j-1)n + n$. At that time steps, the $H^{(i)}$ contains $e_1^{(i)}$ if and only if $Y_i[j] = 1$. Thus, $f(G_{t_j}) = f(G_{t_0}) + w \cdot \sum_{i \in [n]} Y_i[j]$.

Analysis. The initialization phase uses $(m_g + 1)n$ time steps. Each of the d insertion phases and $d - 1$ deletion phases uses n time steps. Thus, $T = (m_g + 2d)n$. There are $N = n \cdot n_g$ nodes in the graph. $\qquad\square$

Proof of Theorem 6.9. **Case $\delta > 0$:** Given T and N, we set $d = \lfloor (T\varepsilon \log T\varepsilon)^{2/3} \rfloor$ and $n = \lfloor \min \left(\frac{\sqrt{d}}{6\varepsilon \log d}, \frac{T}{2m_g}, \frac{N}{n_g} \right) \rfloor$. The reduction from Lemma 6.10 gives a sequence with at most N nodes and $2nd + T/2$ updates, where $2nd \leq \frac{d^{3/2}}{3\varepsilon \log d} \leq \frac{T\varepsilon \log(T\varepsilon)}{3\varepsilon \cdot \frac{2}{3} \log(T\varepsilon \log(T\varepsilon))} \leq \frac{T}{2}$. We can add dummy nodes and dummy time steps to get a dynamic N-node graph sequence of length T. If we can estimate $f(G_t)$ with error up to α at all time steps with probability at least $2/3$, then for all $j \in [d]$, we can estimate the jth marginal given by $\frac{1}{n} \sum_{i \in n} Y_i[j] = \frac{f(G_{t_j}) - f(G_{t_0})}{nw}$ up to error $\frac{2\alpha}{nw}$. Lemma 3.13 gives $\alpha = \Omega \left(wn \cdot \min \left(\frac{\sqrt{d}}{n\varepsilon \log d}, 1 \right) \right) = \Omega \left(w \cdot \min \left(\frac{T^{1/3}}{\varepsilon^{2/3} \log^{2/3} T}, \frac{T}{m_g}, \frac{N}{n_g} \right) \right)$.

Case $\delta = 0$: Given T and N, set $d = \lfloor \sqrt{T\varepsilon} \rfloor$ and $n = \lfloor \min \left(\frac{T}{(m_g+1)}, \frac{N}{n_g}, \frac{d}{4\varepsilon} \right) \rfloor$. The reduction from Lemma 6.10 gives a sequence with at most N nodes and at most T updates, and we can add dummy nodes and dummy time steps to get a dynamic N-node graph sequence of length T. By Lemma 3.13, we have $\alpha = \Omega \left(wn \cdot \min \left(\frac{d}{n\varepsilon}, 1 \right) \right) = \Omega \left(w \cdot \min \left(\sqrt{\frac{T}{\varepsilon}}, \frac{T}{m_g}, \frac{N}{n_g} \right) \right)$. $\qquad\square$