# Sublinear Space Graph Algorithms in the Continual Release Model

**Alessandro Epasto**
Google Research, New York, NY, USA

**Quanquan C. Liu**
Yale University, New Haven, CT, USA

**Tamalika Mukherjee**
Columbia University, New York, NY, USA

**Felix Zhou**
Yale University, New Haven, CT, USA

## Abstract

The graph continual release model of differential privacy seeks to produce differentially private solutions to graph problems under a stream of edge updates where new private solutions are released after each update. Thus far, previously known *edge-differentially private* algorithms for most graph problems including densest subgraph and matchings in the continual release setting only output real-value estimates (not vertex subset solutions) and do not use sublinear space. Instead, they rely on computing exact graph statistics on the input [40, 77]. In this paper, we leverage *sparsification* to address the above shortcomings for edge-insertion streams. Our edge-differentially private algorithms use sublinear space with respect to the number of edges in the graph while some also achieve sublinear space in the number of vertices in the graph. In addition, for the densest subgraph problem, we also output edge-differentially private vertex subset solutions; no previous graph algorithms in the continual release model output such subsets.

We make novel use of assorted sparsification techniques from the non-private streaming and static graph algorithms literature to achieve new results in the sublinear space, continual release setting. This includes algorithms for densest subgraph, maximum matching, as well as the first continual release $k$-core decomposition algorithm. We also develop a novel sparse level data structure for $k$-core decomposition that may be of independent interest. To complement our insertion-only algorithms, we conclude with polynomial additive error lower bounds for edge-privacy in the fully dynamic setting, where only logarithmic lower bounds were previously known.

**2012 ACM Subject Classification** Theory of computation → Theory of database privacy and security; Theory of computation → Dynamic graph algorithms; Theory of computation → Sketching and sampling

**Keywords and phrases** Differential Privacy, Continual Release, Densest Subgraph, $k$-Core Decomposition, Maximum Matching

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2025.40

**Category** RANDOM

**Related Version** *Full Version*: `https://arxiv.org/pdf/2407.17619` [34]

**Funding** Quanquan C. Liu and Felix Zhou are supported by a Google Academic Research Award and by NSF Grant #2453323. Tamalika Mukherjee is supported in part by NSF grant CNS-2138834, a Google Cyber NYC Award, and the Center for Smart Streetscapes, an NSF Engineering Research Center, under grant agreement EEC-2133516. Felix Zhou acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

© Alessandro Epasto, Quanquan C. Liu, Tamalika Mukherjee, and Felix Zhou;
licensed under Creative Commons License CC-BY 4.0



## 1 Introduction

Today's data is not only massive in size but also rapidly growing. As the world becomes increasingly digitized, the sensitive associations between individuals are becoming a key focus for data analysis. Such a focus also comes with privacy risks and the question of how to safeguard privacy. Differential privacy [27] is the gold standard for privacy protection, with a rich body of work focusing on *static* graphs [3, 10, 22, 21, 33, 58, 57, 60, 68, 71, 73, 72, 79]. However, today's graphs are not static; they are rapidly expanding, with new connections between individuals forming every minute. Streaming graph algorithms, a well-studied area in the non-private setting [6, 1, 36, 37, 39, 42, 52, 51, 69, 63, 64, 65], are designed to handle such massively evolving datasets. In this model, updates to the dataset are received in a stream. Online streaming algorithms continuously release accurate graph statistics after each update [16, 46, 17, 45], but these graph statistics may reveal private information.

Our paper focuses on studying graph problems in the well-known *continual release* model [29, 14] of differential privacy that promises a strong guarantee for online streaming outputs: the entire vector of *all outputs* satisfies differential privacy. The notion of graph privacy we consider is edge-differential privacy, where we require the output distributions of the algorithm on edge-neighboring stream inputs to be "close", where edge-neighboring streams differ in one edge update. Despite the large body of differential privacy works on static graph algorithms, there are few works on graph continual release [40, 56, 77] and related graph models [81]. While there has been substantial progress in graph continual release algorithms in recent years, there are some drawbacks to existing algorithms, including: 1) not releasing private (vertex subset) solutions in addition to their real-valued function outputs and 2) requiring exact algorithms for computing graph statistics, thus necessitating linear space usage in the number of *edges*.

In this paper, we take the first steps to address these shortcomings for insertion-only streams in the continual release model via the following contributions:

- We give the first $k$-core decomposition algorithm in the continual release model.
- We formulate the first continual release graph algorithms that return edge-differentially private vertex subset solutions for densest subgraph (in addition to the value of the solutions).
- We give the first *sublinear* space algorithms for general graphs to solve densest subgraph, $k$-core decomposition, and matching problems.

The approximation guarantees of our algorithms nearly match the guarantees for their private static or non-private streaming counterparts. Moreover, as is standard in streaming, all of our results hold for a single pass over the stream.

Resolving these issues closes the gap between continual release graph algorithms and their non-private streaming counterparts for many applications. While the private streaming literature has explored sublinear space outside the graph algorithms setting [13, 35, 80, 11], it remains unaddressed in general graphs for most problems in continual release. Furthermore, while streaming algorithms typically operate on datasets too large for memory, practical applications often require information about each vertex, not just aggregate values. For example, advertisers may want the members of the current densest community, not merely its density. Hence, achieving both goals–sublinear space combined with useful vertex information–is essential to the development of private graph algorithms.

## 1.1 Sparsification

The main algorithmic technique we use in this paper is *sparsification*, traditionally used in the streaming, static, and dynamic graph algorithms literature. Sparsification aims to simplify a large graph by strategically selecting a smaller set of its edges. This smaller "sketch" of the graph still captures the essential information needed for a specific problem. Such sparsifiers exist for a large number of problems in the non-private literature. In general, graph sparsifiers are *problem-specific*, where each sparsifier provides guarantees for only one or two closely related problems. We design novel methods for creating differentially private problem-specific graph sparsifiers in continual release, which we describe in more detail in the technical overview (Section 2).

Using sparsification techniques, we design a number of new continual release algorithms that achieve the same utility guarantees as previous algorithms for the problem in the static DP and non-private streaming settings up to logarithmic factors. Our sparsification techniques directly lead to space savings in the edge-privacy setting for a variety of problems including densest subgraph, $k$-core decomposition, and maximum matching. We return vertex subsets for the densest subgraph problem, approximate $k$-core numbers for every vertex in core decomposition, and estimates for the size of the maximum matching, all using sublinear space in the *number of edges*. For maximum matching, we even use space that is sublinear in the number of vertices.[1]

Finally, to complement our results on insertion-only streams, we demonstrate the difficulty of obtaining algorithms in fully dynamic streams via improved polynomial error lower bounds for maximum cardinality matching, counting connected components, and triangle counting based on reductions from inner product queries. Similar to the recent continual release lower bounds for counting distinct elements given in [54], our lower bounds support the observation from previous works that the fully dynamic setting results in significantly more error [77, 40] than the insertion-only setting within the continual release model.

## 1.2 Our Contributions

Our paper gives the following set of results, stated informally here and given in more precise terms in their respective sections. We group our results by graph problems for $n$-vertex $m$-edge graph streams over $T$ time steps and our multiplicative guarantees are given in terms of a constant factor $\eta > 0$.

For our algorithmic results, we consider insertion-only streams where (possibly empty) edge insertions are given in the stream. For simplicity, we assume that each non-empty edge is inserted at most once. Without requiring privacy, this closely resembles the well-studied online streaming setting (see related works in Section 3). Our algorithms also incur logarithmic additive noise, which is necessary, as demonstrated by lower bounds that hold even in the static model (see Table 1).

Our lower bounds are given for fully dynamic streams where edges can be inserted and deleted multiple times, also known as the *turnstile* model. In this setting, we show that polynomial additive error is necessary, demonstrating a fundamental difficulty in maintaining small errors in the presence of both insertions and deletions.

[1] While all non-private streaming algorithms use space that is sublinear in the number of total edges, very few use space sublinear in the number of vertices, although results sublinear in the number of vertices are most desirable.

All of our results in the continual release model are given for edge-neighboring streams where the two streams differ by one edge update; results in the static DP setting are given for edge-neighbors (graphs that differ by one edge). Throughout the paper, we will use the phrase *$\varepsilon$-edge DP* to refer to these settings. Our algorithms produce $(\beta, \zeta)$-bicriteria approximations where $\beta$ is the multiplicative factor in the approximation and $\zeta$ is the additive error. Our approximation bounds hold, with high probability, for the released solution after *every* update in the stream. Throughout the paper, we assume $n \leq T = O(\text{poly}(n))$. This is reasonable as $O(n^2)$ (non-empty) updates are sufficient to have a complete graph.[2] We compare our results with static edge-DP lower bounds as such lower bounds also hold in the insertion-only continual release setting.[3] A more in-depth discussion of related works can be found in Section 3. Our entire set of results can be found in Table 1, including baseline algorithms and lower bounds in the static edge-DP and non-private streaming settings.

**$k$-Core Decomposition (Section 5).** We show the first $\varepsilon$-edge DP algorithm for the $k$-core decomposition problem in the continual release model. Our algorithm returns approximate $k$-core numbers at every timestep while using $o(m)$ space. The additive errors of our algorithms are near-optimal as they match the recent additive error lower bounds up to logarithmic factors (see Table 1). We note that these lower bounds hold even when allowing for constant multiplicative error.

▶ **Theorem 1** (See Theorem 15). *We obtain a $\left(2+\eta, O\left(\frac{\log^3(n)}{\eta^2\varepsilon}\right)\right)$-approximate $\varepsilon$-edge differentially private $k$-core decomposition algorithm that outputs core number estimates in the insertion-only continual release model using $\tilde{O}\left(\frac{n}{\eta^4\varepsilon}\right)$ space.*

The essence of our $k$-core algorithm is a novel *sparse* level data structure that may be of independent interest beyond our work. To the best of our knowledge, such a data structure did not previously exist in the privacy or graph algorithms literature.

**Densest Subgraph (Section 6).** We obtain the first $\varepsilon$-edge DP densest subgraph algorithms that return subsets of vertices with near-optimal density in the continual release model. Our algorithms also use $o(m)$ space. We emphasize that the approximation guarantees nearly match, up to logarithmic terms, the guarantees of their private static or non-private streaming counterparts. In turn, these nearly match the information-theoretic lower bound up to logarithmic factors. Again, we emphasize that these lower bounds hold even when allowing for constant multiplicative error.

▶ **Theorem 2** (See Theorem 16). *We obtain a $\left(2+\eta, O\left(\frac{\log^2(n)}{\eta^2\varepsilon}\right)\right)$-approximate $\varepsilon$-edge differentially private densest subgraph algorithm that outputs a subset of vertices in the insertion-only continual release model using $\tilde{O}\left(\frac{n}{\eta^2\varepsilon}\right)$ space.*

At the cost of slightly more additive error and space usage, we can improve the multiplicative error.

▶ **Theorem 3** (See Theorem 17). *We obtain a $\left(1+\eta, O\left(\frac{\log^5(n)}{\eta^4\varepsilon}\right)\right)$-approximate $\varepsilon$-edge differentially private densest subgraph algorithm that outputs a subset of vertices in the insertion-only continual release model using $\tilde{O}\left(\frac{n}{\eta^5\varepsilon}\right)$ space.*

---

[2] Our results can be modified to handle streams of length $T = \omega(\text{poly}(n))$.

[3] Otherwise, if we obtain better error in the continual release setting, we can solve the static problem with better error by inserting all of the edges of the static graph and taking the solution released at the end of the stream.

**Table 1** Our results are highlighted in green. Non-private bounds are shown in gray. All results are for edge-privacy. Bicriteria approximations $(\beta, \zeta)$ are given, where $\beta$ is the multiplicative factor and $\zeta$ is the additive error. $\tilde{\alpha}$ is a public upper bound on the maximum arboricity of the stream; our approximation guarantees for maximum matching hold when $\tilde{\alpha}$ upper bounds the private arboricity and our privacy guarantees *always* hold. $\alpha$ in the non-private streaming bound is the maximum arboricity in the stream.

| Problem | Space | Approximation | Reference | Model | Solution Type |
|---|---|---|---|---|---|
| $k$-Core Decomp. | $\tilde{O}\left(\frac{n}{\eta^4\varepsilon}\right)$ | $\left(2+\eta, O\left(\frac{\log^3(n)}{\eta^2\varepsilon}\right)\right)$ | Theorem 15 | Insertion Only | Upper Bound |
| | $\Theta(m+n)$ | $\left(1, O\left(\frac{\log(n)}{\varepsilon}\right)\right)$ | [20] | Static | Upper Bound |
| | | $\left(\beta, \Omega\left(\frac{\log n}{\varepsilon\beta}\right)\right)$ | [48] | Static | Lower bound |
| | $\tilde{O}\left(\frac{n}{\eta^2}\right)$ | $(1+\eta, 0)$ | [37] | Non-Private Streaming | Upper Bound |
| Densest Subgraph | $\tilde{O}\left(\frac{n}{\eta^2\varepsilon}\right)$ | $\left(2+\eta, O\left(\frac{\log^2(n)}{\eta\varepsilon}\right)\right)$ | Theorem 16 | Insertion Only | Upper Bound Vertex Subset |
| | $\tilde{O}\left(\frac{n}{\eta^5\varepsilon}\right)$ | $\left(1+\eta, O\left(\frac{\log^5(n)}{\eta^4\varepsilon}\right)\right)$ | Theorem 17 | Insertion Only | Upper Bound Vertex Subset |
| | $\Theta(m+n)$ | $\left(1+\eta, O\left(\frac{\log^4(n)}{\eta^3\varepsilon}\right)\right)$ | [21] | Static | Upper Bound Vertex Subset |
| | $\Theta(m+n)$ | $\left(2, O\left(\frac{\log(n)}{\varepsilon}\right)\right)$ | [20] | Static | Upper Bound Vertex Subset |
| | $\Theta(m+n)$ | $\left(2+\eta, O\left(\frac{\log^2(n)}{\eta\varepsilon}\right)\right)$ | [22] | Static | Upper Bound Vertex Subset |
| | | $\left(\beta, \Omega\left(\frac{1}{\beta}\sqrt{\frac{\log(n)}{\varepsilon}}\right)\right)$ | [38] | Static | Lower bound |
| | $\Theta(m)$ | $\left(1+\eta, O\left(\frac{\log^2(n)}{\eta\varepsilon}\right)\right)$ | [40] | Insertion Only | Upper Bound Density Only |
| | $\tilde{O}\left(\frac{n}{\eta^2}\right)$ | $(1+\eta, 0)$ | [64] [36] | Non-Private Streaming | Upper Bound |
| Maximum Matching Size | $O\left(\frac{\log^2(n)\log(\tilde{\alpha})}{\eta^2\varepsilon}\right)$ | $\left((1+\eta)(2+\tilde{\alpha}), O\left(\frac{\log^2(n)}{\eta\varepsilon}\right)\right)$ | Theorem 18 | Insertion Only | Upper Bound Arboricity |
| | $\Theta(m)$ | $\left(1+\eta, O\left(\frac{\log^2(n)}{\eta\varepsilon}\right)\right)$ | [40] | Insertion Only | Upper Bound |
| | | $(1, \Omega(\log(T)))$ | [40] | Insertion Only | Lower bound |
| | | $\left(1, \Omega\left(\min\left(\sqrt{\frac{n}{\varepsilon}}, \frac{T^{1/4}}{\varepsilon^{3/4}}, n, T\right)\right)\right)$ | Theorem 19 | Fully Dynamic | Lower Bound |
| | | $(1, \Omega(\log(T)))$ | [40] | Fully Dynamic | Lower Bound |
| | $O\left(\frac{\log(n)\log(\alpha)}{\eta^2}\right)$ | $((1+\eta)(2+\alpha), 0)$ | [65] | Non-Private Streaming | Upper Bound Arboricity |
| Triangle Count | | $\left(1, \Omega\left(\min\left(\sqrt{\frac{n}{\varepsilon}}, \frac{T^{1/4}}{\varepsilon^{3/4}}, n, T\right)\right)\right)$ | Section 8.1 | Fully Dynamic | Lower Bound |
| | | $(1, \Omega(\log(T)))$ | [40] | Fully Dynamic | Lower Bound |
| Connected Components Counting | | $\left(1, \Omega\left(\min\left(\sqrt{\frac{n}{\varepsilon}}, \frac{T^{1/4}}{\varepsilon^{3/4}}, n, T\right)\right)\right)$ | Theorem 20 | Fully Dynamic | Lower Bound |

**Maximum Matching (Section 7).** We give the following results for estimating the size of a maximum matching in bounded arboricity graphs. Arboricity, a measure of local sparsity in the input graph, is formally defined as the minimum number of forests into which a graph's edges can be partitioned. In real-world graphs, the arboricity of a graph is typically $\text{poly}(\log(n))$. In this setting, $\tilde{\alpha} > 0$ is a given public bound on the maximum arboricity $\alpha$ of the graph given by the input stream.[4] We also briefly sketch how to remove this assumption in the full version [34] at the cost of more space and a worse approximation guarantee. The additive error of our algorithms matches the information-theoretic lower bound in bounded arboricity graphs up to logarithmic factors.

Our matching algorithm falls under the *truly sublinear model*, where space usage is sublinear in the *number of nodes* in the graph. The truly sublinear model is the most difficult streaming model to obtain algorithms, and very few graph algorithms, even in the non-private setting, use truly sublinear space. We give an edge-DP algorithm for estimating the size of a maximum matching in bounded arboricity graphs. Our approximation guarantees hold when our public bound $\tilde{\alpha}$ upper bounds the arboricity of the graph, i.e. $\tilde{\alpha} \geq \alpha$, while our privacy guarantees always hold.

▶ **Theorem 4** (See Theorem 18). *Given a public bound $\tilde{\alpha}$, we obtain an $\varepsilon$-edge differentially private approximate maximum cardinality matching algorithm that outputs an approximate matching size using $O\left(\frac{\log^2(n)\log(\tilde{\alpha})}{\eta^2\varepsilon}\right)$ space in the insertion-only continual release model. If $\tilde{\alpha} \geq \alpha$, where $\alpha$ is the (private) maximum arboricity of the input graph, then our algorithm returns a $\left((1+\eta)(2+\tilde{\alpha}), O\left(\frac{\log^2(n)}{\eta\varepsilon}\right)\right)$-approximation of the size of the maximum matching.*

**Fully Dynamic Lower Bounds (Section 8).** Last but not least, we give improved lower bounds in the fully-dynamic setting in Section 8 for estimating the size of a maximum matching and even simpler problems like connected component counting and triangle counting. Our reductions follow a natural framework that generalizes to further problems (Section 8.1).

Note that previous lower bounds all show a $\Omega(\text{poly}\log(T))$ lower bound in the additive error while we strengthen these lower bounds to $\Omega(\text{poly}(T))$, yielding an exponential improvement. Comparing the upper bounds with our new lower bounds for the same problems yield polynomial separations in the error between the insertion-only and fully dynamic settings.

See Table 1 for a summary of our results and previous works.

## 2 Technical Overview

In this section, we provide a detailed overview of our techniques. We begin in Section 2.1 by illustrating the various algorithmic challenges with the graph continual release model. Section 2.2 then describes how we address these challenges.

### 2.1 Challenges in Continual Release

**Returning Multiple Solutions.** The best one-pass streaming algorithms in the non-private setting for the problems we study [37, 65, 64, 36] return the solution only once, at the end of the stream. In the continual release setting, we must return a solution *after every update*. This is a fundamental challenge, especially when considering privacy. Indeed, with a single solution release, each edge update is intuitively used only once to produce the solution. In

[4] Such public bounds are commonly assumed in non-private streaming literature [18, 65].

contrast, releasing solutions multiple times means earlier updates are potentially used many times to produce solutions for each release. Such overuse could result in greater loss of privacy. Thus, for our continual release algorithms, we use novel techniques to simultaneously ensure we achieve edge-privacy for the entire vector of releases as well as ensure our utility bounds for each release.

Previous value-only continual release algorithms ensure privacy over multiple releases via the sparse vector technique (SVT). Their algorithms release the outputs of one or more instances of SVT over the stream and the estimates are a deterministic function of the SVT outputs. Thus, the proof of privacy directly follows from the privacy guarantees of SVT. The outputs of our algorithms cannot be directly derived from standard SVT as we need to output information about individual vertices (as part of vertex subsets or individual outputs for each vertex for $k$-core decomposition). Moreover, our outputs depend on our sampling procedures, which in turn depend on private properties of the stream. These subtleties prevent the use of straightforward analyses of privacy using composition. Instead, we must directly prove the privacy of our algorithms using the definition of edge-privacy and analyze the output distributions of our algorithms over the entire stream. This is accomplished via careful conditioning arguments.

**Ensuring Sublinear Space.** Obtaining sublinear space continual release algorithms presents a new set of challenges. In particular, we must ensure that our sampling techniques are *stable*.[5] Stable sampling algorithms use sampling techniques that do not cause our samples from neighboring streams to differ by more than one edge update. For randomized sampling algorithms, this means that the distribution of sampled edges from edge-neighboring streams should be similar. Consider the following naive sampling scheme that is not stable. Suppose we want to sample each edge insertion in an insertion-only stream with probability proportional to the number of edges we have seen so far. Thus, the $i$-th inserted edge will be sampled with probability $1/i$. Suppose we have two neighboring streams $S$ and $S'$ where the edge that differs is the first update; thus, the first update in $S'$ is an edge insertion, while the first update in $S$ is $\perp$ (an empty update). Then, the $j$-th update in $S$, if it is an edge insertion, has a higher probability of being sampled than the $j$-th update in $S'$. Such a discrepancy produces significantly different output distributions when sampling from neighboring streams, which may lead to sampled graphs that significantly differ.

Hence, a major component of our algorithms and proofs is to formulate and prove stable sampling algorithms with similar output distributions on edge-neighboring streams. Common sampling approaches in the non-private streaming and sparsification literature do not immediately translate to stable algorithms in the continual release setting.

***k*-Core Decomposition.** Consider the $k$-core decomposition problem which is defined in terms of the *cores* of a graph. A $k$-core is a maximal induced subgraph where every vertex in the induced subgraph has degree at least $k$. Then, a vertex $v$'s core number is the largest value of $k$ for which $v$ is part of the $k$-core. In edge-neighboring graphs, the removal of one edge can change the core numbers of *all* vertices. Hence, we cannot use a function that computes the exact core number of each node in the continual release setting, as this can incur $\Omega(n)$ error through composition over the vertices. In the continual release setting, the insertion of a single edge can alter the core number of one or more vertices; hence, we

[5] This term was also used in node-private continual release literature [56], i.e. edge-to-edge and node-to-edge stability.

also cannot run a static differentially private $k$-core decomposition algorithm each time a core number changes since we can potentially see $\Omega(n)$ core number changes incurred over all vertices. Calling the static DP algorithm $\Omega(n)$ times would result in $\Omega(n)$ error through composition over the calls. To overcome these challenges, we develop a novel $k$-core decomposition algorithm using several new techniques (Section 5).

**Releasing Vertex Subsets.** Now, consider the setting where we return vertex subset solutions. Unlike returning real-valued solutions, even a few updates that do not significantly alter the solution's value (e.g. the maximum density in the densest subgraph problem) can still cause the entire set of vertices to change. This can happen even when the global sensitivity is small. Such behavior differs from real-valued solutions, which are less susceptible to large changes due to small updates. To give a concrete example, consider the densest subgraph problem where the goal is to obtain a subset of vertices with maximum induced density. Suppose we are given a number of edge insertions resulting in two disjoint almost $k$-cliques, $A$ and $B$, that is, two $k$-cliques, each with 2 edges removed. A sequence of insertions – one into $A$ followed by two into $B$ – could lead to large changes in the set of vertices in the densest subgraph. After the first insertion, the densest subgraph consists of vertices in $A$; then, after the third insertion, the densest subgraph consists of vertices in $B$. This sequence of only 3 insertions causes a complete change in the optimal vertex subset. It is not hard to extend this example and show that we can force any $O(1)$-approximation algorithm to completely change its output with $O(1)$ insertions. Because of this challenge and the privacy requirement, we allow for $(1+\eta)$-multiplicative approximations along with $\text{poly}(\log n)/\varepsilon$ additive error on the density of the subgraph induced by the vertex subsets we release.

To release a private densest subgraph, we can use a static $\varepsilon$-edge DP densest subgraph algorithm that releases the vertex subset. However, as mentioned above, we cannot afford to release a new private vertex subset each time we receive an update, even if the true vertex subset changes, since we would incur $\Omega(n)$ error from composition. Thus, our algorithms in Section 6 determine appropriate timesteps for releasing new vertex subsets.

## 2.2 Detailed Description of Our Techniques

All of our algorithms use $\tilde{O}(n)$ space, i.e., near-linear in the number of vertices in the stream, and some are sublinear in the number of vertices. We present $\varepsilon$-differentially private algorithms in edge-neighboring streams (Definition 6), where the streams differ by one edge update.

We now describe how we solve each of the aforementioned challenges in Section 2.1 and obtain algorithms for a variety of problems in the next sections. Each of these problems uses a unique sampling scheme as sparsifiers in graph literature are often problem-dependent. We prove the privacy of every algorithm as well as the utility of each of our algorithms within their individual sections. We summarize on a high level our technical contributions for each of the following problems.

***k*-Core Decomposition (Section 5).** Given a stream of updates $S$, the $k$-core decomposition problem asks for a number for each vertex after every update indicating the maximum value $k$ for which the vertex is contained in a $k$-core. A $k$-core is defined as a maximal set of vertices $U \subseteq V$ such that the degree of every vertex in the subgraph induced by $U$ is at least $k$. We return approximate core numbers for every vertex at every timestep $t$ that are $\left(2+\eta, O\left(\frac{\log^3(n)}{\eta^2\varepsilon}\right)\right)$-approximations of their true core numbers (see Theorem 1) while using $\tilde{O}\left(\frac{n}{\eta^4\varepsilon}\right)$ total space.

We formulate a novel differentially private continual release algorithm for $k$-core decomposition that is loosely inspired by the level data structure of the static $\left(2+\eta, O\left(\frac{\log^3(n)}{\eta^2\varepsilon}\right)\right)$-approximation algorithm of [21]. As described in Section 2.1, we cannot use the static private algorithm in a black-box manner since we need to return the value of *every* vertex at each timestep. Every vertex may change its core number many times throughout the duration of the stream. Thus, using any static $k$-core decomposition algorithm in a black-box manner incurs a privacy loss of a factor of $n$ via composition.

In the static setting, the level data structure algorithm for $k$-core decomposition partitions the vertices of the input graph across the levels based on the induced degree of each vertex among the vertices in the same or higher levels. Then, the levels roughly partition the vertices into cores of the graph with higher levels containing larger valued cores and lower levels containing smaller valued cores. In the static setting, vertices move up levels using all of the edges in the graph for each move. However, in the continual release setting, edges are inserted into the graph progressively and vertices move up the levels as new edges are inserted. In the static setting, each vertex releases their level at most $\text{poly}(\log n)$ times, whereas in the continual release setting, each vertex releases its approximate core number $\Omega(T) = \text{poly}(n)$ times. While composition over $\text{poly}(\log n)$ releases leads to $\text{poly}(\log n)$ additive error, composition over $\Omega(n)$ releases in the continual release model would lead to $\Omega(n)$ error.

Due to the above challenge, we need to be able to have vertices release their approximate core numbers $\Omega(T)$ times without losing a factor of $T$ in the additive error due to composition. This problem is not present in the static setting. In our novel continual release algorithm, we use a variant of the *sparse vector technique* adapted to the level data structure. In particular, we use the idea of the *multidimensional sparse vector technique* given in [20] and adapt it to the continual release setting with a new formulation of the algorithm and a new proof allowing for adaptive queries based on new (subsampled) edge insertions to the graph. We call this the *adaptive multidimensional sparse vector technique*.

In the static setting, we can use the multidimensional sparse vector technique to determine when a vertex stops moving up levels and the level in which a vertex resides directly corresponds with an approximation of the core number of the vertex. Thus, once a vertex stops moving up levels in the static setting, it will immediately output a new approximation. Hence, each vertex fails the SVT check at most once. However, in the continual release setting, our usage of the adaptive multidimensional sparse vector technique requires that each vertex must output a value at every timestep. This means that the adaptive multidimensional sparse vector technique must answer adaptive queries resulting from edge insertions that occur after previous levels are released. Moreover, the mechanism needs to allow for $\text{poly}(\log n)$ above-threshold checks for each vertex, indicating whether the vertex moved up one of the $\text{poly}(\log n)$ levels.

In addition to solving the above challenge, we further present a sublinear space algorithm for $k$-core decomposition, which uses $\tilde{O}\left(\frac{n}{\eta^4\varepsilon}\right)$ space. The main workhorse is a *sparse* level data structure. No sparsified versions of the level data structure existed in either the non-private graph literature or the private graph literature. While many non-private sparsification algorithms in the streaming model sample edges uniformly at random, such sampling techniques are *insufficient* for $k$-core decomposition. When all edges are sampled with the same fixed probability, vertices in smaller cores are expected to have no adjacent edges included in the sample. If no adjacent edges are included in the sample for vertex $v$, then we cannot approximate $v$'s core number. Hence, we employ a more sophisticated sampling algorithm; specifically, we sample each inserted edge in the stream with probability

inversely proportional to the level of its lower-level endpoint. This means that edges on higher levels are sampled with a smaller probability than edges on lower levels. In particular, such a sampling scheme ensures that edges in larger valued cores–which inherently contain more edges–are sampled using smaller probabilities than smaller valued cores. Hence, such a sampling scheme simultaneously ensures sublinear space while maintaining a large enough sample of adjacent edges to each vertex to satisfy our approximation bounds on the core numbers of *all* vertices.

Finally, we prove our approximation bounds using a careful Chernoff bound argument on our sampled sets of edges that takes into account the noise resulting from our usage of DP mechanisms, including the adaptive multidimensional sparse vector technique. Our approximation bound uses an original analysis that shows that vertices on a given level in our sparsified structure will be on approximately the same level, with high probability, as in the non-sparsified structure. Thus, the approximation bounds that we obtain from our non-sparsified structure also hold for our sparsified structure.

**Densest Subgraph (DSG) (Section 6).** Given an edge-neighboring stream and timestamp $t$, the densest subgraph in $G_t$ is a subset of vertices $V_{\text{OPT}}$ which maximizes the density of the induced subgraph, $V_{\text{OPT}} = \arg\max_{V' \subseteq V} \left(\frac{|E_t(V')|}{|V'|}\right)$. We return a subset of vertices $V_{approx}$ whose induced subgraph gives a $\left(1+\eta, O\left(\frac{\log^5(n)}{\eta^4 \varepsilon}\right)\right)$-approximation of the density of the densest subgraph (see Table 1) in $\tilde{O}\left(\frac{n}{\eta^5 \varepsilon}\right)$ space. Our algorithm calls the static $\varepsilon$-DP DSG algorithm of [21] in a black-box manner. Using a different static $\varepsilon$-edge DP DSG algorithm [20], we can obtain a $\left(2+\eta, O\left(\frac{\log^2(n)}{\eta \varepsilon}\right)\right)$-approximation while saving a polylogarithmic factor in space.

Our continual release densest subgraph algorithm takes inspiration from the non-private $(1+\eta)$-approximate sparsification algorithms of [36, 64], although we must solve several crucial challenges (Section 2.1) to ensure privacy. Indeed, to ensure accurate approximation guarantees, we present a novel concentration bound proof based on an intricate Chernoff bound argument that accounts for errors introduced by our various private subroutines. Previous works in the continual release model only release the *value* of the densest subgraph and not a *vertex subset* [40, 56]. Since the density of the densest subgraph has sensitivity 1, previous works use the sparse vector technique [30, 75, 47, 62] and private prefix sums to release these values after adding appropriate Laplace noise. In our work, we release a private set of vertices at timestep $t$ whose induced subgraph is an approximation of the densest subgraph in $G_t$.[6] We use the sparse vector technique to determine when to release a new private set of vertices, and we use one of the many existing differentially private densest subgraph algorithms [38, 20, 21, 70, 22] to return a vertex subset in the sparsified graph. However, there are several challenges in simply returning the subgraph obtained from these static algorithms. First, compared to the non-private setting, we cannot obtain an exact densest subgraph in the sparsified graph. Hence, we must ensure that neither the additive nor multiplicative error blows up when the subgraph obtained in the sparsified graph is scaled up to the original graph. To solve this issue, we ensure that the first subgraph we release has size (approximately) at least the additive error returned by the private static algorithms.

There is another important challenge when making our algorithm use sublinear space. Previous non-private algorithms [64, 36] sample using a *fixed* probability because 1) they assume knowledge of the total number of updates in the stream and 2) only produce the

[6] the induced subgraph consists of all updates that occurred on or before timestep $t$

approximate densest subgraph once, at the end of the stream. However, in our setting, we must release an approximate densest subgraph after each update. Furthermore, we do not know the total number of edges in the stream. Thus, we adaptively adjust our probability of sampling as we see more edge insertions. This must be carefully implemented since a naive adjustment procedure may produce an unstable algorithm (as described in Section 2.1).

Towards solving all of these challenges, our sparsification algorithm works as follows. We sparsify the stream by sampling each edge in the stream with probability $p_t$. Earlier on in the stream, our probability of sampling should be set high enough in order to sample enough edges to ensure our concentration bounds. However, as we see more edges, we reduce the probability of sampling to ensure our sublinear space bounds. In edge-neighboring streams, such reductions in probability may not occur at the same time, thus leading to sparsified graphs that are *not* stable. Hence, we use the sparse vector technique to determine when to reduce our probability of sampling an edge. We ensure privacy based on a careful conditioning on the sampling probabilities. Intuitively, the sparse vector technique ensures that we decrease our sampling probability at the same timestamps in edge-neighboring streams. Conditioning on the timestamps when the sampling probabilities are reduced, we can show that our sampled streams are stable.

Unlike the static setting where the privacy analysis of algorithms that rely on an SVT instance can apply simple composition, we need to derive a probabilistic proof "from scratch" since we must output a solution after every update but only lose privacy on timestamps where the SVT answers "above". Furthermore, our analysis must take into account our usage of two SVTs, one for determining the probabilities of sampling and the other for determining when we release a new vertex subset. The second use of SVT depends on the outputs of the first; that is, determining when to release a new vertex subset depends on our sampling probabilities. Such SVTs lead to errors that affect our approximation guarantees and our careful Chernoff bound argument must account for these errors while maintaining our approximation guarantees.

**Maximum Cardinality Matching Size (Section 7).** Given a stream of updates $S$, the maximum cardinality matching problem asks for a real-valued integer at each timestep $t$ equal to the size of the maximum matching in $G_t$. We give an algorithm that uses sublinear space in the number of vertices in the graph with edge-DP guarantees in the continual release model. We provide approximation guarantees using a certain property of the input graph stream: the maximum arboricity of the stream. The arboricity of a graph is a measure of local sparsity. Formally, a graph's arboricity, $\alpha$, is the minimum number of forests into which its edges can be partitioned. Our algorithm uses a public bound $\tilde{\alpha}$. If this public bound upper bounds the private maximum arboricity of the input stream, our algorithm gives approximation guarantees in terms of $\tilde{\alpha}$. We emphasize that our algorithm is always private for all inputs. Specifically, if $\tilde{\alpha}$ upper bounds the private maximum arboricity of the input stream, then we obtain a $\left((1+\eta)(2+\tilde{\alpha}), O\left(\frac{\log^2 n}{\eta\varepsilon}\right)\right)$-approximation of the matching size.

For this algorithm, we take inspiration from the sublinear space non-private $((1+\eta)(2+\alpha))$-approximate maximum matching algorithm of [65], which also requires an upper bound $\alpha$ on the maximum arboricity of the stream. Their approximation guarantees are given in terms of this upper bound. To the best of our knowledge, [65] is the only truly sublinear matching algorithm beyond bipartite graphs that terminates after a single pass, as required by continual release.

The key observation that [65] makes is that maintaining $\Omega(\log n/\eta^2)$ edge samples, sampled uniformly at random in the stream, and counting the number of edges in the sample which are adjacent to at most $\alpha + 1$ edges occurring later in the stream is sufficient for providing a good estimate of the maximum matching size in terms of the arboricity upper bound. Edges that are adjacent to more than $\alpha + 1$ edges that occur later in the stream are discarded from the sample. Unfortunately, in edge-neighboring streams, we cannot add noise to the counts of every edge in the sample since the sensitivity of such counts is $\Omega(\tilde{\alpha})$ (our public bound). Instead, we show through a precise charging argument that the sensitivity of the size of the sample is bounded by 2. Since the sensitivity of the size of the sample is bounded by 2, we can use the sparse vector technique (SVT) to determine when to release a new estimate when the estimate changes by a large amount. Using SVT results in noisy sample sizes which requires a new analysis for the utility bounds.

Previous approximation bound analyses [65] relied on the intuition that edge samples approximately fall into buckets determined by the sampling probability. However, with the use of SVT, it is no longer clear which buckets the sampled edges fall into as SVT introduces an additive $O(\log(n)/\varepsilon)$ noise. Thus, to prove our approximation bounds, we first argue that SVT only causes an additive error of $\tilde{O}\left(\frac{\log^2(n)}{\eta\varepsilon}\right)$ on the estimated number of sampled edges; hence, initializing a sufficiently large valued bucket ensures that the bucket for sampled edges with noise does not deviate by too much from the bucket the sampled edges would fall into without noise. The proof of this property requires a detailed casework analysis. We can further remove the assumption of $\tilde{\alpha}$ using additional space via a parallel guessing trick in the full version [34].

**Fully Dynamic Lower Bounds (Section 8).** We show polynomial lower bounds for the maximum matching, triangle counting, and connected components problems in fully dynamic streams. Our lower bounds reduce each of these graph problems via novel graph constructions to answering inner product queries [23, 19, 28, 66]. The known lower bounds for inner product queries then apply to our problems. We encode the secret dataset given by the inner product query problem via $\Theta(n)$ insertions to construct our graph. Then, we map the value of the answers to each of our graph problems to the original inner product query via the inclusion-exclusion principle. Finally, we delete all insertions through edge deletions and repeat the process for the next query. We remark that this technique is quite general and likely extends to many other natural graph problems. Roughly speaking, the only problem-specific requisite is that we can encode the bits of a secret dataset using the problem structure and that we can easily flip the bits by adding and deleting edges. For example, we can encode each bit of a secret dataset as a (non-)edge in an induced matching so that flipping a bit equates to (adding) deleting the (non-)edge.

## 3 Related Works

**Streaming Algorithms.** The streaming model of computation is a prominent model for large-scale data analysis that has been studied for multiple decades [67]. In this model, one usually seeks space- and time-efficient algorithms that can process the stream on the fly without the need to store all data. A long line of work in this area includes classical results such as the Flajolet-Martin algorithm for counting distinct elements [44] and many others [43, 2]. In the context of streaming graph algorithms, ideally one would like to obtain space sublinear in the number of vertices [65, 52]; but for many graph problems it is necessary to work in the semi-streaming model settling on space near-linear

in the number of vertices [6, 1, 37, 39, 42, 64]. Online streaming algorithms release graph statistics after every update while maintaining the low space bounds [16, 46, 17, 45]. In the non-private streaming setting, there exists many works on graph algorithms, including approximating the size of the maximum matching [1, 7, 4, 42, 65], vertex cover [5], densest subgraph [36, 64], $k$-core decomposition [37, 59], number of connected components [8], and others [6, 1, 39, 42, 52, 51, 69, 63]

**Continual Release Model.** In the context of streaming computation, the DP model of reference is the continual release model [29, 14] where we require algorithms to abide by a strong privacy notion: an observer obtaining *all* future outputs of the algorithm must in essence learn almost nothing about the existence of any single input. Since its introduction, this research area has received vast attention outside of graphs, including many recent works (see e.g. [13, 50, 49, 41, 54, 55]).

Among the insertion-only continual release work, prior research has tackled classical estimation problems [14, 50, 49], as well as heavy hitters-related problems [13, 35]. More recent works tackle the fully dynamic continual release setting [54, 25, 40]. Particularly relevant is [40], which shows hardness results for graph estimation in fully dynamic streams. Our work contributes new hardness results in this new emerging area as well.

Most relevant to our paper is the literature on continual release algorithms in graphs [81, 40, 77, 56]. In this area, [77] studied graph statistics (degree distribution, subgraph counts, etc.) on bounded-degree graphs. [40] focused on estimating a number of graph statistics like maximum matching, triangle counting, and the density of the densest subgraph for both edge and node privacy; they provide approximation guarantees in terms of the maximum degree in the graph as well as lower bounds in insertion-only and fully dynamic streams. [56] explored counting problems in graphs (counting edges, triangles, stars, connected components) for node-privacy and where privacy must hold for arbitrary graphs (e.g., graphs with arbitrary degrees). They give time-aware projection algorithms that can transform any continual release algorithm that gives approximation guarantees for bounded degree graphs into a truly private algorithm on nearly bounded degree graphs. Like in [56] for our algorithms that assume a public bound, say on the stream arboricity, this bound affects only the approximation guarantees but not the privacy claims, hence our algorithms satisfy their notion of *truly private* [56].

**Private Graph Algorithms.** The private literature on graph algorithms includes a large body of work on static graph algorithms (see e.g. [3, 10, 22, 21, 33, 58, 57, 60, 68, 71, 73, 72, 79] and references therein). Aside from the problems we study, work in this area includes results on preserving graph cuts [3], the stochastic block model recovery [15], graph clustering [12, 53], and many other areas.

Our paper focuses on the pure, $\varepsilon$-DP setting. In this setting, there has been a variety of recent works that we study for static graphs. The densest subgraph (DSG) problem has been extensively studied in the non-private streaming context [9, 36, 64]. Our work builds on the results of [36, 64], which show that edge sampling approximately preserves the density of the densest subgraph. In the context of privacy, beyond the already cited work of [40] that focused on density estimation only, all other works are in the static DP or LEDP setting [70, 38, 21, 22, 20]. In the $\varepsilon$-DP setting, the best-known lower bound on the additive error of the densest subgraph problem is $\Omega(\sqrt{\varepsilon^{-1}\log n})$ [38, 70]. Comparatively, the best known upper bounds in the central $\varepsilon$-DP setting are the $(2, O(\varepsilon^{-1}\log n))$-approximation [20], $(2+\eta, O(\varepsilon^{-1}\log^2 n))$-approximate [22], and the $(1+\eta, O(\varepsilon^{-1}\log^4 n))$-approximation [21] upper

bounds. In the $\varepsilon$-LEDP setting, the best known upper bounds are the $(2+\eta, O(\eta^{-1}\log^2 n))$-approximate [22] and $O(2, O(\varepsilon^{-1}\log n))$-approximate [20] bounds. Our paper presents the first continual release algorithm for releasing an actual approximate densest subgraph (as opposed to estimating its density) in edge-private insertion-only streams.

Related to the densest subgraph problem is the $k$-core decomposition of the graph for which some streaming results are known [37, 59, 76]. In the private setting, this problem has been tackled by [21, 20, 48] where the best error bound achieved in both the central $\varepsilon$-DP and $\varepsilon$-LEDP settings provide $(1, O(\varepsilon^{-1}\log n))$-approximations of the core number [20]. This is tight against the recently shown lower bound of $\Omega(\varepsilon^{-1}\log n)$ [48]. In our paper, we provide the first continual release algorithm for approximating the core number of nodes in a graph.

## 3.1 Concurrent Work

Recent concurrent work of [74] study edge-DP algorithms in the fully dynamic continual release model. They give a number of upper and lower bounds for a variety of problems, including triangle counting, connected component counting, maximum matching size, and degree histogram. They consider *event-level* and *item-level* privacy. *Event-level* describes fully dynamic neighboring streams where one update differs, while *item-level* describes neighboring streams where all updates on a particular edge differ. Note that for insertion-only streams with no duplicate insertions, both settings describe the same set of neighboring streams. They give the first $\text{poly}(T, n)$ error lower bounds for $(\varepsilon, \delta)$-DP graph algorithms in continual release for $\delta > 0$; previous works only considered $\delta = 0$ in their lower bounds. Their item-level upper bounds are tight against their lower bounds, with additive error proportional to $\text{poly}(T, n)$.

Our paper considers different problems (densest subgraph and $k$-core decomposition) and focuses on designing algorithms with $\text{poly}(\log n)$ additive errors in insertion-only streams. [74] focuses on fully dynamic streams which in general require additive $O(\text{poly}(T, n))$ error (as demonstrated by [74] and our lower bounds in Section 8). Furthermore, our paper focuses on the sublinear space setting and returning vertex subset solutions, two settings not discussed in [74].

# 4 Preliminaries

We now introduce the notation we use in this paper as well as some definitions. Further standard preliminaries are deferred to the full version [34].

## 4.1 Setting & Notation

A graph $G = (V, E)$ consists of a set of vertices $V$ and a set of edges $E$ where edge $\{u, v\} \in E$ if and only if there is an edge between $u \in V$ and $v \in V$. We write $n := |V|$ and $m := |E|$.

We consider the insertion-only setting within the continual release model, where we begin with an empty graph $G_0 = (V, E_0)$ where $E_0 = \varnothing$ and edge updates in the form of $\{e_t, \perp\}$ arrive in a stream at every timestep $t \in [T]$ where each non-empty edge is inserted at most once. We are required to release an output for the problem of interest after every (possibly empty) update. Our goal is to obtain algorithms that achieve sublinear space, either in the number of edges $\tilde{o}(m)$ (note that $T \geq m$ as we allow for empty updates), or the number of vertices $\tilde{o}(n)$. We assume $T \leq n^c$ for some absolute constant $c \geq 1$ to simplify our presentation. If there are no empty edge updates then it is sufficient to consider $c = 2$.

Throughout the paper, we write $\eta \in (0, 1]$ to denote a fixed multiplicative approximation parameter. We assume $\eta$ is a fixed constant that is ignored under asymptotic notation. Constants used throughout the paper may depend on $\eta$.

We use $\mathcal{M}$ to denote a mechanism, $M(\cdot)$ to denote the size of a maximum matching, and $\mu$ to denote the mean of a random variable.

## 4.2 Continual Release

We use the phrasing of the below definitions as given in [56].

▶ **Definition 5** (Graph Stream [56]). *In the continual release model, a graph stream $S \in \mathcal{S}^T$ of length $T$ is a $T$-element vector where the $i$-th element is an edge update $u_i = \{v, w, insert\}$ indicating an edge insertion of edge $\{v, w\}$, $u_i = \{v, w, delete\}$ indicating an edge deletion of edge $\{v, w\}$, or $\perp$ (an empty operation).*

We use $G_t$ and $E_t$ to denote the graph induced by the set of set of updates in the stream $S$ up to and including update $t$. Now, we define neighboring streams as follows. Intuitively, two graph streams are edge neighbors if one can be obtained from the other by removing one edge update (replacing the edge update by an empty update in a single timestep).

▶ **Definition 6** (Edge Neighboring Streams). *Two streams of edge updates, $S = [u_1, \ldots, u_T]$ and $S' = [u'_1, \ldots, u'_T]$, are* edge-neighboring *if there exists exactly one timestamp $t^* \in [T]$ where $u_{t^*} \neq u'_{t^*}$ and for all $t \neq t^* \in [T]$, it holds that $u_t = u'_t$. Streams may contain any number of empty updates, i.e. $u_t = \perp$. Without loss of generality, we assume for the updates $u_{t^*}$ and $u'_{t^*}$ that $u'_{t^*} = \perp$ and $u_{t^*} = \pm e_{t^*}$ is an edge insertion or deletion.*

The notion of neighboring streams leads to the following definition of an edge differentially private algorithm.

▶ **Definition 7** (Edge Differential Privacy). *Let $\varepsilon, \delta \in (0, 1)$. An algorithm $\mathcal{A}(S) : \mathcal{S}^T \to \mathcal{Y}^T$ that takes as input a graph stream $S \in \mathcal{S}^T$ is said to be $(\varepsilon, \delta)$-*edge differentially private (DP) *if for any pair of edge-neighboring graph streams $S, S'$ that differ by 1 edge update and for every $T$-sized vector of outcomes $Y \subseteq \text{Range}(\mathcal{A})$, $\mathbf{P}[\mathcal{A}(S) \in Y] \leq e^{\varepsilon} \cdot \mathbf{P}[\mathcal{A}(S') \in Y] + \delta$. When $\delta = 0$, we say that $\mathcal{A}$ is $\varepsilon$-*edge DP*.*

We now formalize the concepts of edge edit distance between streams and the concept of *stability* under sparsification.

▶ **Definition 8** (Edge Edit Distance). *Given two streams $S, S' \in \mathcal{S}^T$, the edge edit distance between the two streams is the shortest chain of graph streams $S_0, S_1, \ldots, S_d$ where $S_0 = S$ and $S_d = S'$ where every adjacent pair of streams in the chain are edge-neighboring. The edge edit distance is $d$, the length of the chain.*

An algorithm that takes as input a stream $S$ and outputs a chosen set of updates from the stream is *stable* if on edge-neighboring streams $S$ and $S'$, there exists a coupling[7] between the randomness used in the algorithm on the inputs such that the edge distance between the output streams is $O(1)$.

## 4.3 Differential Privacy Tools

Here, we define the privacy tools commonly used in differential privacy in terms of the continual release model. Throughout the paper, we use some standard privacy mechanisms as building blocks (see [31] for a reference).

[7] A *coupling* of random variables $(X, Y)$ is a joint distribution such that the marginal distributions correspond to $X, Y$ respectively.

▶ **Definition 9** (Global Sensitivity). *The global sensitivity of a function $f : \mathcal{D} \to \mathbb{R}^d$ is defined by $\Delta_f = \max_{D,D' \in \mathcal{D}, D \sim D'} \|f(D) - f(D')\|_1$, where $D \sim D'$ are neighboring datasets and differ by an element.*

▶ **Definition 10** (Laplace Distribution). *We say a random variable $X$ is drawn from a Laplace distribution with mean $\mu$ and scale $b > 0$ if the probability density function of $X$ at $x$ is $\frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$. We use the notation $X \sim \mathsf{Lap}(b)$ to denote that $X$ is drawn from the Laplace distribution with scale $b$ and mean $\mu = 0$.*

The Laplace mechanism for $f : X \to \mathbb{R}$ with global sensitivity $\sigma$ adds Laplace noise to the output of $f$ with scale $b = \sigma/\varepsilon$ before releasing.

▶ **Proposition 11** ([31]). *The Laplace mechanism is $\varepsilon$-DP.*

▶ **Theorem 12** (Adaptive Composition; [27, 26, 32]). *A sequence of DP algorithms, $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$, with privacy parameters $(\varepsilon_1, \ldots, \varepsilon_k)$ form at worst an $(\varepsilon_1 + \cdots + \varepsilon_k)$-DP algorithm under* adaptive composition *(where the adversary can adaptively select algorithms after seeing the output of previous algorithms).*

▶ **Theorem 13** (Group Privacy; Theorem 2.2 in [31]). *Given an $\varepsilon$-edge (node) differentially private algorithm, $\mathcal{A}$, for all pairs of input streams $S$ and $S'$, it holds that for all possible outcomes $Y \in \text{Range}(\mathcal{A})$, $e^{-k\varepsilon} \leq \frac{\mathbf{P}[\mathcal{A}(S') \in Y]}{\mathbf{P}[\mathcal{A}(S) \in Y]} \leq e^{k\varepsilon}$ where $k$ is the edge (node) edit distance between $S$ and $S'$.*

### 4.3.1 Sparse Vector Technique

Below, we define the *sparse vector technique* and give its privacy and approximation guarantees. The sparse vector technique is used to answer *above threshold* queries where an above threshold query checks whether the output of a function that operates on an input graph $G$ exceeds a threshold $T$.

We use the variant introduced by [62] and used by [40]. Let $D$ be an arbitrary (graph) dataset, $(f_t, \tau_t)$ a sequence of (possibly adaptive) query-threshold pairs, $\Delta$ an upper bound on the maximum sensitivity of all queries $f_t$, and an upper bound $c$ on the maximum number of queries to be answered "above". Typically, the AboveThreshold algorithm stops running at the first instance of the input exceeding the threshold, but we use the variant where the input can exceed the threshold at most $c$ times where $c$ is a parameter passed into the function.

Throughout this paper, we use the class $\text{SVT}(\varepsilon, \Delta, c)$ (Algorithm 4.1) where $\varepsilon$ is our privacy parameter, $\Delta$ is an upper bound on the maximum sensitivity of incoming queries, and $c$ is the maximum number of "above" queries we can make. The class provides a ProcessQuery($query, threshold$) function where *query* is the query to SVT and *threshold* is the threshold that we wish to check whether the query exceeds.

▶ **Theorem 14** (Theorem 2 in [62]). *Algorithm 4.1 is $\varepsilon$-DP.*

We remark that the version of SVT we employ (Algorithm 4.1) does not require us to resample the noise for the thresholds (Line 4) after each query but we do need to resample the noise (Line 9) for the queries after each query.

## 5 $k$-Core Decomposition

In this section, we introduce a semi-streaming sampling algorithm that preserves the $k$-cores in an input graph $G = (V, E)$ while ensuring privacy. Specifically, we have the following theorem.

**Algorithm 4.1** Sparse Vector Technique.

Input: privacy budget $\varepsilon$, upper bound on query sensitivity $\Delta$, maximum allowed "above" answers $c$
**Class** `SVT`$(\varepsilon, \Delta, c)$
  $\varepsilon_1, \varepsilon_2 \leftarrow \varepsilon/2$
  $\rho \leftarrow \mathsf{Lap}(\Delta/\varepsilon_1)$
  count $\leftarrow 0$
  **Function** `ProcessQuery`$(f_t(D), \tau_t)$
    **if** count $> c$ **then**
      **return** "abort"
    **if** $f_t(D) + \mathit{Lap}(2c\Delta/\varepsilon_2) \geq \tau_t + \rho$ **then**
      **return** "above"
      count $\leftarrow$ count $+1$
    **else**
      **return** "below"

▶ **Theorem 15** (Sublinear Space Private $k$-Core Decomposition). *Fix $\eta \in (0, 1]$. There is an $\varepsilon$-DP algorithm for the $k$-core decomposition problem in the continual release model for insertion-only streams. At every $t \in [T]$, the algorithm returns a value for each vertex, such that every value is a $\left(2+\eta, O\left(\eta^{-2}\varepsilon^{-1}\log^3(n)\right)\right)$-approximation of the corresponding vertex's true core value, with probability $1 - 1/\operatorname{poly}(n)$. The maximum space used is $O\left(\eta^{-4}\varepsilon^{-1}n\log^5 n\right)$, with probability $1 - 1/\operatorname{poly}(n)$.*

**Algorithm Intuition.** First, we give some intuition. Our algorithm essentially performs a sampling version of the classic peeling algorithm for $k$-core decomposition. The classic peeling algorithm successively *peels* (removes) vertices with the minimum degree until all vertices are removed from the graph. A core that is formed during the peeling process is the induced subgraph consisting of the remaining vertices after a vertex is peeled and the value of such a core is the minimum induced degree within the subgraph. The core number for each vertex $v$ is equal to the maximum valued core that $v$ is a part of during any stage of the peeling. A dynamic version of this algorithm can be obtained by maintaining a *level data structure* where a vertex is moved up a level if its induced degree among vertices in the same or higher levels is larger than a cutoff $C$. One can show that having $O(\log n)$ levels of the structure and appropriately setting $C$ among $O(\log n)$ duplicates of the structure gives a $(2+\eta)$-approximation of the core numbers of the nodes in the non-private, insertion-only setting [78, 61, 21]. Our main innovation is a private and sparse level data structure.

When sparsifying the graph, we cannot simply take a uniform sample of the edges adjacent to each vertex. An easy example to consider is a vertex $v$ which is part of a 10-clique and also adjacent to $n/2$ degree one vertices. A uniform sample of the edges adjacent to $v$ will most likely not discover the 9 edges connecting it to the 10-clique (for large $n$). We call the edges connecting $v$ to the 10-clique the set of *important* edges. Thus, we must take a smarter sample of edges adjacent to $v$. To maintain a sparsified, sampling-based version of the level data structure, we maintain samples of large enough size of the *up-edges* adjacent to each vertex. The up-edges adjacent to each vertex are the edges connecting each vertex to neighbors in the same or higher levels. Once we see enough sampled up-edges, we move

the vertex up one level and continue sampling edges until we either reach the topmost level or the vertex is adjacent to only a very small sample of up-edges. Such a sampling method allows us to keep enough of the important edges which connect to other vertices in higher valued cores.

Finally, to make the above algorithm differentially private, we use SVT to determine when to move the vertex up a level. We show that although many vertices may move up levels, we only lose privacy when the vertices that are adjacent to the edge that differs between neighboring streams move up. Since our total number of levels and duplicates is bounded by $O(\log^2 n)$, the privacy loss from SVT is also bounded by $O(\log^2 n)$.

**Analysis.** We provide the pseudocode and proof of Theorem 15 in the full version [34].

## 6 Densest Subgraph

In this section, we focus on the densest subgraph problem and provide the first differentially private algorithm for densest subgraph in the continual release model using space sublinear in the total number of edges in the graph.

▶ **Theorem 16** (Sublinear Space Private Densest Subgraph). *Fix $\eta \in (0, 1]$. There is an $\varepsilon$-edge differentially private algorithm for the densest subgraph problem in the continual release model for insertion-only streams. The algorithm returns a set of vertices whose induced subgraph is a $\left(2+\eta, O(\eta^{-1}\varepsilon^{-1}\log^2(n))\right)$-approximation of the densest subgraph in $G_t$, with probability at least $1 - 1/\operatorname{poly}(n)$, for all $t \in [T]$. The maximum space used is $O\left(\eta^{-2}\varepsilon^{-1}n\log^2(n)\right)$, with probability at least $1 - 1/\operatorname{poly}(n)$, for all $t \in [T]$.*

We can also reduce the multiplicative error to $(1+\eta)$ at the cost of increasing the space usage by a $\operatorname{poly}(\log(n))$ factor.

▶ **Theorem 17** (Sublinear Space Private Densest Subgraph). *Fix $\eta \in (0, 1]$. There exists an $\varepsilon$-edge differentially private algorithm for the densest subgraph problem in the continual release model for insertion-only streams. The algorithm returns a set of vertices whose induced subgraph is a $\left(1+\eta, \eta^{-4}\varepsilon^{-1}\log^5(n)\right)$-approximation of the densest subgraph in $G_t$, with probability at least $1-1/\operatorname{poly}(n)$, for all $t \in [T]$. The maximum space used is $O\left(\eta^{-5}\varepsilon^{-1}n\log^5(n)\right)$, with probability at least $1 - 1/\operatorname{poly}(n)$, for all $t \in [T]$.*

**Algorithm Intuition.** We revise the algorithms of [64] and [36] to the insertion-only continual release setting. On a high-level, our algorithm maintains a sample of edges over time and releases a DP set of vertices by running a black-box DP densest subgraph algorithm on the subgraph induced by the sample. At every timestep $t \in [T]$, an edge $e_t$ is sampled with probability $p$ – the sampling probability is initialized to 1 as in the beginning we can afford to store every edge. We privately check whether the number of edges seen so far exceeds a certain threshold using a sparse vector technique (SVT) query and adjust the sampling probability $p$ accordingly. In order to avoid privacy loss that grows linearly in $T$, we do not invoke the black-box DP densest subgraph algorithm at every timestep and instead invoke it only if the current density of the sample exceeds a certain threshold using another SVT query.

**Analysis.** The pseudocode and proof of Theorems 16 and 17 are deferred to the full version [34].

## 7 Maximum Matching

In this section we give an edge-DP algorithm for maximum matching that uses truly sublinear space. We adapt the algorithm of [65] to obtain a private approximate maximum cardinality algorithm in the continual release model using sublinear space in the number of vertices. As in their algorithm, we assume that we are provided with a public upper bound $\tilde{\alpha}$ on the maximum arboricity $\alpha$ of the graph at any point in the stream.

The privacy of our algorithm is always guaranteed. However, we do not assume that $\tilde{\alpha}$ is guaranteed to upper bound $\alpha$. When $\tilde{\alpha} \geq \alpha$, our approximation guarantees hold with high probability. Otherwise, our approximation guarantees do not necessarily hold. Note that the same type of guarantee holds for the original non-private streaming algorithm [65] where their utility guarantee is only given when their public estimate of $\alpha$ upper bounds the maximum arboricity of the input. We prove the following result in this section.

▶ **Theorem 18.** *Fix $\eta \in (0, 1]$. Given a public estimate $\tilde{\alpha}$ of the maximum arboricity $\alpha$ over the stream,*[8] *There is an $\varepsilon$-edge DP algorithm for estimating the size of the maximum matching in the continual release model for insertion-only streams. If $\tilde{\alpha} \geq \alpha$, then with probability at least $1 - 1/\operatorname{poly}(n)$, our algorithm returns a $\left((1+\eta)(2+\tilde{\alpha}), O(\eta^{-1}\varepsilon^{-1}\log^2(n))\right)$-approximation of the size of the maximum matching at every timestamp. Moreover, our algorithm uses $O(\eta^{-2}\varepsilon^{-1}\log^2(n)\log(\tilde{\alpha}))$ space with probability $1 - 1/\operatorname{poly}(n)$.*

**Algorithm Intuition.** We revise the algorithm of [65] to the insertion-only continual release setting. On a high-level, [65] showed that the cardinality of a carefully chosen subset of edges $F \subseteq E$ is a good estimator for the size of the maximum matching for graphs of bounded arboricity $\alpha$. $F$ is obtained from $E$ by deleting edges $e$ adjacent to a vertex $v$ if more than $\alpha$ other edges adjacent to $v$ arrived after $e$. Then their algorithm maintains a small sample $S \subseteq F$ throughout the algorithm by down-sampling edges when the current sample exceeds some threshold. Our algorithm follows a similar approach with two main adjustments to satisfy privacy. First, we release powers of $(1 + \eta)$ based on an SVT comparison against the current value of the estimator. Second, the decision to down-sample is also based on an SVT comparison against the threshold.

**Analysis.** The proof and pseudocode for Theorem 18 is deferred to the full version [34].

## 8 Lower Bounds for Fully Dynamic Streams

In this section, we establish lower bounds on the additive error of differentially private algorithms for estimating the size of a maximum matching and the number of connected components in the continual release model. Similar to the lower bound for counting distinct elements in the continual release model [54], we reduce the problem of answering matching queries and connected component queries to answering *inner product queries*. Then, we leverage known lower bounds for the inner product problem [23, 28, 66, 19] to obtain our lower bounds.

[8] We can eliminate this assumption with an additional pass of the stream or notify the observer when the utility guarantees no longer hold in the one-pass setting. See the full version [34].

▶ **Theorem 19.** *Fix $\varepsilon \in (0,1)$. If $\mathcal{A}$ is an $\varepsilon$-DP mechanism that answers maximum matching queries on graphs with $n$ vertices in the continual release model within additive error $\zeta$ with probability at least 0.99 for fully dynamic streams of length $T$, then $\zeta = \Omega\left(\min\left(\sqrt{\frac{n}{\varepsilon}}, \frac{T^{1/4}}{\varepsilon^{3/4}}, n, T\right)\right)$. Moreover, we may assume the graph is bipartite, has maximum degree 2, and arboricity 1.*

▶ **Theorem 20.** *Fix $\varepsilon \in (0,1)$. If $\mathcal{A}$ is an $\varepsilon$-DP mechanism that answers connected component queries on graphs with $n$ vertices in the continual release model within additive error $\zeta$ with probability at least 0.99 for fully dynamic streams of length $T$, then $\zeta = \Omega\left(\min\left(\sqrt{\frac{n}{\varepsilon}}, \frac{T^{1/4}}{\varepsilon^{3/4}}, n, T\right)\right)$. Moreover, we may assume the graph is bipartite, has maximum degree 2, and arboricity 2.*

The proofs of Theorems 19 and 20 are presented in the full version [34].

## 8.1 Further Graph Statistics

The underlying idea for the maximum matching and connected components lower bounds (Theorem 19, Theorem 20) is that we can encode the bits of a secret database $y \in \{0,1\}^n$ within the structure of a sparse graph. Privately answering an inner product query on this database then reduces to answering a "bitwise OR" query by the inclusion-exclusion principle.

It is not hard to see that this technique extends to $k$-edge-connected component queries, $k$-vertex-connected component queries, and triangle counting queries for sparse graphs.

# 9 Conclusion & Future Work

In this paper, we initiated the study of low-space continual release algorithms for general graph problems. Using techniques from the non-private graph sparsification literature, we provided continual release algorithms for a variety of general graph problems that for the first time, achieve nearly the same space and approximation guarantees of their non-private streaming counterparts. The improved space bounds are especially relevant for enabling computations on massive datasets, which are the core motivation of the field of online streaming algorithms.

For our upper bounds, we focused on the insertion-only setting of continual release. As the area of fully-dynamic algorithms in the continual release model is largely unexplored, we believe that an interesting future research direction is closing the gap in our theoretical understanding of this model. As observed in prior work, and hinted by our hardness results, the fully dynamic setting is significantly harder in continual release, with even basic graph problems requiring $\Omega(\text{poly}(n))$ additive error for dynamic streams while admitting $\tilde{O}(\text{poly}(\log(n))/\varepsilon)$ additive error in the insertion-only case. In this context, it would be especially interesting to deepen our understanding of the interplay between the dynamicity of the continual release setting (insertion-only vs fully-dynamic) and the *space* lower bounds (as opposed to error lower bounds) imposed by privacy. This is an area that has only recently received attention [24] and is an intriguing future direction to explore.

## References

1 Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Proceedings of the 38th International Conference on Automata, Languages and Programming – Volume Part II*, volume 6756 of *Lecture Notes in Computer Science*, pages 526–538, Berlin, Heidelberg, 2011. Springer. doi:10.1007/978-3-642-22012-8_42.

2 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 20–29. ACM, 1996. doi:10.1145/237814.237823.

3 Raman Arora and Jalaj Upadhyay. On differentially private graph sparsification and applications. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13378–13389, 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/e44e875c12109e4fa3716c05008048b2-Abstract.html.

4 Sepehr Assadi. A two-pass (conditional) lower bound for semi-streaming maximum matching. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 708–742. SIAM, 2022. doi:10.1137/1.9781611977073.32.

5 Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1616–1635. SIAM, 2019. doi:10.1137/1.9781611975482.98.

6 Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 627–669. SIAM, SIAM, 2022. doi:10.1137/1.9781611977073.29.

7 Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan. An auction algorithm for bipartite matching in streaming and massively parallel computation models. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 165–171. SIAM, 2021. doi:10.1137/1.9781611976496.18.

8 Petra Berenbrink, Bruce Krayenhoff, and Frederik Mallmann-Trenn. Estimating the number of connected components in sublinear time. *Inf. Process. Lett.*, 114(11):639–642, 2014. doi:10.1016/j.ipl.2014.05.008.

9 Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 173–182. ACM, 2015. doi:10.1145/2746539.2746592.

10 Jeremiah Blocki, Elena Grigorescu, and Tamalika Mukherjee. Privately estimating graph parameters in sublinear time. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 26:1–26:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.26.

11 Jeremiah Blocki, Elena Grigorescu, Tamalika Mukherjee, and Samson Zhou. How to make your approximation algorithm private: A black-box differentially-private transformation for tunable approximation algorithms of functions with low sensitivity. In Nicole Megow and Adam D. Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, September 11-13, 2023, Atlanta, Georgia, USA*, volume 275 of *LIPIcs*, pages 59:1–59:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.59.

12 Mark Bun, Marek Elias, and Janardhan Kulkarni. Differentially private correlation clustering. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1136–1146. PMLR, PMLR, 2021. URL: http://proceedings.mlr.press/v139/bun21a.html.

13 T.-H. Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In Simone Fischer-Hübner and Matthew K. Wright, editors, *Privacy Enhancing Technologies – 12th International Symposium, PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings*, volume 7384 of *Lecture Notes in Computer Science*, pages 140–159. Springer, 2012. doi:10.1007/978-3-642-31680-7_8.

14 T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, November 2011. doi:10.1145/2043621.2043626.

15 Hongjie Chen, Vincent Cohen-Addad, Tommaso d'Orsi, Alessandro Epasto, Jacob Imola, David Steurer, and Stefan Tiegel. Private estimation algorithms for stochastic block models and mixture models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10–16, 2023*, volume 36, pages 68134–68183, 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/d702d78b2468d2bc80b22a2fc3e59faf-Abstract-Conference.html.

16 Xiuge Chen, Rajesh Chitnis, Patrick Eades, and Anthony Wirth. Sublinear-space streaming algorithms for estimating graph parameters on sparse graphs. In Pat Morin and Subhash Suri, editors, *Algorithms and Data Structures – 18th International Symposium, WADS 2023, Montreal, QC, Canada, July 31 – August 2, 2023, Proceedings*, volume 14079 of *Lecture Notes in Computer Science*, pages 247–261. Springer, Springer, 2023. doi:10.1007/978-3-031-38906-1_17.

17 Graham Cormode, Jacques Dark, and Christian Konrad. Approximating the caro-wei bound for independent sets in graph streams. In Jon Lee, Giovanni Rinaldi, and Ali Ridha Mahjoub, editors, *Combinatorial Optimization – 5th International Symposium, ISCO 2018, Marrakesh, Morocco, April 11-13, 2018, Revised Selected Papers*, volume 10856 of *Lecture Notes in Computer Science*, pages 101–114. Springer, Springer, 2018. doi:10.1007/978-3-319-96151-4_9.

18 Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 29:1–29:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.29.

19 Anindya De. Lower bounds in differential privacy. In Ronald Cramer, editor, *Theory of Cryptography – 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 321–338. Springer, 2012. doi:10.1007/978-3-642-28914-9_18.

20 Laxman Dhulipala, George Z. Li, and Quanquan C. Liu. Near-optimal differentially private k-core decomposition. *CoRR*, abs/2312.07706, 2023. doi:10.48550/arXiv.2312.07706.

21 Laxman Dhulipala, Quanquan C. Liu, Sofya Raskhodnikova, Jessica Shi, Julian Shun, and Shangdi Yu. Differential privacy from locally adjustable graph algorithms: k-core decomposition, low out-degree ordering, and densest subgraphs. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 – November 3, 2022*, pages 754–765. IEEE, 2022. doi:10.1109/FOCS54457.2022.00077.

22 Michael Dinitz, Satyen Kale, Silvio Lattanzi, and Sergei Vassilvitskii. Almost tight bounds for differentially private densest subgraph. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 2908–2950. SIAM, 2025. doi:10.1137/1.9781611978322.94.

23 Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In Frank Neven, Catriel Beeri, and Tova Milo, editors, *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 202–210. ACM, 2003. doi:10.1145/773153.773173.

24 Itai Dinur, Uri Stemmer, David P Woodruff, and Samson Zhou. On differential privacy and adaptive data analysis with bounded space. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023 – 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 35–65. Springer, Springer, 2023. doi:10.1007/978-3-031-30620-4_2.

25 Max Dupré la Tour, Monika Henzinger, and David Saulpic. Differential privacy for clustering under continual observation. *arXiv e-prints*, pages arXiv–2307, 2023. doi:10.48550/arXiv.2307.03430.

26 Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 371–380. ACM, 2009. doi:10.1145/1536414.1536466.

27 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284, Berlin, Heidelberg, 2006. Springer. doi:10.1007/11681878_14.

28 Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of LP decoding. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 85–94. ACM, 2007. doi:10.1145/1250790.1250804.

29 Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 715–724. ACM, 2010. doi:10.1145/1806689.1806787.

30 Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 381–390. ACM, 2009. doi:10.1145/1536414.1536467.

31 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014. doi:10.1561/0400000042.

32 Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 51–60. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.12.

33 Talya Eden, Quanquan C. Liu, Sofya Raskhodnikova, and Adam D. Smith. Triangle counting with local edge differential privacy. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 52:1–52:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.52.

34 Alessandro Epasto, Quanquan C Liu, Tamalika Mukherjee, and Felix Zhou. Sublinear space graph algorithms in the continual release model. *arXiv preprint arXiv:2407.17619*, 2024. doi:10.48550/arXiv.2407.17619.

**35** Alessandro Epasto, Jieming Mao, Andres Muñoz Medina, Vahab Mirrokni, Sergei Vassilvitskii, and Peilin Zhong. Differentially private continual releases of streaming frequency moment estimations. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 48:1–48:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.48.

**36** Hossein Esfandiari, MohammadTaghi Hajiaghayi, and David P. Woodruff. Brief announcement: Applications of uniform sampling: Densest subgraph and beyond. In Christian Scheideler and Seth Gilbert, editors, *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific Grove, CA, USA, July 11-13, 2016*, pages 397–399. ACM, 2016. doi:10.1145/2935764.2935813.

**37** Hossein Esfandiari, Silvio Lattanzi, and Vahab Mirrokni. Parallel and streaming algorithms for $k$-core decomposition. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1396–1405. PMLR, 2018. URL: http://proceedings.mlr.press/v80/esfandiari18a.html.

**38** Alireza Farhadi, MohammadTaghi Hajiaghayi, and Elaine Shi. Differentially private densest subgraph. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, volume 151 of *Proceedings of Machine Learning Research*, pages 11581–11597. PMLR, 2022. URL: https://proceedings.mlr.press/v151/farhadi22a.html.

**39** Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. doi:10.1016/j.tcs.2005.09.013.

**40** Hendrik Fichtenberger, Monika Henzinger, and Wolfgang Ost. Differentially private algorithms for graphs under continual observation. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 42:1–42:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.42.

**41** Hendrik Fichtenberger, Monika Henzinger, and Jalaj Upadhyay. Constant matters: Fine-grained error bound on differentially private continual observation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10072–10092. PMLR, PMLR, 2023. URL: https://proceedings.mlr.press/v202/fichtenberger23a.html.

**42** Manuela Fischer, Slobodan Mitrović, and Jara Uitto. Deterministic $(1 + \varepsilon)$-approximate maximum matching with poly$(1/\varepsilon)$ passes in the semi-streaming model and beyond. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 – 24, 2022*, STOC 2022, pages 248–260, New York, NY, USA, 2022. ACM. doi:10.1145/3519935.3520039.

**43** Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete mathematics & theoretical computer science*, 2007. doi:10.46298/dmtcs.3545.

**44** Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985. doi:10.1016/0022-0000(85)90041-8.

**45** Prantar Ghosh and Manuel Stoeckl. Low-memory algorithms for online edge coloring. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 71:1–71:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.71.

46 Bjarni V Halldórsson, Magnús M Halldórsson, Elena Losievskaja, and Mario Szegedy. Streaming algorithms for independent sets in sparse hypergraphs. *Algorithmica*, 76(2):490–501, 2016. doi:10.1007/s00453-015-0051-5.

47 Moritz Hardt and Guy N Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 61–70. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.85.

48 Monika Henzinger, A. R. Sricharan, and Leqi Zhu. Tighter bounds for local differentially private core decomposition and densest subgraph. *CoRR*, abs/2402.18020, 2024. doi:10.48550/arXiv.2402.18020.

49 Monika Henzinger, AR Sricharan, and Teresa Anna Steiner. Differentially private histogram, predecessor, and set cardinality under continual observation. *arXiv preprint arXiv:2306.10428*, 2023. doi:10.48550/arXiv.2306.10428.

50 Monika Henzinger, Jalaj Upadhyay, and Sarvagya Upadhyay. A unifying framework for differentially private sums under continual observation. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 995–1018. SIAM, SIAM, 2024. doi:10.1137/1.9781611977912.38.

51 Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. In James M. Abello and Jeffrey Scott Vitter, editors, *External Memory Algorithms, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, May 20-22, 1998*, volume 50 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 107–118. DIMACS/AMS, 1998. doi:10.1090/dimacs/050/05.

52 Zengfeng Huang and Pan Peng. Dynamic graph stream algorithms in o(n) space. *Algorithmica*, 81(5):1965–1987, 2019. doi:10.1007/S00453-018-0520-8.

53 Jacob Imola, Alessandro Epasto, Mohammad Mahdian, Vincent Cohen-Addad, and Vahab Mirrokni. Differentially private hierarchical clustering with provable approximation guarantees. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 14353–14375. PMLR, PMLR, 2023. URL: https://proceedings.mlr.press/v202/imola23a.html.

54 Palak Jain, Iden Kalemaj, Sofya Raskhodnikova, Satchit Sivakumar, and Adam D. Smith. Counting distinct elements in the turnstile model with differential privacy under continual observation. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 – 16, 2023*, 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/0ef1afa0daa888d695dcd5e9513bafa3-Abstract-Conference.html.

55 Palak Jain, Sofya Raskhodnikova, Satchit Sivakumar, and Adam D. Smith. The price of differential privacy under continual observation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 14654–14678. PMLR, 2023. URL: https://proceedings.mlr.press/v202/jain23b.html.

56 Palak Jain, Adam D. Smith, and Connor Wagaman. Time-aware projections: Truly node-private graph statistics under continual observation. In *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*, pages 127–145. IEEE, 2024. doi:10.1109/SP54263.2024.00196.

57 Iden Kalemaj, Sofya Raskhodnikova, Adam Smith, and Charalampos E Tsourakakis. Node-differentially private estimation of the number of connected components. In Floris Geerts, Hung Q. Ngo, and Stavros Sintos, editors, *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2023, Seattle, WA, USA, June 18-23, 2023*, pages 183–194. ACM, 2023. doi:10.1145/3584372.3588671.

**58** Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In Amit Sahai, editor, *Theory of Cryptography – 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, volume 7785 of *Lecture Notes in Computer Science*, pages 457–476. Springer, Springer, 2013. doi:10.1007/978-3-642-36594-2_26.

**59** Valerie King, Alex Thomo, and Quinton Yong. Computing (1+epsilon)-approximate degeneracy in sublinear time. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 2160–2168. ijcai.org, 2023. doi:10.24963/IJCAI.2023/240.

**60** Jingcheng Liu, Jalaj Upadhyay, and Zongrui Zou. Optimal bounds on private graph approximation. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 1019–1049. SIAM, 2024. doi:10.1137/1.9781611977912.39.

**61** Quanquan C. Liu, Jessica Shi, Shangdi Yu, Laxman Dhulipala, and Julian Shun. Parallel batch-dynamic algorithms for $k$-core decomposition and related graph problems. In Kunal Agrawal and I-Ting Angelina Lee, editors, *SPAA '22: 34th ACM Symposium on Parallelism in Algorithms and Architectures, Philadelphia, PA, USA, July 11–14, 2022*, pages 191–204. ACM, 2022. doi:10.1145/3490148.3538569.

**62** Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *Proc. VLDB Endow.*, 10(6):637–648, 2017. doi:10.14778/3055330.3055331.

**63** Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014. doi:10.1145/2627692.2627694.

**64** Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T Vu. Densest subgraph in dynamic graph streams. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Mathematical Foundations of Computer Science 2015 – 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II*, volume 9235 of *Lecture Notes in Computer Science*, pages 472–482. Springer, Springer, 2015. doi:10.1007/978-3-662-48054-0_39.

**65** Andrew McGregor and Sofya Vorotnikova. A simple, space-efficient, streaming algorithm for matchings in low arboricity graphs. In Raimund Seidel, editor, *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, volume 61 of *OASIcs*, pages 14:1–14:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/OASIcs.SOSA.2018.14.

**66** Darakhshan Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 37–48. ACM, 2011. doi:10.1145/1989284.1989290.

**67** Robert Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, 1978. doi:10.1145/359619.359627.

**68** Tamara T Mueller, Dmitrii Usynin, Johannes C Paetzold, Daniel Rueckert, and Georgios Kaissis. Sok: Differential privacy on graph-structured data. *arXiv preprint arXiv:2203.09205*, 2022. doi:10.48550/arXiv.2203.09205.

**69** Shanmugavelayutham Muthukrishnan et al. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, 2005. doi:10.1561/0400000002.

**70** Dung Nguyen and Anil Vullikanti. Differentially private densest subgraph detection. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8140–8151. PMLR, 2021. URL: http://proceedings.mlr.press/v139/nguyen21i.html.

71 Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 75–84. ACM, 2007. doi:10.1145/1250790.1250803.

72 Sofya Raskhodnikova and Adam Smith. Differentially private analysis of graphs. In *Encyclopedia of Algorithms*, pages 543–547. Springer, 2016. doi:10.1007/978-1-4939-2864-4_549.

73 Sofya Raskhodnikova and Adam D. Smith. Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 495–504. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.60.

74 Sofya Raskhodnikova and Teresa Anna Steiner. Fully dynamic graph algorithms with edge differential privacy. *arXiv preprint arXiv:2409.17623*, 2024. doi:10.48550/arXiv.2409.17623.

75 Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 765–774. ACM, 2010. doi:10.1145/1806689.1806794.

76 Ahmet Erdem Sariyüce, Bugra Gedik, Gabriela Jacques-Silva, Kun-Lung Wu, and Ümit V. Çatalyürek. Streaming algorithms for k-core decomposition. *Proc. VLDB Endow.*, 6(6):433–444, April 2013. doi:10.14778/2536336.2536344.

77 Shuang Song, Susan Little, Sanjay Mehta, Staal A. Vinterbo, and Kamalika Chaudhuri. Differentially private continual release of graph statistics. *CoRR*, abs/1809.02575, 2018. doi:10.48550/arXiv.1809.02575.

78 Bintao Sun, T.-H. Hubert Chan, and Mauro Sozio. Fully dynamic approximate $k$-core decomposition in hypergraphs. *ACM Trans. Knowl. Discov. Data*, 14(4):39:1–39:21, May 2020. doi:10.1145/3385416.

79 Jalaj Upadhyay. Random projections, graph sparsification, and differential privacy. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013 – 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 276–295. Springer, 2013. doi:10.1007/978-3-642-42033-7_15.

80 Jalaj Upadhyay. Sublinear space private algorithms under the sliding window model. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6363–6372. PMLR, PMLR, 2019. URL: http://proceedings.mlr.press/v97/upadhyay19a.html.

81 Jalaj Upadhyay, Sarvagya Upadhyay, and Raman Arora. Differentially private analysis on graph streams. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 1171–1179. PMLR, PMLR, 2021. URL: http://proceedings.mlr.press/v130/upadhyay21a.html.