How to Make Your Approximation Algorithm Private:

A Black-Box Differentially-Private Transformation
for Tunable Approximation Algorithms of Functions with Low Sensitivity

Jeremiah Blocki[*]
Purdue University

Elena Grigorescu[†]
Purdue University

Tamalika Mukherjee[‡]
Columbia University

Samson Zhou[§]
UC Berkeley and Rice University

September 12, 2023

Abstract

We develop a framework for efficiently transforming certain approximation algorithms into differentially-private variants, in a black-box manner. Specifically, our results focus on algorithms A that output an approximation to a function f of the form $(1-\alpha)f(x)-\kappa \le A(x) \le (1+\alpha)f(x)+\kappa$, where $\kappa \in \mathbb{R}_{\ge 0}$ denotes additive error and $\alpha \in [0,1)$ denotes multiplicative error can be"tuned" to small-enough values while incurring only a polynomial blowup in the running time/space. We show that such algorithms can be made differentially private without sacrificing accuracy, as long as the function f has small "global sensitivity". We achieve these results by applying the "smooth sensitivity" framework developed by Nissim, Raskhodnikova, and Smith (STOC 2007).

Our framework naturally applies to transform non-private FPRAS and FPTAS algorithms into ε-differentially private approximation algorithms where the former case requires an additional postprocessing step. We apply our framework in the context of sublinear-time and sublinear-space algorithms, while preserving the nature of the algorithm in meaningful ranges of the parameters. Our results include the first (to the best of our knowledge) ε-edge differentially-private sublinear-time algorithm for estimating the number of triangles, the number of connected components, and the weight of a minimum spanning tree of a graph whose accuracy holds with high probability. In the area of streaming algorithms, our results include ε-DP algorithms for estimating L_p-norms, distinct elements, and weighted minimum spanning tree for both insertion-only and turnstile streams. Our transformation also provides a private version of the smooth histogram framework, which is commonly used for converting streaming algorithms into sliding window variants, and achieves a multiplicative approximation to many problems, such as estimating L_p-norms, distinct elements, and the length of the longest increasing subsequence.

[*]E-mail: jblocki@purdue.edu. Supported in part by NSF CCF-1910659, NSF CNS-1931443. and NSF CAREER award CNS-2047272

[†]E-mail: elena-g@purdue.edu. Supported in part by NSF CCF-1910659, NSF CCF-1910411, and NSF CCF-2228814.

[‡]E-mail: tm3391@columbia.edu. Most of the work was done as a Ph.D. student at Purdue University. Supported in part by the Bilsland Dissertation Fellowship, NSF CCF-1910659 and and NSF CCF-2228814.

[§]E-mail: samsonzhou@gmail.com. Work done in part while at Carnegie Mellon University and supported in part by a Simons Investigator Award and by NSF CCF-1815840.

1 Introduction

Approximation algorithms are often used to efficiently approximate a function $f : \mathcal{D} \to \mathbb{R}^+$ in settings where resource constraints prevent us from computing the function exactly. For example, problems such as Knapsack are NP-Hard and, unless $\mathsf{P} = \mathsf{NP}$, do not admit a polynomial time solution. However, the Knapsack problem admits a fully polynomial time approximation scheme (FPTAS) i.e., for any $\alpha > 0$ there is a deterministic algorithm, running in time $\mathrm{poly}(n, 1/\alpha)$, which outputs a solution that is guaranteed to be be nearly as good (up to multiplicative factor $1 \pm \alpha$) as the optimal solution. As a second example, even if the problem is computationally tractable it may still be the case that the input dataset $D \in \mathcal{D}$ is extremely large, making it infeasible to load the entire dataset into RAM, or impractical to execute a linear time algorithm. To remedy such shortcomings, models such as sublinear-space and sublinear-time algorithms have been proposed. For example, one may want to estimate frequencies of elements that appear in a stream of n elements up to a multiplicative $1 \pm \alpha$ factor, while using only $\mathrm{poly}\left(\log n, \frac{1}{\alpha}\right)$ memory cells. Or, one may want to estimate the number of connected components of a dense graph on n vertices up to (relative) additive error κ by only inspecting $\mathrm{poly}(\log n, \frac{1}{\kappa})$ many edges.

In addition to time and space efficiency, user privacy is another important consideration in contexts where the input to our function f is sensitive user data. Differential privacy (DP) [Dwo06, DMNS06] is a rigorous mathematical concept that gives provable guarantees on what it means for an algorithm to preserve the privacy of individual information in the input dataset. Informally, a randomized function computed on a dataset D is *differentially private* if the distribution of the function's output does not change significantly with the presence or absence of an individual data point. Thus, a natural goal is to develop efficient differentially private algorithms to approximate functions/queries of interest.

One general way to preserve differential privacy is to add noise scaled to the global sensitivity Δ_f of our function f, i.e., the maximum amount $|f(D) - f(D')|$ that the answer could change by modifying a single record in our dataset D to obtain a new dataset D'. This general approach yields efficient and accurate approximations for f as long as we have an efficient algorithm to compute f *exactly* and the global sensitivity of f is sufficiently small. However, in some resource-constrained settings, we may need to use an approximation algorithm $\mathcal{A}_f$ instead of evaluating f exactly. Unfortunately, the mechanism that computes $\mathcal{A}_f(D)$ and then adds noise scaled to the global sensitivity Δ_f of our function f is not necessarily differentially private. In particular, even if we are guaranteed that $|\mathcal{A}_f(D) - f(D)| \le \alpha f(D)$ we might still have $|\mathcal{A}_f(D) - \mathcal{A}_f(D')| \ge 2\alpha f(D) \gg \Delta_f$ for neighboring datasets D and D', e.g., suppose $\mathcal{A}_f(D) = (1+\alpha)f(D)$ and $\mathcal{A}_f(D') = (1-\alpha)f(D')$. Thus, the global sensitivity of $\mathcal{A}_f$ can be quite large and adding noise proportional to $\Delta_{\mathcal{A}_f}$ would prevent us from providing meaningful accuracy guarantees. This raises a natural question: Suppose that our function f admits an accurate (but not necessarily resource-efficient) differentially private approximation algorithm and that f also admits an efficient (but not necessarily private) approximation algorithm. Is it necessarily the case that there is also an equally efficient differentially private approximation algorithm?

Unfortunately, a result of [HT10, BUV18] suggests that the answer to the previous question may be no. Suppose our dataset D consists of n users $x_1, \ldots, x_n$ with n binary attributes i.e., $x_i \in \{0,1\}^n$. Consider the function $f(D)$ that computes all of the one-way marginals i.e., $f(D) = \langle \frac{1}{n} \sum_{i=1}^n x_i[j] \rangle_{j=1}^n \in \mathbb{R}^n$. In particular, there is a non-private sublinear time algorithm that samples $\mathcal{O}(\log n)$ users and (with high probability) outputs a good approximation to *all* n one-way marginals. However, if we require that our algorithm satisfy pure, i.e, ε-differential privacy (resp. approximate, i.e., (ε, δ)-differential privacy) then we need to look at *at least* $\Omega(n/\varepsilon)$ (resp. $\Omega(\sqrt{n \log(1/\delta)})$) samples [HT10, BUV18]. In light of this, we pose the following general questions:

What are sufficient conditions for an approximation algorithm to be made differentially private? Can an approximation algorithm be made differentially private in an efficient black-box manner?

Over the years, many differentially private approximation algorithms have been developed for problems in optimization, machine learning, and distribution testing (see for e.g., [GLM+10, ASZ18, GKMN21, GXP+20]), in a somewhat ad-hoc manner. Often, these results give a differentially private algorithm for that specific problem and do not easily generalize to give differentially private algorithms for a large class of problems. A general framework for developing differentially private approximation algorithms for a large class of problems is desirable as this would not only make DP approximation algorithms more easily accessible to non-DP experts, but more importantly, it would shed light on what kinds of algorithms are more amenable to differential privacy. Furthermore, a framework that uses the underlying approximation algorithm as a *black-box* is desirable as this avoids the need to (re)design, (re)analyze, and (re)implement the new differentially private versions of these approximation algorithms. We emphasize that this type of framework has been well-studied for *computing* functions privately by calibrating noise proportional to their global or smooth sensitivity [DMNS16, NRS07] (see Section 1.3 for more discussion).

Our work makes partial progress towards answering these general questions. In particular, we give an efficient black-box framework for converting a non-private approximation algorithm $\mathcal{A}_f$ with tunable accuracy parameters into a differentially private approximation algorithm $\mathcal{A}'_f$ with reasonable accuracy guarantees as long as the global sensitivity Δ_f of the function f being approximated is sufficiently low. For the case when $\mathcal{A}_f$ is deterministic, we achieve a pure ε-differentially private approximation algorithm via a direct transformation, and when $\mathcal{A}_f$ is randomized, i.e., has a small failure probability, we achieve ε-differential privacy by first applying a transformation that gives a (ε, δ)-differentially private algorithm and then apply a postprocessing step to achieve ε-differentially privacy. For example, suppose that for any $\alpha > 0$ our algorithm $\mathcal{A}_f$, taking α and our dataset D as input, provides the guarantee that $|\mathcal{A}_f(D) - f(D)| \leq \alpha f(D)$ e.g., any FPTAS algorithm would satisfy our tunable accuracy requirement. In such a case, for any $\alpha > 0$ we can transform our non-private algorithm $\mathcal{A}_f$ into a differentially private version with multiplicative error α and small additive error term which (necessarily) comes from the noise that we added. Intuitively, we exploit the fact that we can run $\mathcal{A}_f$ with an even smaller accuracy parameter $\rho \ll \alpha$ which can be tuned to ensure that the smooth sensitivity of our algorithm is sufficiently small. Our same general framework still applies if we allow that the approximation algorithm $\mathcal{A}_f$ has a small additive error term i.e., $|\mathcal{A}_f(D) - f(D)| \leq \alpha f(D) + \kappa$. If $\mathcal{A}_f(D)$ is only guaranteed to output a good approximation (i.e., $|\mathcal{A}_f(D) - f(D)| \leq \alpha f(D) + \kappa$) with probability $1 - \delta/2$ (e.g., an FPRAS algorithm would satisfy this requirement with additive error $\kappa = 0$) then our framework achieves ε-differential privacy by first obtaining an approximate (ε, δ)-differential privacy algorithm and then a postprocessing step. In cases where the approximation algorithm is not tunably accurate our black-box framework does not necessarily apply[1]. For example, the best known approximation algorithms for vertex cover achieve the guarantee $f(G) \leq \mathcal{A}_f(G) \leq 2f(G)$ i.e., because there is no way to control the smooth sensitivity of our approximation algorithm.

[1]One could still apply our black-box transformation. However, the accuracy guarantees would be degraded and we would only achieve (ε, δ)-differential privacy for sufficiently large values of $\varepsilon, \delta > 0$ which depend on the approximation error parameter α.

1.1 Our Contributions

We introduce a generic black-box framework for converting certain approximation algorithms for a function $f : \mathcal{D} \to \mathbb{R}^+$ into a differentially private approximation algorithm using smooth sensitivity [NRS07]. We first introduce a definition for *tunable* approximation algorithms used throughout our paper, and then present our main results for the DP framework, and then give new differentially private algorithms for a variety of approximation algorithms obtained via this unifying framework.

Definition 1.1 ((α, κ, δ)-approximation). *An algorithm $\mathcal{A}_f$ is a (α, κ, δ)-approximation for f if for every $D \in \mathcal{D}$ with probability at least $1-\delta$, we have that $(1-\alpha)f(D)-\kappa \le \mathcal{A}_f(D) \le (1+\alpha)f(D)+\kappa$.*

We may abuse notation and omit the failure probability δ parameter in the above definition, if it is clear from the context. Some algorithms $\mathcal{A}_f$ may take the approximation parameters $\alpha, \kappa, \delta \ge 0$ as input[2].

Definition 1.2 (tunable approximation). *$\mathcal{A}_f(D, \alpha, \kappa, \delta)$ provides a tunable approximation of f if for every $\alpha, \kappa, \delta \ge 0$ the algorithm $\mathcal{A}_f(\cdot, \alpha, \kappa, \delta)$ obtained by hardcoding α, κ and δ is a (α, κ, δ)-approximation for f.*
When the parameters α, κ, δ are clear from the context, we may abuse notation and just write $\mathcal{A}_f(D)$. For a tunable approximation algorithm we will use $R(n, \alpha, \kappa, \delta)$ to denote the amount of a particular resource used by the algorithm. The resources we consider in this work include time, space and query complexity of the algorithm (depending on the model) which we denote by $T(\cdot,\cdot,\cdot,\cdot)$, $S(\cdot,\cdot,\cdot,\cdot)$, and $Q(\cdot,\cdot,\cdot,\cdot)$ respectively.

As a concrete example any FPTAS algorithm $\mathcal{A}_f$ for f would be a tunable approximation for f with running time $T(n, \alpha, \kappa, \delta) = \mathrm{poly}(n, 1/\alpha)$ for any $\alpha > 0$ and any $\kappa, \delta \ge 0$ — an FPTAS has no additive error ($\kappa = 0$) and zero failure probability ($\delta = 0$). Similarly a FPRAS algorithm would be a tunable approximation with running time $T(n, \alpha, \kappa, \delta) = \mathrm{poly}(n, \alpha, \log(1/\delta))$ for any $\alpha, \delta > 0$ and any $\kappa \ge 0$ — an FPRAS also has no additive error ($\kappa = 0$).

General Framework for Approximation Algorithms. Our main result gives a framework for converting any existing non-DP algorithm $\mathcal{A}_f$ that provides an (α, κ, δ)-approximation of f into an ε-DP algorithm $\mathcal{A}''_f$ in the following manner: (1) Apply Algorithm 1 to obtain an (ε, δ)-DP algorithm $\mathcal{A}'_f$ that achieves an $(\alpha', \kappa', \delta')$-approximation (see Theorem 1.3), (2) Apply a postprocessing step on the output of $\mathcal{A}'_f$ outlined in Theorem 1.5 to achieve an ε-DP algorithm $\mathcal{A}''_f$ with the same accuracy guarantees as $\mathcal{A}'_f$ barring an additive error of $o(1)$. We emphasize that $\mathcal{A}_f$ is a tunable approximation, in other words, $\mathcal{A}_f$ takes the parameters (α, κ, δ) as input.

Theorem 1.3. *((ε, δ)-privacy) Suppose that $\mathcal{A}_f$ is a tunable approximation of $f : \mathcal{D} \to \mathbb{R}^+$. Then for all $\varepsilon > 0$, $\delta = \delta(n) > 0$[3], $\alpha \ge 0$ and $\kappa \ge 0$, there is an algorithm $\mathcal{A}'_f$ such that*

(1) (Privacy) $\mathcal{A}'_f$ is (ε, δ')-differentially private where $\delta' = \delta(1 + \exp(\varepsilon/2))$.

(2) (Accuracy) For all $D \in \mathcal{D}$, and $0 < \gamma$, with probability $1 - \delta - \exp(-\gamma)$,

$$(1-\alpha')f(D) - \kappa' - \frac{2\Delta_f}{\varepsilon} \cdot \gamma \le \mathcal{A}'_f(D) \le (1+\alpha')f(D) + \kappa' + \frac{2\Delta_f}{\varepsilon} \cdot \gamma$$

[2]We allow that $\alpha = \kappa = \delta = 0$ in which case $\mathcal{A}_f$ can simply compute f exactly — whether or not this computation is efficient.

[3]typically we set $\delta = \mathrm{negl}(n)$ or $\delta = n^{-c}$ for some constant $c > 0$. In particular $\delta(n)$ may approach zero as $n \to \infty$.

where $\alpha' = \frac{\alpha(\varepsilon+16\gamma)}{12\log(4/\delta)}$, *and* $\kappa' = \kappa\left(\frac{2\gamma\alpha}{3\log(4/\delta)} + \frac{8\gamma}{\varepsilon} + 1\right)$, *and* $\Delta_f := \max_{D,D'\in\mathcal{D},D\sim D'} \|f(D) - f(D')\|_1$.

(3) (Resource) $\mathcal{A}'_f$ *uses* $R\left(n, \frac{\varepsilon\alpha}{\log(4/\delta)}, \kappa, \delta\right)$ *resource, where* $R(\cdot,\cdot,\cdot,\cdot)$ *is the resource used by* $\mathcal{A}_f$*.*

We illustrate the utility of Theorem 1.3 with specific parameters — if we have a non-private algorithm $\mathcal{A}_f$ that guarantees an $(\alpha, 0, \delta)$-approximation, then for constant ε, $\delta = \frac{1}{n^c}$ and $\gamma = c\log(n)$, we see that the DP algorithm $\mathcal{A}_f$ achieves an $\left(\alpha(1+o(1)), \mathcal{O}\left(\frac{\Delta_f \log(n)}{\varepsilon}\right), \frac{1}{n^c}\right)$-approximation. We typically use these parameters for δ, γ in our applications for streaming and sublinear-time algorithms.

Our reduction in Theorem 1.3 is quite simple – we describe the associated Algorithm 1 below.

Algorithm 1 (ε, δ)-differentially private framework $\mathcal{A}'_f$ for tunable approximation algorithm $\mathcal{A}_f$

```
Input: Input set $D$, accuracy parameters $\alpha \in (0,1)$ and $\kappa$, DP parameter $\varepsilon$, DP failure probability
    $\delta \in (0,1)$, approx. algorithm $\mathcal{A}_f$.
1: Let $x_A := \mathcal{A}_f(D, \rho, \tau, \delta/2)$, where $\rho := \left(\frac{\varepsilon\alpha}{12\log(4/\delta)}\right)$, and $\tau := \kappa$.
2: return $x_A + X$ where $X \sim \mathsf{Lap}\left(\frac{2(4\rho x_A + 4\tau + \Delta_f)}{\varepsilon}\right)$
```

Note that in Algorithm 1, we leave our additive parameter κ as is when running $\mathcal{A}_f$, but we still choose to define $\tau := \kappa$. This is because depending on the problem, and the accuracy/efficiency guarantees desired, we can set τ to be a tuned version of κ (for e.g., we set $\tau := \kappa/\log(n)$ for the problem of estimating the number of connected components).

Remark 1.4. *We also note that, even if the failure probability* $\delta > 0$ *of* $\mathcal{A}_f$ *is non-negligible, that we can always boost the success probability by running* $\mathcal{A}_f(D)$ *multiple times and computing the median over all outputs. Even if the error rate* $0 < \delta < 1/2$ *is a constant we can always reduce the failure probability to a lower target* $0 < \delta' \ll \delta$ *while increasing the running time by a multiplicative factor* $O(\log(1/\delta'))$*. In particular, we can set* δ' *to be a negligible function of* n *such as* $\delta' = n^{-\log n}$ *whilst only incurring a* $\mathcal{O}(\log^2 n)$ *blowup in our running time.*

We stress that we can only apply Theorem 1.3 to existing non-DP algorithms $\mathcal{A}_f$ that give an approximation guarantee of the form $(1-\alpha)f(D) - \kappa \le \mathcal{A}_f \le (1+\alpha)f(D) + \kappa$. For example, we cannot apply Theorem 1.3 to obtain an (ε, δ)-DP algorithm for estimating the minimum vertex cover size in sublinear time. This is because the non-DP sublinear-time algorithm $\mathcal{A}_{vc}$ has an approximation guarantee of the form $2VC(G) - \kappa n \le \mathcal{A}_{vc} \le 2VC(G) + \kappa n$. On the other hand, we can use our DP framework to obtain an (ε, δ)-DP algorithm for obtaining a $(0, \kappa n, \delta)$-approximation of the maximum matching size in sublinear time (see Corollary 4.10). Intriguingly, both the minimum vertex cover size and the maximum matching size algorithms use the same underlying strategy of estimating a greedy maximal matching in a local fashion, but since they return different estimators based on the objective and we can only use our framework as a black-box, we cannot apply our framework to the former while we can still apply it to the latter.

Finally, by applying a post-processing step described below, we show how to obtain an ε-DP algorithm from the (ε, δ)-DP algorithm obtained in Theorem 1.3. Importantly, the accuracy guarantee of the resulting ε-DP algorithm only differs by a small additive factor of $1/(KM)$, where

$M = \max_D f(D)$ is the maximum possible output value, e.g., $M \leq n^3$ for triangle counting and $K > 0$. Moreover for negligible δ, the accuracy guarantee of the resulting pure DP algorithm still holds with high probability.

Theorem 1.5. *Let $M = \max_D f(D)$ and let parameter $K > 0$. If $\mathcal{A}'_f(D)$ is (ε, δ)-DP algorithm with accuracy guarantee $(1-\alpha)f(D) - \kappa \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D) + \kappa$ holding with probability $1-\eta$ then there exists an algorithm $\mathcal{A}''_f(D)$ which is ε-DP with accuracy guarantee $(1-\alpha)f(D) - \kappa - \frac{1}{KM} \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D) + \kappa + \frac{1}{KM}$ with probability at least $1-\eta-p$ where $p = \frac{\delta K(M+1)}{e^{\varepsilon}-1+\delta K(M+1)}$.*

Our second result is an analogous framework for converting any existing deterministic non-DP approximation algorithm $\mathcal{A}_f$ that provides an $(\alpha, \kappa, 0)$-approximation of f into an ε-DP algorithm $\mathcal{A}'_f$.

Theorem 1.6. *(ε-privacy) Suppose that $\mathcal{A}_f$ is a deterministic tunable approximation of $f : \mathcal{D} \to \mathbb{R}^+$. Then for all $\varepsilon > 0$, $\alpha \geq 0$ and $\kappa \geq 0$, there is an algorithm $\mathcal{A}'_f$ such that*

(1) (Privacy) $\mathcal{A}'_f$ is ε-differentially private.

(2) (Accuracy) For all $D \in \mathcal{D}$, we have that with probability $\geq 9/10$,

$$(1-\alpha')f(D) - \kappa' - \frac{7\Delta_f}{\varepsilon} \leq \mathcal{A}'_f(D) \leq (1+\alpha')f(D) + \kappa' + \frac{7\Delta_f}{\varepsilon}$$

where $\alpha' := \alpha C_1(\varepsilon + C_2\gamma)$, $\kappa' := \kappa C_3(\alpha + \frac{C_4}{\varepsilon})$ for some constants $C_1, C_2, C_3, C_4 > 0$ and $\Delta_f := \max_{D, D' \in \mathcal{D}, D \sim D'} \|f(D) - f(D')\|_1$.

(3) (Resource) $\mathcal{A}'_f$ uses $R\left(n, \frac{\varepsilon\alpha}{36}, \kappa\right)$ resource, where $R(\cdot,\cdot,\cdot)$ is the resource used by $\mathcal{A}_f$.

DP Sublinear-time Results. We use Theorem 1.3 in conjunction with Theorem 1.5 in a black-box manner to obtain pure differentially-private sublinear time algorithms for several problems (see Table 1 for a summary).

In many models of sublinear-time computation the efficiency of the algorithm is measured in the number of queries made to the input, rather than the time complexity of the algorithm. It is often the case that the two are polynomially related, but there are instances in which the actual time complexity of the algorithm may be exponentially larger than the query complexity, in terms of the approximation factor. Nevertheless, in these instances too, the literature uses time and query complexity interchangeably. This is because the sublinear-time model assumes restricted or expensive access to the input, while further computation on local machines with the answers obtained from queries is considered to be cheap. We use query complexity for the sake of clarity.

We note that in the sublinear-time literature, the approximation parameters α, κ are usually considered to be a constant, but the analyses for most of these theorems hold for $\alpha = \alpha(n), \kappa = \kappa(n) \in (0, 1)$, where n is the input size.

Here we do not explicitly define the sublinear model (or the queries allowed) for each problem, see Section 4 for more details. For a graph G we denote the number of vertices as n, the number of edges as m, and the average degree of the graph as $\bar{d}$.

Typically, the accuracy guarantees of the non-DP results are presented with probability at least $2/3$ — in order to apply our framework, we apply the median trick (see Remark 1.4) to boost the probability of success to $1-\delta$. For simplicity of comparing our results, for any constant $c > 0$, we set $\delta := 1/n^c$ in the sequel.

Problem	Reference	Privacy	Mult. error	Add. error	Query Complexity
Number of Triangles	[ELRS17]	Non-Private	α	0	$\mathcal{O}\left((\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t})\,\mathrm{poly}(\log(n), \frac{1}{\alpha})\right)$
	This Work	ε-edge DP	α	$\mathcal{O}\left(\frac{n\log(n)}{\varepsilon}\right)$	$\mathcal{O}\left((\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t})\,\mathrm{poly}(\log(n), \frac{1}{\alpha\varepsilon})\right)$
Connected Components	[BKM14]	Non-Private	0	κn	$\mathcal{O}\left(\frac{1}{\kappa^2}\log\left(\frac{1}{\kappa}\right)\log(n)\right)$
	This Work	ε-edge DP	0	$\mathcal{O}(\kappa n) + \mathcal{O}\left(\frac{\log(n)}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{\log^3(n)}{\kappa^2}\log\left(\frac{\log(n)}{\kappa}\right)\right)$
Weighted MST	[CRT05]	Non-Private	α	0	$\mathcal{O}\left(\bar{d}w\alpha^{-2}\log\left(\frac{\bar{d}w}{\alpha}\right)\log(n)\right)$
	This Work	ε-edge DP	α	$\mathcal{O}\left(\frac{\log(n)}{\varepsilon}\right)$	$\mathcal{O}\left(\bar{d}w\frac{\log^2(n)}{\alpha^2\varepsilon^2}\log\left(\frac{\bar{d}w\log(n)}{\alpha\varepsilon}\right)\log(n)\right)$
Average Degree	[GR04]	Non-Private	α	0	$\mathcal{O}\left(\frac{n}{\sqrt{m}}\,\mathrm{poly}\left(\frac{\log(n)}{\alpha}\right)\log(n)\right)$
	[BGM22]	ε-edge DP	α	0	$\mathcal{O}\left(\sqrt{n}\,\mathrm{poly}\left(\frac{\log(n)}{\alpha}\right)\mathrm{poly}\left(\frac{1}{\varepsilon}\right)\log(n)\right)$ (analysis assumes $\bar{d} \geq 1$)
	This Work	ε-edge DP	α	0	$\mathcal{O}\left(\frac{n}{\sqrt{m}}\,\mathrm{poly}\left(\frac{\log^2(n)}{\alpha\varepsilon}\right)\log(n)\right)$ for $\bar{d} = \Omega(\frac{\log(n)}{n\varepsilon})$
Maximum Matching Size	[YYI12]	Non-Private	0	κn	$\mathcal{O}\left(d^{\mathcal{O}(1/\kappa^2)}\log(n)\right)$
	This Work	ε-node DP	0	$\mathcal{O}\left(\frac{\kappa n}{\varepsilon}\right)$	$\mathcal{O}\left(d^{\mathcal{O}(1/\kappa^2)}\log(n)\right)$
Distance to Bipartiteness	[GMRS22]	Non-Private	0	κn^2	$\mathcal{O}\left((1/\kappa^3)\log(n)\right)$
	This Work	ε-edge DP	0	$\mathcal{O}\left(\kappa n^2\right) + \mathcal{O}\left(\frac{\log(n)}{\varepsilon}\right)$	$\mathcal{O}\left((\log^4(n)/\kappa^3)\right)$

Table 1: Summary of Sublinear-time DP graph algorithms obtained via our black-box DP transformation. According to our notation multiplicative error α means a multiplicative factor of $(1 \pm \alpha)$. See Section 4 for details.

We give the first (to the best of our knowledge) ε-DP sublinear time algorithm for estimating the number of triangles, connected components, and the weight of a minimum spanning tree whose accuracy guarantees hold with high probability.

For estimating the average degree of a graph, in recent work, [BGM22] gave a pure ε-DP algorithm that achieves an $(\alpha, 0)$-approximation — a crucial observation is that their analysis only holds under the assumption that the average degree is at least one i.e., $\bar{d} \geq 1$ (see Section 4 for details). In this work, we remove the need for this assumption in the DP setting, by directly applying our black-box DP transformation to the original algorithm of [GR04] which works substantially better whenever we have $m = \omega(n)$ edges.

For estimating the maximum matching size in a graph, although [BGM22] gave an ε-DP algorithm for estimating the maximum matching size that achieves a 2-multiplicative factor and κn additive factor, they left the task of finding an $(0, \kappa n)$-approximation in the DP setting as an open problem. In this work, we partially resolve this problem by presenting an ε-DP algorithm that gives a $(0, \mathcal{O}\left(\frac{\kappa n}{\varepsilon}\right))$-approximation of the maximum matching size. Crucially, our resulting analysis cannot guarantee that the added Laplace noise will be small with high probability, but only guarantees this will be the case with *constant probability*. This problem highlights a limitation of our black-box DP framework — if the non-DP algorithm that we want to apply our DP transformation on has a time/space/query complexity that has an exponential dependence on the approximation parameters then the resulting DP algorithm that achieves a similar approximation guarantee with high probability may be highly inefficient in terms of time/space/query complexity.

We also show how to apply our DP framework to an algorithm estimating the distance to bipartiteness in dense graphs [GMRS22, AdlVKK03], which is accurate with probability $1 - o(1)$. The same reduction can be similarly applied to other natural properties that enjoy the feature that they admit distance-estimation algorithms with $\mathrm{poly}(1/\kappa)$ query complexity, where κ is the additive (normalized) error. For example, in the fundamental results of [GGR98] an efficient distance approximation algorithm for the maximum k-cut problem, and thus k-colorability is presented. [FR21], also based on results from [AE02], generalizes these properties to the notion of "semi-homogeneous partition properties" and show efficient distance estimation algorithms for properties such as Induced P_3-freeness, induced P_4-freeness, and chordality.[4]

DP Streaming Results. We also apply our framework given by Theorem 1.3 and Theorem 1.5 to obtain differentially-private streaming algorithms for many fundamental problems, i.e., see Table 2 and Table 3. We remark that while the accuracy guarantees of our resulting algorithms may be surpassed by recent works studying these problems on an individual basis, our applications are black-box reductions that avoid individual utility and privacy analysis of each non-private streaming algorithm, which can be heavily involved and quite non-trivial, e.g., [MMNW11, BBDS12, SST20, BGK$^+$21, WPS22, BMWZ22].

In the streaming model, elements of an underlying dataset arrive one-by-one and the goal is to compute or approximate some predetermined function on the dataset using space that is sublinear in the size of the dataset. Our reductions also have wide applications to various archetypes of data stream models, which we now discuss. In insertion-only streams, the updates of the stream increment the underlying dataset, such as adding edges to a graph, adding terms to a sequence, or increasing the coordinates of a frequency vector. In turnstile (or dynamic) streams, the updates of the stream can both increase and decrease (or insert and delete) elements of the underlying dataset. Finally, in the sliding window model, only the W most recent updates of the data stream define the underlying dataset. Both the turnstile streaming model and the sliding window model are generalizations of insertion-only streams, and our framework has implications in all three models.

We first show that our framework can be applied to existing non-private dynamic algorithms for weighted minimum spanning tree, L_p norm estimation for $p \geq 1$ (and also F_p moment estimation for $0 < p < 1$), and distinct elements estimation. Thus using our framework, we essentially get private dynamic algorithms for these problems for free (in terms of correctness, not optimality). Since the dynamic streaming model generalizes the insertion-only streaming model, we also obtain private streaming algorithms in the insertion-only model as well. We summarize these results in Table 2.

We then apply our framework in Theorem 1.3 to the sliding window model. To that end, we first recall that given a $(\alpha, 0)$-approximation algorithm for the insertion-only streaming model, the smooth histogram framework [BO10] provides a transformation that obtains a $(\alpha, 0)$-approximation algorithm in the sliding window model for a "smooth" function. Although there are problems that are known to not be smooth, e.g., [BOZ12, BDM$^+$20, BWZ21, EMMZ22, JWZ22], the smooth histogram framework does provide a $(\alpha, 0)$-approximation to many important problems, such as counting, longest increasing subsequence, L_p norm estimation for $p \geq 1$ (and also F_p moment estimation for $0 < p < 1$), and distinct elements estimation. We remark that if we tried to apply the non-private smooth histogram framework to a DP insertion-only streaming algorithm, this might preserve privacy by post-processing, but may significantly increase the error in terms of accuracy. On the other hand, our framework avoids these issues and achieves private analogs of these algorithms in the sliding window model without compromising utility. We summarize our

[4]In general, distance estimation is closely related to tolerant testing [PRR06], and for dense graph properties it is known that if a property is testable with a number of queries of the form $f(\kappa)$, then they admit a distance estimator [FN07] with an exponential blowup in $\frac{1}{\kappa}$ in the query complexity. Hence, in its general form the query complexity of estimating the distance to "hereditary" graph properties is a tower of exponential of height $\mathrm{poly}(1/\kappa)$ [AFNS09].

Problem	Reference	Privacy	Mult. error	Add. error	Space Complexity
Weighted Minimum Spanning Tree	[AGM12]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha} n \log^4 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{M \log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\alpha\varepsilon} n \log^5 n\right)$
L_p-norm, $p > 2$	[GW18]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} n^{1-\frac{2}{p}} \log^2 n + \frac{1}{\alpha^{4/p}} n^{1-\frac{2}{p}} \log^{2/p} n \log^2 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{p}{\alpha^2\varepsilon^2} n^{1-2/p}\right) \cdot \text{poly}\left(\log n, \log \frac{1}{\alpha\varepsilon}\right)$
L_p-norm, $p = 2$	[AMS99]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} \log^2 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\alpha^2\varepsilon^2} \log^4 n\right)$
L_p-norm, $p \in (0,2)$	[KNPW11]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} \log^2 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\alpha^2\varepsilon^2} \log^4 n\right)$
L_p-norm, $p = 0$	[KNW10]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} \log^2 n \log \frac{1}{\alpha}\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\tilde{\mathcal{O}}\left(\frac{1}{\alpha^2\varepsilon^2} \log^4 n\right)$

Table 2: Summary of DP algorithms in the dynamic/turnstile model obtained via our black-box DP transformation. According to our notation multiplicative error α means a multiplicative factor of $(1 \pm \alpha)$. See Section 5.1 for details.

results for the sliding window model in Table 3. We note that in recent work, [EMM$^+$23] give a generalized smooth histogram approach to convert a DP continual release streaming algorithm into a sliding window algorithm in the continual release setting. We focus on the one-shot streaming setting in our work.

Problem	Reference	Privacy	Mult. error	Add. error	Space Complexity
Longest Increasing Subsequence	[SW07]	Non-Private	α	0	$\mathcal{O}\left(\frac{k^2}{\alpha} \log^2 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{k^2}{\alpha\varepsilon} \log^4 n\right)$
Distinct Elements	[Bla20]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^3} \log^2 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\alpha^3\varepsilon^3} \log^5 n\right)$
L_p-norm, $p = 2$	[WZ21]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} \log^3 n \log^3 \frac{1}{\alpha}\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\tilde{\mathcal{O}}\left(\frac{1}{\alpha^2\varepsilon^2} \log^5 n \log^3 \frac{1}{\alpha\varepsilon}\right)$
L_p-norm, $p \in (0,2)$	[WZ21]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} \log^3 n (\log\log n)^2 \log^3 \frac{1}{\alpha}\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\tilde{\mathcal{O}}\left(\frac{1}{\alpha^2\varepsilon^2} \log^5 n\right)$

Table 3: Summary of DP algorithms in the sliding window model obtained via our black-box DP transformation. According to our notation multiplicative error α means a multiplicative factor of $(1 \pm \alpha)$. See Section 5.2 for details.

1.2 Our Techniques

Given a tunable (α, κ, δ)-approximation algorithm $\mathcal{A}_f$ for the function $f : \mathcal{D} \to \mathbb{R}^+$, our goal is to obtain a differentially private approximation algorithm that achieves a target $(\alpha', \kappa', \delta')$-approximation of f where α', κ' are in terms of α, κ.

Warm-up: When $\mathcal{A}_f$ is deterministic and only has multiplicative error. For simplicity, let us first consider an $(\alpha, 0, 0)$-approximation algorithm $\mathcal{A}_f$, in other words, $\mathcal{A}_f$ *always* outputs a value such that $(1-\alpha)f(D) \le \mathcal{A}_f(D) \le (1+\alpha)f(D)$. Since we want to make $\mathcal{A}_f$ differentially private, intuitively, we need to add noise to the output of $\mathcal{A}_f$. The local sensitivity of $\mathcal{A}_f$ at D (i.e., $LS_{\mathcal{A}_f}(D) = \max_{D' \sim D} |\mathcal{A}_f(D) - \mathcal{A}_f(D')|$) is upper bounded by $2\alpha f(D) + \Delta_f$. Since Δ_f is small and we can tune α to be arbitrarily small, it is tempting to think that we can just add noise proportional to $2\alpha f(D) + \Delta_f$. Unfortunately, scaling noise proportional to local sensitivity is not necessarily private. On the other hand we could ensure privacy by scaling noise proportional to the global sensitivity (i.e., $\max_{D\in\mathcal{D}} LS_{A_f}(D) \le \max_{D\in\mathcal{D}} 2\alpha f(D)+\Delta_f$) but noise will likely be too large to obtain meaningful accuracy guarantees. We adopt the strategy of adding noise proportional to the smooth sensitivity [NRS07] of $\mathcal{A}_f$ instead. In particular, [NRS07] observed that if we can find a "sufficiently smooth" function $S_f(D) \ge LS_{\mathcal{A}_f}(D)$ upper bounding the local sensitivity of $\mathcal{A}_f$ then we can preserve privacy by computing $\mathcal{A}_f(D)$ and adding noise scaled according to $S_f(D)$.

We can show that the function $S_f(D) = 4\alpha\mathcal{A}_f(D)+\Delta_f$ is a β-smooth upper bound on the local sensitivity of $\mathcal{A}_f$ for $\beta = 6\alpha$ where S_f is β-smooth if $S_f(D) \le e^{\beta} S_f(D')$ for all pairs of neighboring datasets $D \sim D'$. To achieve privacy using the smooth sensitivity framework we need to ensure that β is sufficiently small relative to our privacy parameters ε and δ (if applicable). For example, we can achieve $\left(\varepsilon, \delta\left(1+\exp\left(\frac{\varepsilon}{2}\right)\right)\right)$-differential privacy by adding Laplace Noise scaled by $\frac{2S_f(D)}{\varepsilon}$, but only if S_f is β-smooth for $\beta \le \frac{\varepsilon}{2\ln(2/\delta)}$. For pure differential privacy we require that $\beta < \frac{\varepsilon}{2(\lambda+1)}$ where λ is a parameter of the noise distribution — smaller λ implies higher variance.

If we want to ensure that the output is accurate, we also need to ensure that the calibrated noise with $S_f(D)$ is small e.g., $o(f(D)) + \mathcal{O}(\Delta_f)$. Note that by definition since $S_f(D) = 4\alpha\mathcal{A}_f(D) + \Delta_f$, and we add noise proportional $S_f(D)$, we expect that the noise added may be $> \alpha f(D)$. Thus, in order to address this challenge, our basic strategy is to run the original (non-private) approximation algorithm $\mathcal{A}_f$ with *tuned* error factors e.g., we decrease α by a multiplicative factor of $\frac{\varepsilon}{\ln(n)}$, let $\rho := \frac{\varepsilon\alpha}{\ln(n)}$. Since we are now running $\mathcal{A}_f(D, \rho, 0, 0)$, we have that the function $S_f(D) = 4\rho\mathcal{A}_f(D)+\Delta_f$ is a β-smooth upper bound on the local sensitivity of the algorithm $\mathcal{A}_f(\cdot, \rho, 0, 0)$. Assuming the global sensitivity Δ_f is small, we can now show that w.h.p. the noise sampled proportional to $S_f(D)$ is at most $\alpha f(D) + O(\Delta_f/\varepsilon)$ thus resulting in an ε-differentially private algorithm with reasonable accuracy.

By tuning the parameter α we actually accomplish two useful properties (1. accuracy) we decrease both the local sensitivity and our smooth upper bound $S_f(D)$ which reduces the magnitude of the noise that we add, and (2. privacy) we achieve β-smoothness for increasingly small values of β so that the required condition $\beta \le \frac{\varepsilon}{2\ln(2/\delta)}$ (or $\beta < \frac{\varepsilon}{2(\lambda+1)}$) can be satisfied if we want to scale noise according to $S_f(D)$.

Extending to deterministic $\mathcal{A}_f$ with multiplicative and additive error. More generally, if we have an $(\alpha, \kappa, 0)$-approximation algorithm $\mathcal{A}_f$ then we can show that $S_f(D) = 4\alpha\mathcal{A}_f(D)+\Delta_f+4\tau$ is a β-smooth upper bound on the local sensitivity of $\mathcal{A}_f$ with $\beta = 6\alpha$ (see Lemma 3.8). In particular, note that the additive error term κ does not adversely impact smoothness. Thus, we can achieve pure differentially privacy by tuning α such that $6\alpha < \beta < \frac{\varepsilon}{2(\lambda+1)}$ and scaling our noise according to $S_f(D)$ (see Lemma 3.9). We can also obtain stronger accuracy guarantees by relaxing the requirement for pure DP and tuning α such that $6\alpha \le \beta \le \frac{\varepsilon}{2\ln(2/\delta)}$ so that we can sample our noise from the Laplace distribution which has strong concentration guarantees.

When $\mathcal{A}_f$ is randomized. The remaining challenge is to handle randomized approximation algorithms $\mathcal{A}_f$ which are only guaranteed to output a good approximation with high probability i.e., with non-zero probability $\delta > 0$ the algorithm is allowed to output an arbitrarily bad approximation. In particular, let us consider an (α, κ, δ)-approximation algorithm $\mathcal{A}_f$. For any possible input D we are always guaranteed that with probability $\ge 1-\delta$ the algorithm $\mathcal{A}_f(D)$

outputs a good approximation $(1-\alpha)f(D) \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D)$. Unfortunately, the function $S_f(D) = 4\alpha\mathcal{A}_f(D) + \Delta_f + 4\kappa$ is no longer guaranteed to be a β-smooth upper bound on the local sensitivity of $\mathcal{A}_f$ since $\mathcal{A}_f$ may sometimes output a value outside the specified approximation bounds.

In order to address this challenge, we define a function $g_f(D)$ that matches $\mathcal{A}_f(D)$ with probability at least $1-\delta/2$ and is *always* guaranteed to output a good approximation. We emphasize that $g_f(D)$ may not be efficiently computable, but it is well-defined and only used for the purpose of analysis. More specifically, we set $g_f(D) = \mathcal{A}_f(D)$ as long as $(1-\alpha)f(D) - \kappa \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D) + \kappa$. If $\mathcal{A}_f(D) > (1+\alpha)f(D)$, then we define $g_f(D) := (1+\alpha)f(D)$, similarly, if $\mathcal{A}_f(D) < (1-\alpha)f(D)$, then we define $g_f(D) := (1-\alpha)f(D) + \kappa$. Observe that we are always guaranteed that $(1-\alpha)f(D) - \kappa \leq g_f(D) \leq (1+\alpha)f(D) + \kappa$. Thus, $S_f(D) = 4\alpha g_f(D) + \Delta_f + 4\tau$ is a $\beta = 6\alpha$-smooth upper bound on the local sensitivity of g_f (see Lemma 3.2). As long as $6\alpha \leq \beta \leq \frac{\varepsilon}{2\ln(2/\delta)}$ we could preserve $\left(\varepsilon, \delta\left(1+\exp\left(\frac{\varepsilon}{2}\right)\right)\right)$-differential privacy by outputting $g_f(D)$ plus Laplace Noise scaled by $\frac{8\alpha g_f(D)+\Delta_f+8\tau}{\varepsilon}$ i.e., scaled according to our β-smooth upper bound on the local sensitivity of g_f. Unfortunately, the function g_f may not be efficiently computable. Thus, we substitute g_f for $\mathcal{A}_f$ and instead output $\mathcal{A}_f(D)$ plus Laplace noise scaled according to $\frac{8\alpha\mathcal{A}_f(D)+\Delta_f+8\tau}{\varepsilon}$. While $4\alpha\mathcal{A}_f(D) + \Delta_f + 4\tau$ is not necessarily a β-smooth upper bound on the local sensitivity of $\mathcal{A}$, the key point is that the latter (efficiently computable) procedure is equivalent to the former (differentially private) procedure as long as $g_f(D) = A_f(D)$ which happens as long as $\mathcal{A}_f$ outputs a good approximation i.e., except with probability $\delta/2$. Thus, we can apply a hybrid argument to argue that the final efficiently computable algorithm is $\left(\varepsilon, \frac{\delta}{2} + \delta\left(1+\exp\left(\frac{\varepsilon}{2}\right)\right)\right)$-differential privacy (see Lemma 3.4). In order to ensure accuracy, we use the same strategy as before, i.e., we run $\mathcal{A}_f(D, \rho, \tau, \delta/2)$, where $\rho \leq \frac{\varepsilon\alpha}{\log(1/\delta)}$. Sampling noise proportional to $S_f(D)$ (where $S_f(D)$ is now defined in terms of ρ), and absorbing the failure probability of algorithm $\mathcal{A}_f$ into the DP failure probability term δ, results in an approximate differentially private algorithm. Finally applying the postprocessing step results in a pure differentially private algorithm. We refer to the full proofs (Section 3) for additional details.

Applications. We give some intuition on how we apply Theorem 1.3 to various applications by choosing appropriate parameters. Recall that with probability $1-\delta-\exp(-\gamma)$, $\mathcal{A}'_f$ outputs $(1-\alpha')f(D) - \kappa' - \frac{2\Delta_f}{\varepsilon}\cdot\gamma \leq \mathcal{A}'_f(D) \leq (1+\alpha')f(D) + \kappa' + \frac{2\Delta_f}{\varepsilon}\cdot\gamma$, where $\alpha' = \frac{\alpha(\varepsilon+16\gamma)}{12\log(4/\delta)}$, and $\kappa' = \kappa\left(\frac{2\gamma\alpha}{3\log(4/\delta)} + \frac{8\gamma}{\varepsilon} + 1\right)$ with a time/space/query complexity blow-up incurred by running the original algorithm $\mathcal{A}_f$ with multiplicative accuracy parameter $\rho = \frac{\varepsilon\alpha}{\log(4/\delta)}$. First, observe that if the original algorithm $\mathcal{A}_f$ has time/space/query complexity with a dependence on $\text{poly}(\frac{1}{\alpha})$, then the resulting time/space/query complexities for $\mathcal{A}'_f$ will still have a polynomial dependence, i.e., $\text{poly}(\frac{\log(4/\delta)}{\alpha})$ — this naturally leads to FPRAS or FPTAS applications, as well as other classes of approximation algorithms like sublinear time or space. On the otherhand, if the time/space/query complexity of $\mathcal{A}_f$ has a non-polynomial dependence on $1/\alpha$, e.g., $\exp(\frac{1}{\alpha})$, then since δ is typically $\mathsf{negl}(n)$ or $\frac{1}{n^c}$ for $c > 0$, the resulting DP algorithm $\mathcal{A}'_f$ could have much worse time/space/query-guarantees with respect to n, e.g., in an extreme case if we set $\delta = 2^{-\text{poly}(n)}$ then $\rho = \Omega(\text{poly}(n)/\alpha)$ and we could incur a $\exp(\frac{\text{poly}(n)}{\alpha})$ multiplicative overhead in the running time. It is worth noting that one could optionally reduce the additive error term κ' for $\mathcal{A}'_f$ by reducing the error term κ for $\mathcal{A}$.

We further emphasize this trade-off between obtaining small failure probability bounds and the accuracy or resource guarantees. Consider the following two examples— (Example 1) if we set the probability of failure, i.e., $\exp(-\gamma) = \delta = \frac{1}{n^c}$ for any $c > 0$, then the resulting approximation parameters are roughly $\alpha' = \alpha(1 + o(1))$, and $\kappa' = \kappa(\alpha + \frac{\log(n)}{\varepsilon} + 1)$[5], and the additional error term depending on global sensitivity is roughly $\frac{\Delta_f \log(n)}{\varepsilon}$. We incur a time/space overhead by running $\mathcal{A}_f$ with multiplicative accuracy parameter $\rho = \Omega(\varepsilon\alpha/\log n)$ instead of α. (Example 2) if we set the probability of failure, i.e., $\exp(-\gamma) = \delta = \frac{1}{n^{\log\log(n)}} - \mathsf{negl}(n)$, then α' remains the same as before, and now $\kappa' = \kappa(\alpha + \frac{\log(n)\log\log(n)}{\varepsilon} + 1)$, but the additional error term depending on global sensitivity becomes $\frac{\Delta_f \log(n)\log\log(n)}{\varepsilon}$. In this latter case we incur time or space overhead by running $\mathcal{A}_f$ with multiplicative accuracy parameter $\rho = \Omega\left(\frac{\varepsilon\alpha}{\log n \log\log n}\right)$ — to reduce the $\kappa' \log n \log\log n$ error term it could be useful to run $\mathcal{A}_f$ with additive error parameter $\frac{\kappa}{\log n \log\log n}$ which may incur additional time or space overhead. Thus these two examples illustrate how, as we decrease the failure probability, the accuracy and resource (time/space in this case) guarantees become worse. See Section 4, and Section 5 for applications to sublinear time and streaming algorithms.

1.3 Related Work

One of the first differentially private frameworks for computing general functions was introduced by [DMNS16] which released functions with additive noise, where the noise is calibrated according to the *global sensitivity* of the function f. This framework was generalized by [NRS07], to handle functions which might have a high global sensitivity but are usually less sensitive in practice. The framework allows the release of functions with instance-specific noise, where the noise that is added is not just determined by f but by the input dataset as well. The noise magnitude calibrated is according to the *smooth sensitivity* of f on the input dataset which is a smooth upper bound on the *local sensitivity* of f on an input dataset. The smooth sensitivity of a function may be hard to compute, therefore in the same work, [NRS07] give a generic method called the *sample and aggregate* method that bypasses the explicit computation of the smooth sensitivity of the function and works even when the function is given as a black-box. [DL09] suggested a framework called *Propose-Test-Release* to release statistical estimators with additive noise where the noise is calibrated according to the *local sensitivity* of the estimator. Note that adding noise proportional to the local sensitivity of a function with respect to an input set usually does not preserve privacy, but their approach first proposes a bound on the local sensitivity and privately tests whether this bound holds for the specific input set, and then releases the noisy response to the query.

In the context of developing differentially private frameworks for approximation algorithms, [BGM22] formally introduced the notion of *coupled global sensitivity* of a randomized algorithm, which gives an analogous framework as that of the global sensitivity framework [DMNS16], but for randomized approximation algorithms instead of deterministic functions. In this framework, one can run a non-private randomized approximation algorithm $\mathcal{A}_f(D)$ on the dataset, and privacy is obtained by adding noise proportional to the coupled global sensitivity of $\mathcal{A}_f$. More formally, the coupled global sensitivity measures the worst-case L_1-sensitivity of the outputs of a randomized

[5]In the applications we consider the original (non-private) approximation algorithm typically has only multiplicative or only additive error and not both. In particular, we typically either have $\alpha > 0$ and $\kappa = 0$ or $\kappa > 0$ and $\alpha = 0$, but not the case where $\alpha > 0$ and $\kappa > 0$. Considering the case when $\kappa \neq 0$, and $\alpha = 0$, we (roughly) have $\kappa' = \kappa(\frac{\log(n)}{\varepsilon} + 1)$.

algorithm $\mathcal{A}_f$ on neighboring inputs over a minimum coupling of the internal coin tosses of $\mathcal{A}_f$.

In independent work, Tetek [Tet22] also explores the problem of transforming randomized approximation algorithms into (pure) differentially private approximation algorithms. In contrast to our results Tetek's transformation [Tet22] assumes that the error of the original approximation algorithm either has small subexponential diameter or bounded mean error — assumptions that would not apply generically to every (tunable) approximation algorithm. Assuming subexponential error their work shows that it is possible to achieve ε-DP by adding Laplace Noise yielding accuracy guarantees that hold with high probability. However, the assumption of the error being subexponential is quite strong and does not often hold for many randomized approximation algorithms. While assuming bounded mean error is a weaker assumption on the error of the non-private randomized algorithm, however the DP noise is sampled from the Pareto distribution, which has polynomial tail bounds. This leads to accuracy guarantees which only hold with constant probability. Note that applying the median trick commonly used to amplify success probability in the non-private literature adversely affects the privacy budget and is thus not desirable. In contrast, our transformation applies generically to any (tunably) accurate approximation algorithm and we achieve accuracy guarantees that hold with high probability for the same problems studied in their paper. Finally, we correct an outdated claim[6] from the comparison to our work detailed in [Tet22] that says that we only achieve approximate privacy. We can achieve *pure* DP algorithms by applying a postprocessing step to the output of our transformation as outlined in Theorem 1.5.

2 Preliminaries

We use the notation $\tilde{\mathcal{O}}(f(n))$ to mean $f(n) \cdot \text{polylog}(f(n))$. We define datasets D and D' as *neighboring*, denoted as $D \sim D'$, if removing or adding one point in D results in D'; alternatively, if changing one data point in D results in D'.

Definition 2.1 (Differential privacy). *[DMNS06] An algorithm $\mathcal{A}$ is (ε, δ)-DP if for every pair of neighboring datasets $D \sim D'$, and for all sets $\mathcal{S}$ of possible outputs, we have that $\Pr[\mathcal{A}(D) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[\mathcal{A}(D') \in \mathcal{S}] + \delta$. When $\delta = 0$ we simply say that the algorithm is ε-DP.*

Given an (ε, δ)-DP algorithm, one can obtain an ε-DP algorithm under certain conditions outlined below. We include the proof for completeness in Appendix A.

Theorem 2.2. *[Approximate DP to Pure DP] Let $\mathcal{A} : \mathcal{D} \rightarrow \mathcal{R}$. If $\mathcal{A}$ is an (ε, δ)-DP algorithm such that $\delta \leq \frac{(e^{\varepsilon}-1)p}{|\mathcal{R}|(1-p)}$ then there is an algorithm $\mathcal{A}'$ such that $\mathcal{A}'$ is ε-DP defined in the following manner.*

$$\mathcal{A}'(D) = \begin{cases} \mathcal{A}(D) & \textit{with probability } 1-p \\ \mathsf{random}(\mathcal{R}) & \textit{with probability } p \end{cases}$$

where $\mathcal{R}$ is the range of $\mathcal{A}_f$.

We define the distributions we will use to sample additive noise from below.

[6]A prior version of the paper achieved pure DP, but that transformation (Theorem 1.5) only applied to deterministic tunable approximation algorithms

Definition 2.3 (Laplace distribution). *We say a random variable X is drawn from a Laplace distribution with mean μ 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 $\mu = 0$.*

Definition 2.4 (Cauchy distribution). *We say a random variable X is drawn from a Cauchy distribution with location parameter x_0 and scale $b > 0$ if the probability density function of X at x is $\frac{1}{\pi b}\left(\frac{b^2}{(x-x_0)^2+b^2}\right)$. We use the notation $X \sim \mathsf{C}(b)$ to denote that X is drawn from the Cauchy distribution with scale b and location parameter $x_0 = 0$.*

We formally define the concept of global sensitivity which is a worst-case notion of sensitivity for deterministic functions below.

Definition 2.5 (Global sensitivity). *The global sensitivity of a function $f : \mathcal{D} \to \mathbb{R}^d$ is defined by*

$$\Delta_f = \max_{D, D' \in \mathcal{D}, D \sim D'} \|f(D) - f(D')\|_1.$$

We define the notion of local sensitivity for a fixed input, which can be much smaller than the global sensitivity, but in general, adding noise calibrated according to the local sensitivity does not preserve DP.

Definition 2.6 (Local sensitivity). *For $f : \mathcal{D} \to \mathbb{R}$ and $D \in \mathcal{D}$, the local sensitivity of f at D is defined as*

$$LS_f(D) = \max_{D' : D \sim D'} \|f(D) - f(D')\|_1.$$

Note: if $f : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$ is a randomized function which, in addition to a dataset $D \in \mathcal{D}$ takes random coins $r \in \mathcal{R}$ as input we simply define $LS_f(D) = \max_{r \in \mathcal{R}} LS_{f_r}$ where $f_r(D) \doteq f(D; r)$.

In order to add instance-specific noise, we define the notions of β-smooth upper bound which is a smooth upper bound on the local sensitivity.

Definition 2.7 (Smooth upper bound on local sensitivity). *For $\beta > 0$, a function $S : \mathcal{D} \to \mathbb{R}$ is a β-smooth upper bound on the local sensitivity of $f : \mathcal{D} \to \mathbb{R}$ if*

(1) For all $D \in \mathcal{D}$, we have $S(D) \geq LS_f(D)$.

(2) For all $D, D' \in \mathcal{D}$ with $\|D - D'\|_1 = 1$, we have $S(D) \leq e^{\beta} \cdot S(D')$.

Finally, although one cannot add noise calibrated with local sensitivity, one can add noise proportional to a β-smooth upper bound on the local sensitivity as follows.

Theorem 2.8 (Corollary 2.4 in [NRS07]). *Let $f : \mathcal{D} \to \mathbb{R}$ and $S : \mathcal{D} \to \mathbb{R}$ be a β-smooth upper bound on the local sensitivity of f.*

(1) If $\beta \leq \frac{\varepsilon}{2(\lambda+1)}$ and $\lambda > 1$, the algorithm $D \to f(D) + \frac{2(\lambda+1)S(D)}{\varepsilon} \cdot \eta$, where η is sampled from the distribution with density $h(z) \propto \frac{1}{1+|z|^{\lambda}}$, is ε-differentially private.

(2) If $\beta \leq \frac{\varepsilon}{2\ln(2/\delta)}$ and $\delta \in (0,1)$, then the algorithm $D \to f(D) + \frac{2S(D)}{\varepsilon} \cdot \eta$ where $\eta \sim \mathsf{Lap}(1)$ is (ε, δ')-differentially private for $\delta' = \delta\left(1 + \exp\left(\frac{\varepsilon}{2}\right)\right)$[7].

[7]These bounds differ slightly from those listed in the original paper (Corollary 2.4 in [NRS07]). We confirmed with the authors in private communication that δ should be multiplied by $(1 + \exp(\varepsilon/2))$.

3 General Transformation for Approximation Algorithms

In this section, we formally define our black-box differentially private transformation for (randomized) approximation algorithms. Given a tunable approximation (see Definition 1.2) algorithm of f, call it $\mathcal{A}_f$, that outputs an (α, κ, δ)-approximation, our framework for randomized algorithms involves two steps — (1) Apply Algorithm 1 to $\mathcal{A}_f$ to obtain an (ε, δ)-DP algorithm $\mathcal{A}'_f$ with accuracy guarantees outlined in Theorem 1.3 (2) Apply postprocessing step to the output of $\mathcal{A}'_f$ to obtain an ε-DP algorithm (see Theorem 1.5).

We first prove Theorem 1.3 that provides theoretical guarantees for algorithm $\mathcal{A}'_f$ (Algorithm 1). This is our main contribution as the postprocessing step to obtain pure DP applies a folkore result.

Observe that even for the case when the original algorithm $\mathcal{A}_f$ gives an $(\alpha, 0, \delta)$-approximation of f (i.e., $\kappa = 0$), the resulting DP algorithm $\mathcal{A}'_f$ will still have an additive error, this additive error is inherent due to the requirement of adding Laplace noise to preserve DP. We emphasize that the Laplace noise added to the output of algorithm $\mathcal{A}_f$ depends on the global sensitivity of the function f, therefore, we can only get meaningful DP approximation algorithms using this transformation for functions with low global sensitivity.

Theorem 1.3. *((ε, δ)-privacy) Suppose that $\mathcal{A}_f$ is a tunable approximation of $f : \mathcal{D} \to \mathbb{R}^+$. Then for all $\varepsilon > 0$, $\delta = \delta(n) > 0$[8], $\alpha \geq 0$ and $\kappa \geq 0$, there is an algorithm $\mathcal{A}'_f$ such that*

(1) (Privacy) $\mathcal{A}'_f$ is (ε, δ')-differentially private where $\delta' = \delta(1 + \exp(\varepsilon/2))$.

(2) (Accuracy) For all $D \in \mathcal{D}$, and $0 < \gamma$, with probability $1 - \delta - \exp(-\gamma)$,

$$(1-\alpha')f(D) - \kappa' - \frac{2\Delta_f}{\varepsilon} \cdot \gamma \leq \mathcal{A}'_f(D) \leq (1+\alpha')f(D) + \kappa' + \frac{2\Delta_f}{\varepsilon} \cdot \gamma$$

where $\alpha' = \frac{\alpha(\varepsilon+16\gamma)}{12\log(4/\delta)}$, and $\kappa' = \kappa\left(\frac{2\gamma\alpha}{3\log(4/\delta)} + \frac{8\gamma}{\varepsilon} + 1\right)$, and $\Delta_f := \max_{D,D' \in \mathcal{D}, D \sim D'} \|f(D) - f(D')\|_1$.

(3) (Resource) $\mathcal{A}'_f$ uses $R\left(n, \frac{\varepsilon\alpha}{\log(4/\delta)}, \kappa, \delta\right)$ resource, where $R(\cdot,\cdot,\cdot,\cdot)$ is the resource used by $\mathcal{A}_f$.

Proof. $\mathcal{A}'_f$ is defined in Algorithm 1 — it first runs $\mathcal{A}_f(D, \rho, \kappa, \delta/2)$ where $\rho := \frac{\varepsilon\alpha}{12\log(4/\delta)}$ and then adds Laplace Noise. Thus, the resource used by $\mathcal{A}'_f$ is $R(n, \rho, \kappa, \delta)$. The privacy guarantee follows from Lemma 3.4, and the accuracy guarantee follows from Lemma 3.6. □

Remark 3.1. *When $\mathcal{A}_f$ is a PRAS, by definition, the output of $\mathcal{A}_f$ is an $(\alpha, 0, \delta)$-approximation of f running in time $T(n, \alpha, 0, \delta) = \mathrm{poly}(n, 1/\alpha, \log(1/\delta))$. Applying Theorem 1.3 with negligible $\delta = n^{-\log n}$ and $\gamma = \log^2 n$ for any $\alpha' > 0$ we obtain a private $\left(\alpha', O\left(\frac{\Delta_f}{\varepsilon \log^2 n}\right), 2n^{-\log n}\right)$-approximation with polynomial running time $\mathrm{poly}(n, 1/\varepsilon, 1/\alpha')$.*

Lemma 3.2. *Let $0 < \rho < 1/2$. Suppose that $\mathcal{A}_f$ outputs a (ρ, τ, δ)-approximation of a function $f : \mathcal{D} \to \mathbb{R}^+$ with global sensitivity Δ_f. Let $\mathcal{A}_{f,R}$ denote a deterministic run of $\mathcal{A}$ using a fixed set of random coins R. Define function $g_{f,R}$ by*

$$g_{f,R}(D) = \begin{cases} \mathcal{A}_{f,R}(D) & \text{if } (1-\rho)f(D) - \tau \leq \mathcal{A}_{f,R}(D) \leq (1+\rho)f(D) + \tau \\ (1-\rho)f(D) - \tau & \text{if } \mathcal{A}_{f,R}(D) < (1-\rho)f(D) - \tau \\ (1+\rho)f(D) + \tau & \text{if } \mathcal{A}_{f,R}(D) > (1+\rho)f(D) + \tau \end{cases}$$

[8]typically we set $\delta = \mathrm{negl}(n)$ or $\delta = n^{-c}$ for some constant $c > 0$. In particular $\delta(n)$ may approach zero as $n \to \infty$.

Then the function $S_f(D) = 4\rho g_{f,R}(D) + 4\tau + \Delta_f$ is a β-smooth upper bound for $g_{f,R}$ where $\beta \geq 6\rho$.

Proof. Fix an arbitrary set of random coin tosses R. We frequently use the fact that $(1-\rho)f(D) - \tau \leq g_{f,R}(D) \leq (1+\rho)f(D) + \tau$. We also note that since $0 \leq \rho < 1/2$, we have that $\frac{1}{2}f(D) - \tau \leq g_{f,R}(D) \leq 2f(D) + \tau$.

First, we show that Condition 1 of Definition 2.7 holds. Without loss of generality, we assume $f(D) \geq f(D')$, where D' is any neighboring input. Then:

$$\begin{aligned}
LS_{g_{f,R}}(D) &= \max_{D':D\sim D'} \|g_{f,R}(D) - g_{f,R}(D')\| \\
&\leq \|(1+\rho)f(D) + \tau - (1-\rho)f(D') + \tau\| \\
&\leq \rho\|f(D) + f(D')\| + 2\tau + \Delta_f \\
&\leq 2\rho f(D) + 2\tau + \Delta_f \\
&\leq 4\rho g_{f,R}(D) + 2\rho\tau + 2\tau + \Delta_f && \text{as } \frac{1}{2}f(D) - \tau \leq g_{f,R}(D) \\
&\leq 4\rho g_{f,R}(D) + 4\tau + \Delta_f \\
&= S_f(D)
\end{aligned}$$

Next, we show that Condition 2 of Definition 2.7 holds below. We have:

$$\begin{aligned}
S_f(D) &= 4\rho g_{f,R}(D) + 4\tau + \Delta_f \\
&\leq 4\rho\,(1+\rho)\,f(D) + 4\rho\tau + 4\tau + \Delta_f && \text{by def of } g_{f,R} \\
&\leq 4\rho\,(1+\rho)\,(\Delta_f + f(D')) + 4\rho\tau + 4\tau + \Delta_f && \text{since } D \sim D' \\
&\leq 4\rho(1+\rho)f(D') + (4\rho(1+\rho)+1)\Delta_f + 4\rho\tau + 4\tau \\
&\leq 4\rho\frac{(1+\rho)}{1-\rho}(g_{f,R}(D') + \tau) + (1+6\rho)\Delta_f + 4\rho\tau + 4\tau && \text{by def of } g_{f,R} \\
&\leq 4\rho(1+\rho)(1+2\rho)(g_{f,R}(D') + \tau) + (1+6\rho)\Delta_f + 4\rho\tau + 4\tau \\
&\leq 4\rho(1+\rho)(1+2\rho)g_{f,R}(D') + 12\rho\tau + (1+6\rho)\Delta_f + 4\rho\tau + 4\tau \\
&\leq 4\rho(1+\rho)(1+2\rho)g_{f,R}(D') + (1+6\rho)(\Delta_f + 4\tau) \\
&\leq 4\rho(1+4\rho)g_{f,R}(D') + (1+6\rho)(\Delta_f + 4\tau) \\
&\leq (1+6\rho)(4\rho g_{f,R}(D') + \Delta_f + 4\tau) \\
&\leq e^{6\rho} \cdot (4\rho g_{f,R}(D') + \Delta_f + 4\tau) = e^{\beta} S_f(D'),
\end{aligned}$$

where $\beta \geq 6\rho$.

□

Remark 3.3. *As a special case if we have a $(0, \kappa, 0)$-approximation algorithm $\mathcal{A}_f$ (i.e., no multiplicative error, zero failure probability) then applying Lemma 3.2 yields the smooth upper bound $S_f(D) = 4\tau + \Delta_f$. We observe that this smooth upper bound is independent of D and, therefore, S_f is just an upper bound on the global sensitivity of g_f. Furthermore, in this special case we are guaranteed that $\mathcal{A}_f(D) = g_f(D)$ with probability 1. Thus, in this special case, we can achieve pure ε-DP by computing $\mathcal{A}_f(D)$ and adding Laplace noise proportional to S_f.*
If we have a $(0, \kappa, \delta)$-approximation algorithm for $\delta \neq 0$ then we still have $S_f(D) = 4\tau + \Delta_f$ which

means that $S_f(D)$ is an upper bound on the global sensitivity of g_f. However, computing $\mathcal{A}_f(D)$ and adding Laplace noise proportional to S_f does not necessarily yield a pure DP algorithm since we may have $\Delta_f(D) \neq \mathcal{A}_f(D)$ with non-zero probability δ. If we have $(\alpha, \kappa, 0)$-approximation algorithm $\mathcal{A}_f$, and $\alpha \neq 0$, since $S_f(D) = 4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f$ still depends on the input D. However, we can still achieve DP using Theorem 1.6.

Applying Lemma 3.2 to the theorem calibrating noise to smooth bounds on the smooth sensitivity [NRS07] we show that the Algorithm 1 preserves privacy below.

Lemma 3.4 (Privacy)**.** *Algorithm 1 is (ε, δ')-differentially private where $\delta' = \delta(1 + \exp(\varepsilon/2))$.*

Proof. Consider a modification of Algorithm 1, call it Algorithm 1' where instead of computing $\mathcal{A}_f(D, \rho, \tau, \delta/2)$ we instead sample the random coins R that $\mathcal{A}_f$ would have used and replace the value $\mathcal{A}_f(D, \rho, \tau, \delta/2; R)$ (which we denote as $\mathcal{A}_{f,R}(D)$ in the sequel) with $g_{f,R}(D)$.The function $g_{f,R}$ may not be efficiently computable, but we only use Algorithm 1' for the purpose of analysis. We first observe that by Lemma 3.2, for any $\beta \geq 6\rho$ the function $S_f(D) = 4\rho g_{f,R}(D) + 4\tau + \Delta_f$ is a β-smooth upper bound on the sensitivity of $g_{f,R}$. Thus, by Theorem 2.8, it is sufficient to set $\rho := \frac{\varepsilon\alpha}{12\log(4/\delta)}$ for $6\rho \leq \beta \leq \frac{\varepsilon}{2\ln\left(\frac{4}{\delta}\right)}$ and add noise proportional to $\mathsf{Lap}\left(\frac{2S_f(D)}{\varepsilon}\right) = \mathsf{Lap}\left(\frac{2(4\rho g_{f,R}(D)+4\tau+\Delta_f)}{\varepsilon}\right)$ to preserve $(\varepsilon, \delta/2)$-privacy of Algorithm 1'.

Since $g_{f,R}(D)$ is $\mathcal{A}_{f,R}(D)$ except with probability $\delta/2$, Algorithm 1 is identical to Algorithm 1' except with probability $\delta/2$. Thus, this shows that Algorithm 1 is (ε, δ)-private. □

Fact 3.5. *If $Y \sim \mathsf{Lap}(b)$, then $\Pr[|Y| \geq \ell \cdot b] = \exp(-\ell)$.*

Lemma 3.6 (Accuracy)**.** *For all $\gamma > 0$, with probability $1 - \exp(-\gamma) - \delta$,*

$$\big(1 - \rho(1 + \frac{16\gamma}{\varepsilon})\big)f(D) - \tau\big(\frac{8\gamma\rho}{\varepsilon} + \frac{8\gamma}{\varepsilon} + 1\big) - \frac{2\Delta_f\gamma}{\varepsilon} \leq \mathcal{A}'(D) \leq \big(1 + \rho(1 + \frac{16\gamma}{\varepsilon})\big)f(D)$$
$$+ \tau\big(\frac{8\gamma\rho}{\varepsilon} + \frac{8\gamma}{\varepsilon} + 1\big) + \frac{2\Delta_f\gamma}{\varepsilon}.$$

Proof. First, using Fact 3.5, for any $\gamma > 0$, we have that,

$$\mathbf{Pr}\left[|X| \geq \frac{2(4\rho\mathcal{A}(D) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma\right] = \exp(-\gamma)$$

$\mathcal{A}_f$ is a $(\rho, \tau, \delta/2)$-approximation of f so for any $D \in \mathcal{D}$, we have that $\mathcal{A}_f(D) \leq (1 + \rho)f(D) + \tau$ with probability $1 - \delta/2$. Since $0 < \rho < 1/2$ we have $(1 + \rho)f(D) + \tau \leq 2f(D) + \tau$. Therefore, by a union bound,

$$\Pr\big[\big(\mathcal{A}_f(D) > (1 + \rho)f(D) + \tau\big) \vee \big(\mathcal{A}_f(D) < (1 - \rho)f(D) - \tau\big) \vee \big(|X|$$
$$\geq \frac{2(4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma\big)\big] \leq \delta/2 + \exp(-\gamma)$$

Thus, with probability $1 - \exp(-\gamma) - \delta/2$, we have that

$$(1 - \rho)f(D) - \tau \leq \mathcal{A}_f(D) \leq (1 + \rho)f(D) + \tau \leq 2f(D) + \tau \qquad (1)$$

and

$$|X| < \frac{2(4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma \tag{2}$$

By plugging in Eq. 1 into Eq. 2, we have that with probability $1 - \exp(-\gamma) - \delta/2$,

$$|X| < \frac{2(4\rho(2f(D) + \tau) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma$$

Overall, this means that with probability $1 - \exp(-\gamma) - \delta/2$,

$$(1-\rho)f(D) - \tau - \frac{2(4\rho(2f(D) + \tau) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma \le \mathcal{A}'_f(D)$$
$$\le (1+\rho)f(D) + \tau + \frac{2(4\rho(2f(D) + \tau) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma$$

Grouping the like terms together gives the theorem statement. □

Theorem 1.5. *Let $M = \max_D f(D)$ and let parameter $K > 0$. If $\mathcal{A}'_f(D)$ is (ε, δ)-DP algorithm with accuracy guarantee $(1-\alpha)f(D) - \kappa \le \mathcal{A}_f(D) \le (1+\alpha)f(D) + \kappa$ holding with probability $1 - \eta$ then there exists an algorithm $\mathcal{A}''_f(D)$ which is ε-DP with accuracy guarantee $(1-\alpha)f(D) - \kappa - \frac{1}{KM} \le \mathcal{A}_f(D) \le (1+\alpha)f(D) + \kappa + \frac{1}{KM}$ with probability at least $1 - \eta - p$ where $p = \frac{\delta K(M+1)}{e^{\varepsilon} - 1 + \delta K(M+1)}$.*

The proof of Theorem 1.5 is in Appendix A. At a high level our idea is to define $\mathcal{A}'_f(D) = \frac{\lceil K\mathcal{A}_f(D)\rceil}{KM}$. The rounding step introduces a small additive error term $\le \frac{1}{KM}$ and ensures that $\mathcal{A}'_f(D)$ now has bounded range $|\mathcal{R}| \le (M+1)K$. Since $\mathcal{A}'_f$ has bounded range we can apply a folklore result (see Theorem 2.2) to transform this (ε, δ)-DP algorithm into an ε-DP algorithm.

3.1 Achieving Pure DP for Approximation Algorithms with Zero Failure Probability

In this section we show how one can achieve pure differential privacy ($\delta = 0$) when we have a tunable $(\alpha, \kappa, 0)$-approximation algorithm. The basic framework is the same except that we use the Cauchy distribution instead of Laplace when applying the Smooth Sensitivity framework — see Theorem 2.8. Since we assume $\delta = 0$ in this section we will sometimes simplify notation and write $T(n, \alpha, \kappa)$ (resp. $S(n, \alpha, \kappa)$) instead of $T(n, \alpha, \kappa, 0)$ (resp. $S(n, \alpha, \kappa, 0)$).

Algorithm 2 ε-differentially private framework for tunable deterministic approximation algorithms

```
Input: Input set $D$, accuracy parameter $\alpha \in (0,1)$, differential privacy parameter $\varepsilon$, approx. algorithm $\mathcal{A}_f$.
1: Let $x_A := \mathcal{A}_f(D, \rho, \tau, 0)$ where $\rho := \frac{\varepsilon\alpha}{36}$ and $\tau := \kappa$.
2: return $x_A + X$ where $X \sim \mathsf{C}\left(\frac{6(4\rho x_A + \Delta_f)}{\varepsilon}\right)$
```

Remark 3.7. *When $\mathcal{A}_f$ is a PTAS, by definition, the output of $\mathcal{A}_f$ is an $(\alpha, 0, 0)$-approximation of f running in time $T(n, \alpha, 0) = \mathrm{poly}(n, 1/\alpha)$. Applying Theorem 1.6 with for any $\alpha > 0$ we obtain a private $\left(\alpha, O\left(\frac{\Delta_f}{\varepsilon}\right), 9/10\right)$-approximation with polynomial running time $\mathrm{poly}(n, 1/\varepsilon, 1/\alpha)$.*

Theorem 1.6. *(ε-privacy) Suppose that $\mathcal{A}_f$ is a deterministic tunable approximation of $f : \mathcal{D} \to \mathbb{R}^+$. Then for all $\varepsilon > 0$, $\alpha \geq 0$ and $\kappa \geq 0$, there is an algorithm $\mathcal{A}'_f$ such that*

(1) (Privacy) $\mathcal{A}'_f$ is ε-differentially private.

(2) (Accuracy) For all $D \in \mathcal{D}$, we have that with probability $\geq 9/10$,

$$(1-\alpha')f(D) - \kappa' - \frac{7\Delta_f}{\varepsilon} \leq \mathcal{A}'_f(D) \leq (1+\alpha')f(D) + \kappa' + \frac{7\Delta_f}{\varepsilon}$$

where $\alpha' := \alpha C_1(\varepsilon + C_2\gamma)$, $\kappa' := \kappa C_3(\alpha + \frac{C_4}{\varepsilon})$ for some constants $C_1, C_2, C_3, C_4 > 0$ and $\Delta_f := \max_{D,D' \in \mathcal{D}, D \sim D'} \|f(D) - f(D')\|_1$.

(3) (Resource) $\mathcal{A}'_f$ uses $R\left(n, \frac{\varepsilon\alpha}{36}, \kappa\right)$ resource, where $R(\cdot,\cdot,\cdot)$ is the resource used by $\mathcal{A}_f$.

Proof. $\mathcal{A}'_f$ is defined in Algorithm 2 — we first run $\mathcal{A}_f(D, \rho, \kappa)$ where $\rho := \frac{\varepsilon\alpha}{36}$ and then we add noise proportional to the standard Cauchy distribution. Thus, the resource used will be $R(n, \rho, \kappa)$.

The privacy guarantee follows from Lemma 3.9, and the accuracy guarantee follows from Lemma 3.11. □

Lemma 3.8. *Suppose that $\mathcal{A}_f$ outputs a $(\rho, \tau, 0)$-approximation where $0 < \rho < 1/2$ of a function $f : \mathcal{D} \to \mathbb{R}^+$ with global sensitivity Δ_f. Then the function $S_f(D) = 4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f$ is a β-smooth upper bound for $\mathcal{A}_f$ where $\beta \geq 6\rho$.*

The proof remains the same as in Lemma 3.2. Applying Lemma 3.8 to the theorem calibrating noise to smooth bounds on the smooth sensitivity [NRS07] we show that the Algorithm 2 preserves privacy below.

Lemma 3.9. *Algorithm 2 is ε-differentially private.*

Proof. We first observe that by Lemma 3.8, $S_f(D) = 4\mathcal{A}_f(D) + 4\tau + \Delta_f$ is a β-smooth upper bound for $\mathcal{A}_f$. Recall that $\rho := \frac{\varepsilon\alpha}{36}$, thus we can apply Theorem 2.8 (with $\lambda = 2$) where $6\rho \leq \beta \leq \frac{\varepsilon}{6}$ and conclude that it is sufficient to add noise proportional to $\mathsf{C}\left(\frac{2(2+1)S_f(x)}{\varepsilon}\right) = \mathsf{C}\left(\frac{6(4\rho\mathcal{A}_f(D)+4\tau+\Delta_f)}{\varepsilon}\right)$ to preserve ε-privacy. □

Fact 3.10. *If $Y \sim \mathsf{C}(x; 0, b)$, then $\Pr[|Y| \geq \ell b] = 1 - \frac{2\tan^{-1}(\ell)}{\pi}$.*

Lemma 3.11. *For all $\gamma > 6.5$, with probability at least $9/10$,*

$$\left(1 - \rho\left(1 + \frac{48\gamma}{\varepsilon}\right)\right) f(D) - \frac{24(\rho+1)\gamma\tau}{\varepsilon} - \frac{6\Delta_f}{\varepsilon}\cdot\gamma \leq \mathcal{A}'_f(D)$$
$$\leq \left(1 + \rho\left(1 + \frac{48\gamma}{\varepsilon}\right)\right) f(D) + \frac{24(\rho+1)\gamma\tau}{\varepsilon} + \frac{6\Delta_f}{\varepsilon}\cdot\gamma$$

Proof. First, we invoke Fact 3.10 below,

$$\Pr\left[|X| \geq \frac{6(4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f)}{\varepsilon}\cdot\gamma\right] = 1 - \frac{2\tan^{-1}(\gamma)}{\pi} \leq \frac{1}{10}$$

where the final inequality comes from using the fact that $\gamma > 6.5$. In other words, with probability $\geq 9/10$,

$$|X| \leq \frac{6(4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma \tag{3}$$

$\mathcal{A}_f$ is a $(\rho, \tau, 0)$-approximation of f so for any $D \in \mathcal{D}$, we have that $\mathcal{A}_f(D) \leq (1+\rho)f(D) + \tau$. Since $0 < \rho < 1/2$ we have $(1+\rho)f(D) + \tau \leq 2f(D) + \tau$.

By plugging in the relation $\mathcal{A}_f(D) \leq 2f(D) + \tau$ into Eq. 3, we have that with probability at least 9/10,

$$|X| \leq \frac{6(8\rho f(D) + 4\rho\tau + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma$$

Thus with probability at least 9/10,

$$(1-\rho)f(D) - \tau - \frac{6(8\rho f(D) + 4\rho\tau + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma \leq \mathcal{A}'_f(D)$$
$$\leq (1+\rho)f(D) + \tau + \frac{6(8\rho f(D) + 4\rho\tau + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma$$

Rearranging the like terms together in the above expression completes the proof. □

Remark 3.12. *For simplicity, we have chosen to sample from the standard cauchy distribution $\lambda = 2$, more generally, if we sample noise with density $h(z) \propto \frac{1}{1+|z|^\lambda}$, where $\lambda = c$, then with probability $1-\delta$, $\gamma = \frac{1}{\delta^{1/c}}$ in Lemma 3.11.*

Application to the Knapsack Problem As a fun example we consider the knapsack problem. The knapsack problem is well known to be NP-Hard, but also admits an FPTAS. To define an instance of the knapsack problem we have a maximum weight capacity W for the knapsack and n items each with a value $v_{max} \geq v_i \geq 0$ and a weight $w_i \geq 0$. The goal is to find a subset $S \subseteq [n]$ of items to put in the knapsack maximizing the total value $v(S) = \sum_{i\in S} v_i$ subject to the constraint that the total weight $w(S) = \sum_{i\in S} w_i$ does not exceed our capacity i.e., $w(S) \leq W$.

For the purpose of this illustration let's fix the capacity W and weights $w_1, \ldots, w_n$ and let $f(v_1, \ldots, v_n)$ denote the value of the optimal knapsack solution given values $v_1, \ldots, v_n$. Let's say that two knapsack instances $(W, v_1, \ldots, v_n, w_1, \ldots, w_n)$ and $(W, v'_1, \ldots, v'_n, w_1, \ldots, w_n)$ are neighbors if $\sum_i \|v_i - v'_i| \leq 1$. Thus, we are viewing the exact value of each item as sensitive and the goal of differential privacy is to prevent an attacker from inferring these sensitive values exactly. Observe that the global sensitivity of f is upper bounded by $\Delta_f \leq \max_{v\sim v'} \max_{S\subseteq[n]} |v(S) - v'(S)| \leq 1$[9].

Since there is an FPTAS algorithm for Knapsack we can find a non-private approximation algorithm $\mathcal{A}_f(\vec{v}, \alpha, \kappa = 0)$ running in time $T(n, \alpha) = \text{poly}(n, 1/\alpha)$. If we apply Theorem 1.6 then for any target α' our ε-DP algorithm $\mathcal{A}'_f$ runs in time $\text{poly}(n, 1/\varepsilon, 1/\alpha)$ and solves Knapsack with additive error $\mathcal{O}(1/\varepsilon)$ and multiplicative error α' with probability at least 9/10. If we don't require pure DP then we can also apply Theorem 1.3 then for any target α' our algorithm $\mathcal{A}'_f$ runs in time $\text{poly}(n, 1/\varepsilon, 1/\alpha, \log(1/\delta))$ and solves Knapsack with probability at least $1 - \delta - \exp(-\gamma)$ with additive error at most $\mathcal{O}(\gamma/\varepsilon)$ and multiplicative error α'.

[9]We could also define neighboring knapsack instances such that we can completely replace the value of any item i.e., v and v' are neighbors if there exists some index $i \in [n]$ such that $v_i \neq v'_i$ and $v_j = v'_j$ for all $j \neq i$. However, in this case we can we would have large global sensitivity $\Delta_f = v_{max}$. Thus, we won't be able to design an accurate differentially private approximation even if we are willing to solve the NP-Hard knapsack problem exactly.

4 Private Sublinear-time Algorithms

We present a variety of DP results for sublinear-time graph algorithms by directly applying our DP black-box transformation (see Theorem 1.3 and Theorem 1.5). Let $\mathcal{G}_n$ denote the set of all n-node graphs. For a graph $G \in \mathcal{G}_n$ on m edges, we denote $\bar{d}$ as the average degree of G. Graphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$ are *node-neighboring*, denoted by $G_1 \sim_v G_2$, if there exists a vertex $v \in V$ such that $E_1(V \setminus \{v\}) = E_2(V \setminus \{v\})$. Graphs G_1 and G_2 are *edge-neighboring* i.e., $G_1 \sim_e G_2$ if there exists an edge e such that $E_1 \setminus \{e\} = E_2 \setminus \{e\}$.

We first define the sublinear models that we will be working with in the following problems. In the *adjacency list query* model the following queries can be answered in constant time — (1) degree queries: given $v \in V$, output $\deg(v)$. (2) neighbor queries: given $v \in V$, and $i \in [n]$, output the i-th neighbor of v or $\perp$ if $i > \deg(v)$. In the *adjacency matrix query* model the following queries can be answering in constant time— pair queries: given $u, v \in V$, output whether (u, v) is an edge in G or not. The *general query* model allows for degree queries, neighbor queries, and pair queries.

Typically the success probability for non-DP sublinear-time algorithms is stated in terms of constants (i.e., $\geq 2/3$). We implicitly apply the standard median trick to boost the probability of success to $1 - \delta$ through $\mathcal{O}\left(\log \frac{1}{\delta}\right)$ parallel iterations, and state the non-DP results with success probability $1 - \delta$ in the following exposition.

Estimating the number of Triangles. Given a graph G, we study the problem of estimating the number of triangles in the general query model. [ELRS17] gave an approximation algorithm for this problem with an expected query complexity of $\mathcal{O}\left(\left(\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t}\right) \operatorname{poly}(\log(n), 1/\alpha) \log(1/\delta)\right)$. In order to achieve a high probability bound on the query-complexity, we run $\Theta(\log(1/\delta))$ instances of the algorithm in parallel and return the output of the instance that terminates first.

Theorem 4.1. *[ELRS17] Let t be the number of triangles in graph G. For any $0 < \alpha < 1$, there is an algorithm that makes $\mathcal{O}\left(\left(\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t}\right) \operatorname{poly}(\log(n), 1/\alpha) \log(1/\delta)\right)$ queries and outputs an estimate $\hat{t}$ such that with probability at least $1 - \delta$,*

$$(1 - \alpha) \cdot t \leq \hat{t} \leq (1 + \alpha) \cdot t$$

Observe that the global sensitivity (under edge-DP) of the function that computes the number of triangles is $n - 2$, since each edge in an n node graph is incident to at most $n - 2$ triangles.

We first show that any ε-edge DP algorithm for estimating the number of triangles in an arbitrary graph must have an $\Omega(n)$ additive error. Consider graphs G_1 and G_2 such that $G_1 := K_{2,n-2}$ (the complete bipartite graph where the left partite set contains 2 vertices and the right partite set contains $n - 2$ vertices), and G_2 is $K_{2,n-2}$ with an edge between the two vertices in the left partite set. Clearly, $G_1 \sim_e G_2$, and G_1 has zero triangles, while G_2 has $n - 2$ triangles. A non-DP sublinear-time algorithm with multiplicative error may output 0 for G_1 and a value in the range $(1 \pm \alpha)(n - 2)$ for G_2, but any ε-DP algorithm must output an answer with an $\Omega(n)$ additive error. We also note that when the number of triangles in the graph is $\Omega(n)$, this additive error may be absorbed into the multiplicative error.

We apply our black-box transformation (Theorem 1.3 and Theorem 1.5) to the result of [ELRS17] to get the following ε-DP algorithm.

Corollary 4.2. *Let t denote the number of triangles in a graph. Then for any integer $K > 0$ and for all $c > 0$, $\varepsilon > 0$, there exists an algorithm $\mathcal{A}'$ such that*

(1) (Privacy) $\mathcal{A}'$ is ε-edge differentially private where ε is constant.

(2) (Accuracy) For all graphs G on n vertices, with probability $1 - \frac{1}{n^c} - \frac{1}{n^{c-3}(e^\varepsilon - 1)/K + 1}$,

$$(1-\alpha')t - \frac{2c}{\varepsilon}\cdot n\log(n) - \frac{1}{Kn^3} \le \mathcal{A}'(G) \le (1+\alpha')t + \frac{2c}{\varepsilon}\cdot n\log(n) + \frac{1}{Kn^3}$$

where $\alpha' = \alpha\left(1 + \frac{\varepsilon}{C'\log(n)}\right)$ for some constant $C' > 0$.

(3) (Query) $\mathcal{A}'$ makes $\mathcal{O}\left(\left(\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t}\right)\mathrm{poly}(\log(n), \frac{1}{\alpha\varepsilon})\right)$ queries.

Estimating the number of Connected Components. Given a graph G, we study the problem of estimating the number of connected components in a graph in the adjacency query list model. We recall the result by [CRT05], whose query complexity was later improved by [BKM14].

Theorem 4.3. *[CRT05, BKM14] Let C be the number of connected components in a graph with n vertices. Then for $\kappa > 0$, there is an algorithm that makes $\mathcal{O}\left(\frac{1}{\kappa^2}\log(\frac{1}{\kappa})\log(\frac{1}{\delta})\right)$ queries and with probability at least $1-\delta$,*

$$\mathsf{C} - \kappa n \le \tilde{\mathsf{C}} \le \mathsf{C} + \kappa n\ ,$$

where $\tilde{\mathsf{C}}$ denotes the output of the algorithm.

Observe that the global sensitivity of the function computing the number of connected components under edge-DP is 2, since the removal or addition of an edge can increase the number of connected components by at most 2. We apply our blackbox transformation to the result of [BKM14] to obtain an ε-DP approximation algorithm with the following guarantees.

Corollary 4.4. *Let the number of connected components of a graph G be denoted as $\mathsf{C}(G)$. Then for any integer $K > 0$ and for all $c > 0, \varepsilon > 0, \kappa > 0$, there is an algorithm $\mathcal{A}'$ such that*

(1) (Privacy) $\mathcal{A}'$ is ε-differentially private where ε is constant.

(2) (Accuracy) For all graphs G on n vertices, with probability $1 - \frac{1}{n^c} - \frac{1}{n^{c-1}(e^\varepsilon - 1)/K + 1}$,

$$\mathsf{C}(G) - \kappa n\left(\frac{8c}{\varepsilon} + \frac{1}{\log(n)}\right) - \frac{2c\log(n)}{\varepsilon} - \frac{1}{Kn} \le \mathcal{A}'(G)$$
$$\le \mathsf{C}(G) + \kappa n\left(\frac{8c}{\varepsilon} + \frac{1}{\log(n)}\right) + \frac{2c\log(n)}{\varepsilon} + \frac{1}{Kn}$$

(3) (Query) $\mathcal{A}'$ makes $\mathcal{O}\left(\frac{\log^3(n)}{\kappa^2}\log\left(\frac{\log(n)}{\kappa}\right)\right)$ queries.

We note that [Tet22] give a pure DP algorithm for this problem as well, however their accuracy guarantee holds with constant probability.

Estimating the weight of the MST. Given a (connected) graph $G = (V, E)$, with an associated weight function $wt : E(G) \to \{1, \ldots, w\}$, the problem is to estimate the weight of a minimum spanning tree in sublinear time. [CRT05] reduce the problem of computing the weight of the minimum spanning tree to counting the number of connected components in various subgraphs of G.

Theorem 4.5. *[CRT05] Let G be a graph on n vertices, with an associated weight function $wt : E(G) \to \{1, \ldots, w\}$, and $w < n/2$. Let M be the weight of the MST of G. Then for any $0 < \alpha < 1$, there is an algorithm that makes $\mathcal{O}\left(\bar{d}w\alpha^{-2} \log \frac{\bar{d}w}{\varepsilon} \log(1/\delta)\right)$ queries and outputs a value $\tilde{\mathsf{M}}$ such that with probability at least $1 - \delta$,*

$$|\mathsf{M} - \tilde{\mathsf{M}}| \leq \alpha \mathsf{M} .$$

where $\tilde{\mathsf{M}}$ denotes the output of the algorithm.

We first observe that the global sensitivity of the weight of the MST under edge-DP is w where w is the maximum weight of an edge, since the addition or deletion of an edge can change the total weight by at most w. By applying Theorem 1.3 and Theorem 1.5 to the result of [CRT05], we obtain the following DP result for integral weights.

Corollary 4.6. *Let $\mathsf{M}(G)$ be the weight of the MST of G. Then for all integers $K > 0$ and all $c > 0, \varepsilon > 0$, there exists an algorithm $\mathcal{A}'$ such that*

(1) (Privacy) $\mathcal{A}'$ is ε-edge differentially private where ε is constant.

(2) (Accuracy) For all graphs G on n vertices, with probability $1 - \frac{1}{n^c} - \frac{1}{n^{c-3(e^\varepsilon - 1)/K+1}}$,

$$(1 - \alpha')\mathsf{M}(G) - \frac{2cw\log(n)}{\varepsilon} - \frac{1}{Kn^3} \leq \mathcal{A}'(G) \leq (1 + \alpha')\mathsf{M}(G) + \frac{2cw\log(n)}{\varepsilon} + \frac{1}{Kn^3}$$

where $\alpha' = \alpha\left(1 + \frac{\varepsilon}{C' \log(n)}\right)$ for some constant $C' > 0$.

(3) (Query) $\mathcal{A}'$ makes $\mathcal{O}\left(\bar{d}w\frac{\log^3(n)}{\alpha^2\varepsilon^2} \log\left(\frac{\bar{d}w\log(n)}{\alpha\varepsilon}\right)\right)$ queries.

Estimating the average degree. Recall that $\bar{d}$ denotes the average degree of a graph. We study the problem of estimating the average degree of the graph in the adjacency query list model. We first recall the result of [GR04] below,

Theorem 4.7. *[GR04] For every $\alpha \in (0, 1)$, there is an algorithm that makes $\mathcal{O}\left(\frac{n}{\sqrt{m}} \operatorname{poly}\left(\frac{\log(n)}{\alpha}\right) \log(1/\delta)\right)$ queries and outputs $\tilde{d}$ such that with probability at least $1 - \delta$,*

$$|\bar{d} - \tilde{d}| \leq \alpha\bar{d} .$$

where $\tilde{d}$ denotes the output of the algorithm.

We will first present our result using our DP framework and then compare this result to existing DP results. In particular, by applying our blackbox transformation (Theorem 1.3 and Theorem 1.5) directly to the result of [GR04], and observing that the global sensitivity of the average degree under edge-DP is $2/n$ (since the addition or deletion of an edge affects two vertices in the sum), we obtain the following.

Corollary 4.8. *Let $\bar{d}$ denote the average degree of a graph. Then for any integer $K > 0$ and for all $c > 0$, $\varepsilon > 0$, there exists an algorithm $\mathcal{A}'$ such that*

(1) (Privacy) $\mathcal{A}'$ is ε-edge differentially private where ε is constant.

(2) (Accuracy) For all graphs G on n vertices, with probability $1 - \frac{1}{n^c} - \frac{1}{n^{c-1}(e^{\varepsilon}-1)/K+1}$,

$$(1-\alpha')\bar{d} - \frac{4c}{\varepsilon} \cdot \frac{\log(n)}{n} - \frac{1}{Kn} \leq \mathcal{A}'(G) \leq (1+\alpha')\bar{d} + \frac{4c}{\varepsilon} \cdot \frac{\log(n)}{n} + \frac{1}{Kn}$$

where $\alpha' = \alpha\left(1 + \frac{\varepsilon}{C' \log(n)}\right)$ for some constant $C' > 0$.

(3) (Query) $\mathcal{A}'$ makes $\mathcal{O}\left(\frac{n}{\sqrt{m}} \operatorname{poly}\left(\frac{\log^2(n)}{\varepsilon\alpha}\right) \log(n)\right)$ queries.

In recent work, [BGM22] adapted the algorithm by [GR04] to obtain an ε-DP $(\alpha, 0, \delta)$-approximation of $\bar{d}$ in $\mathcal{O}\left(\sqrt{n} \operatorname{poly}\left(\frac{\log(n)}{\alpha}\right) \operatorname{poly}\left(\frac{1}{\varepsilon}\right) \log(1/\delta)\right)$ for $\bar{d} \geq 1$ queries. First observe that, [BGM22] achieves a multiplicative approximation, whereas Corollary 4.8 has an additive error (along with the multiplicative error). Thus, it may seem that the algorithm given by [BGM22] is a better algorithm overall. But the algorithm in Corollary 4.8 has a few advantages. The first advantage is that unlike [BGM22], the algorithm works for all values $\bar{d} \geq 0$, whereas [BGM22] only works for the (albeit natural) assumption that $\bar{d} \geq 1$. Also for reasonable values of $\bar{d}$, i.e., for $\bar{d} \geq \Omega(\frac{\log n}{n})$, the additive error above is absorbed into the multiplicative error thus resulting in a pure DP algorithm with multiplicative error and no prior assumptions on the average degree $\bar{d}$. The query complexity achieved in Corollary 4.8 can also be much better than [BGM22]. For example, if $m = \Omega(n^2)$ then our algorithm runs in $\operatorname{poly}\left(\frac{\log(n)}{\alpha\varepsilon}\right)$ queries. In general if the graph has $m = \omega(n)$ edges then the query complexity of the original non-DP algorithm [GR04] is superior to [BGM22], and the query complexity of our algorithm is comparable to [GR04] — only incurs a multiplicative factor of $\operatorname{poly}\left(\frac{\log(n)}{\alpha\varepsilon}\right)$ overhead. Lastly, due to its black-box nature, the algorithm in Corollary 4.8 is much simpler in nature. The algorithm of [BGM22] is more complex due to its careful addition of noise in a way that results in (essentially) the same approximation guarantees as the non-DP algorithm.

We note that [GR04] avoided assuming any lowerbound on the average degree by first designing a sublinear-time algorithm that estimates the average degree by taking a degree lower bound ℓ as an input parameter and then removing this lower bound parameter via a geometric search. Implementing this type of geometric-search procedure in the DP setting would result in a privacy loss of $\log(n)$, and thus [BGM22] avoided this step by apriori assuming $\bar{d} \geq 1$. In recent work [Tet22] gives a general technique of making this procedure DP and thus achieves a pure DP algorithm for the average degree problem with (essentially) the same guarantees as the non-DP algorithm and no prior assumption on $\bar{d}$. However, their accuracy guarantees only hold with constant probability, whereas the accuracy guarantees achieved by applying our DP framework hold with high probability.

Estimating the maximum matching size. We study the problem of estimating the maximum matching size in a graph of maximum degree d in the adjacency query list model. [YYI12] gave an $(0, \kappa n)$-approximation for this problem with an expected query complexity of $d^{\mathcal{O}(1/\kappa^2)}$. In order to achieve a high probability bound on the query-complexity, we run $\Theta(\log(1/\delta))$ instances of the algorithm in parallel and return the output of the instance that terminates first.

Theorem 4.9. *[YYI12] For all $\kappa > 0$, there is a $(0, \kappa n, \delta)$-approximation algorithm for the maximum matching problem that uses $\mathcal{O}\left(d^{\mathcal{O}(1/\kappa^2)} \log(1/\delta)\right)$ queries with probability at least $1 - \delta$.*

We observe that the global sensitivity of the size of a maximum matching of a graph under node (and edge-DP) is at most one, since the node that is added/removed could have only contributed to at most one edge in the matching. By applying our black-box DP transformation to the results of [YYI12], we get the following guarantees.

Corollary 4.10. *Let the size of the maximum matching of a graph G be denoted as $\mathsf{MM}(G)$. Then for any integer $K > 0$ and for all $c > 0$, $\varepsilon > 0$, there is an algorithm $\mathcal{A}'$ such that*

(1) (Privacy) $\mathcal{A}'$ is ε-node (and edge) differentially private where ε is constant.

(2) (Accuracy) For all graphs G on n vertices, with probability $1 - (\frac{1}{n^c} + \frac{1}{n^{c-1}(e^\varepsilon - 1)/K+1} + \frac{1}{10})$,

$$\mathsf{MM}(G) - \kappa n\left(\frac{8\log 10}{\varepsilon} + 1\right) - \frac{2\log 10}{\varepsilon} - \frac{1}{Kn} \le \mathcal{A}'(G)$$
$$\le \mathsf{MM}(G) + \kappa n\left(\frac{8\log 10}{\varepsilon} + 1\right) + \frac{2\log 10}{\varepsilon} + \frac{1}{Kn}$$

(3) (Query) $\mathcal{A}'$ makes $\mathcal{O}\left(d^{\mathcal{O}(1/\kappa^2)} \log(n)\right)$ queries.

In recent work, [BGM22] obtains an approximation for the maximum matching size with a multiplicative factor of 2 and additive factor of κn in time $\tilde{O}\left((\bar{d}+1)/\kappa^2\right)$. They left the problem of designing a DP algorithm that achieves a $(0, \kappa n, \delta)$-approximation as an open problem. Corollary 4.10 partially resolves this open problem. We note that [Tet22] fully resolves this problem by giving a pure DP algorithm whose accuracy guarantee holds with high probability.

One disadvantage of our framework is that if the resource (e.g., time, query or space) complexity of the non-DP approximation algorithm depends on the approximation parameter in a non-polynomial way (e.g., exponentially), then the resulting DP algorithm that achieves an accuracy of similar magnitude with high probability may be inefficient. This is the case for the algorithm of [YYI12] — the non-DP approximation algorithm has a query complexity of $\mathcal{O}\left(d^{\mathcal{O}(1/\kappa^2)} \log(1/\delta)\right)$ (which is exponential in the approximation parameter κ). If we wanted to guarantee that the additive error due to sampling from the Laplace distribution is small *with high probability* (e.g., with probability $\geq 1 - \frac{1}{n}$), then we would need to set $\gamma = \log(n)$ in Theorem 1.3. Without tuning κ, the resulting accuracy would have an error of $\mathcal{O}(n \log n)$, which would render the output of the algorithm meaningless (since the matching size $\leq n$). On the otherhand, if we tune κ to $\tau := \kappa/\log(n)$ in Algorithm 1, the approximation achieved would be meaningful, i.e., the additive error would be small enough, but the query complexity of the resulting DP algorithm would become $\mathcal{O}\left(d^{\mathcal{O}(\log^2(n)/\kappa^2)} \log(1/\delta)\right)$. Thus in order to achieve the guarantees in Corollary 4.10, we need to relax the guarantee that the additive error due to sampling from the Laplace distribution is small *with high probability*, and ensure this is true *with constant probability* instead (for e.g., with probability $\geq 9/10$).

Estimating the distance to bipartiteness. We study the problem of estimating the distance to bipartiteness in the adjacency list query model. The distance to the graph property of bipartiteness is said to be ψ if ψn^2 edges need to be added or removed from the graph on n vertices in order to obtain a bipartite graph. We first recall the result by [GMRS22] as follows.

Theorem 4.11. *[GMRS22] For every $\kappa > 0$, there exists an algorithm that given input graph G inspects a random subgraph of G on $\tilde{\mathcal{O}}\left((1/\kappa^3)\log(1/\delta)\right)$ vertices and estimates the distance from G to bipartiteness to within an additive error of κn^2 with probability $\geq 1-\delta$.*

We observe that the global sensitivity (under edge-DP) of the distance to bipartiteness function is $1/n^2$ (or 1 without the normalization factor), since by adding an edge one can introduce an odd cycle, making a previously bipartite graph into a non-bipartite graph, and conversely, by removing an edge, one can remove the existence of an odd cycle thus making a previously non-bipartite graph into a bipartite graph. By applying Theorem 1.3 to the results of [GMRS22], we get the following guarantees.

Corollary 4.12. *Let the distance from a graph G to bipartiteness be denoted as $\mathsf{d_B}(G)$. For any integer $K > 0$ and for all $c > 0, \varepsilon > 0$, there is an algorithm $\mathcal{A}'$ such that*

(1) (Privacy) $\mathcal{A}'$ is ε-differentially private where ε is constant.

(2) (Accuracy) For all graphs G on n vertices, with probability $1-\frac{1}{n^c}-\frac{1}{n^{c-2}(e^\varepsilon-1)/K+1}$,

$$\mathsf{d_B}(G) - \kappa n^2\left(\frac{8c}{\varepsilon}+\frac{1}{\log(n)}\right) - \frac{2c\log(n)}{\varepsilon} - \frac{1}{Kn^2} \leq \mathcal{A}'(G)$$
$$\leq \mathsf{d_B}(G) + \kappa n^2\left(\frac{8c}{\varepsilon}+\frac{1}{\log(n)}\right) + \frac{2c\log(n)}{\varepsilon} + \frac{1}{Kn^2}$$

(3) (Query) $\mathcal{A}'$ inspects $\tilde{\mathcal{O}}\left(\frac{\log^4(n)}{\kappa^3}\right)$ vertices.

5 Private Streaming Algorithms

In this section, we apply our general framework to achieve private algorithms on data streams, where updates to an underlying dataset arrive sequentially and the goal is to output or approximate some predetermined function using space sublinear in the size of the input. In Section 5.1, we consider applications to dynamic/turnstile streams, where updates of the stream can be both insertions and deletions. In Section 5.2, we consider applications to the sliding window model, where all updates of the stream are insertions, but elements are also implicitly deleted by the sliding window once they are past their expiration, i.e., when W subsequent updates arrive in the stream, for a fixed window parameter $W > 0$.

5.1 Dynamic/Turnstile Streams

In this section, we first apply our general framework to problems in the dynamic/turnstile streaming model. We remark that since this model generalizes the insertion-only model, our results also automatically apply to the insertion-only model. We say that streams $\mathfrak{S}$ and $\mathfrak{S}'$ are *neighboring* if there exists a single update $i \in [m]$ such that $u_i \neq u_i'$, where $u_1, \ldots, u_m$ are the updates of $\mathfrak{S}$ and $u_1', \ldots, u_m'$ are the updates of $\mathfrak{S}'$. The DP streaming algorithms we present here are in the one-shot model, i.e., the algorithm outputs at the end of the stream.

Weighted minimum spanning tree. We first study the problem of estimating the weighted minimum spanning tree of a graph implicitly defined in a data stream. Given a set of vertices V, each update in the stream has the form $[-M, M] \times (u, v)$ for some $u, v \in V$, which changes the weight of edge (u, v) by some amount between $-M$ and M. We recall the following turnstile streaming algorithm for estimating the weighted minimum spanning tree by [AGM12].

Theorem 5.1. *[AGM12] For any $\alpha \in (0, 1]$, there exists a single-pass algorithm that outputs a $(\alpha, 0)$-approximation to the weight of the minimum spanning tree of a dynamic graph with probability at least $\frac{2}{3}$, using $\mathcal{O}\left(\frac{1}{\alpha} n \log^3 n\right)$ bits of space.*

Lemma 5.2. *Suppose a graph is implicitly defined by a dynamic stream that changes the weight of each edge between $[-M, M]$. Then the global sensitivity of the weighted minimum spanning tree is at most $2M$.*

Proof. Let G, G' be two connected graphs so that some edge (u, v) has weight W in G and W' in G' and suppose without loss of generality that $W < W'$. Let T be the minimum weighted spanning tree of G and T' be the minimum weighted spanning tree of G'. Note that since $W' > W$, then T is also a valid tree in G, so that the minimum weighted spanning tree of G' has at most the weight of T in G'. Since the weight of (u, v) changed by at most M, then the weight of T in G' is at most M more than the weight of T in G. Hence, the minimum weighted spanning tree of G' has weight at most M more than that of the minimum weighted spanning tree of G. Moreover, the weight of each edge in G' is at least the weight of each edge in G, so the minimum weighted spanning tree of G' has weight at least that of the minimum weighted spanning tree of G. Thus the minimum weighted spanning trees of G and G' differ by at most M. By the triangle inequality, it follows that the global sensitivity of the weighted minimum spanning tree of a dynamic graph is at most $2M$. $\square$

Therefore, by first applying the median trick (see Remark 1.4) to Theorem 5.1 to boost the success probability and then applying our main framework, and noting the global sensitivity bounds in Lemma 5.2, we have the following guarantee:

Corollary 5.3. *For any integer $K > 0$ and for all $\alpha \in (0, 1)$, $c > 0$, $\varepsilon > 0$, and $\rho := \mathcal{O}\left(\frac{\alpha\eta}{\log m}\right)$ on a turnstile stream $\mathfrak{S}$ of length $m = \text{poly}(n)$, where $\eta = \min(\varepsilon, 1)$, there exists a turnstile streaming algorithm $\mathcal{A}$ algorithm that:*

(1) (Privacy) The algorithm $\mathcal{A}$ is ε-differentially private where ε is constant.

(2) (Accuracy) The algorithm $\mathcal{A}$ outputs $\widehat{W}(\mathfrak{S})$ such that with probability at least $1 - \frac{1}{m^c} - \frac{1}{m^{c-1}(e^\varepsilon - 1)/(KMm)+1}$,

$$(1-\alpha)W(\mathfrak{S}) - \frac{12cM \log m}{\varepsilon} - \frac{1}{KMm} \le \widehat{W}(\mathfrak{S}) \le (1+\alpha)W(\mathfrak{S}) + \frac{12cM \log m}{\varepsilon} + \frac{1}{KMm},$$

where $W(\mathfrak{S})$ denotes the minimum weighted spanning tree in the graph induced by the stream $\mathfrak{S}$.

(3) (Space) The algorithm $\mathcal{A}$ uses space $\mathcal{O}\left(\frac{1}{\alpha\eta} n \log^5 n\right)$ bits.

L_p norm and F_p moment estimation. We next study the related problems of L_p norm and F_p moment estimation, where a frequency vector $x \in \mathbb{R}^n$ is implicitly defined by updates of the stream and the goal is to approximate the the F_p moment of x, i.e., $x_1^p + \ldots + x_n^p$, or the L_p norm of x, i.e., $(x_1^p + \ldots + x_n^p)^{1/p}$. Each update of the stream has the form $\{-1, +1\} \times [n]$, indicating whether the update implicitly decrements or increments the corresponding coordinate of the frequency vector x. Finally, the L_0 norm of x is defined as $\|x\|_0 = |\{i \in [n] \mid x_i \neq 0\}|$, or the number of nonzero coordinates of x. We assume the data stream has length m.

We first upper bound the global sensitivity of the L_p norm for $p \geq 1$ and the F_p moment for $p \in [0, 1]$.

Lemma 5.4. *For $p \geq 1$, the global sensitivity of the L_p norm is at most 2. For $p \in [0, 1]$, the global sensitivity of the F_p moment is at most 2.*

Proof. Let $x, x' \in \mathbb{R}^n$ be two vectors defined by neighboring streams, so that $\|x - x'\|_1 \leq 2$. Then for $p \geq 1$, we have $\|x - x'\|_p \leq \|x - x'\|_1 \leq 2$. For $p \in (0, 1]$, it suffices to note that $|y^p - (y-1)^p| \leq 1$ and at most two coordinates differ between x and x', each by at most one. Finally, for $p = 0$, note that since at most two coordinates differ between x and x', then $|x - x'|_0 \leq 2$. $\square$

For $p > 2$, we use the following turnstile streaming algorithm of [GW18]:

Theorem 5.5. *[GW18] For $p > 2$, and accuracy parameter $\alpha > 0$ there is a turnstile streaming algorithm for $(\alpha, 0)$-approximation of L_p with probability at least $1 - \delta$ that uses $\mathcal{O}\left(\frac{1}{\alpha^2} n^{1-\frac{2}{p}} \log \frac{1}{\delta} \log n + \frac{1}{\alpha^{4/p}} n^{1-\frac{2}{p}} \log^{2/p} \frac{1}{\delta} \log^2 n\right)$ bits of space.*

Thus by applying our main framework from Theorem 1.3 and Theorem 1.5 to Theorem 5.5 and noting the global sensitivity bounds in Lemma 5.4, we have the following guarantee:

Corollary 5.6. *For any integer $K > 0$, and for all $p > 2$, $\alpha > 0$, $c > 0$, $\varepsilon > 0$, and $\rho := \mathcal{O}\left(\frac{\alpha\eta}{\log m}\right)$ on a stream $\mathfrak{S}$ of length $m = \text{poly}(n)$, where $\eta = \min(\varepsilon, 1)$, there exists a turnstile streaming algorithm $\mathcal{A}$ algorithm that:*

(1) (Privacy) The algorithm $\mathcal{A}$ is ε-differentially private where ε is constant.

(2) (Accuracy) The algorithm $\mathcal{A}$ outputs $\widehat{L_p}(\mathfrak{S})$ such that with probability at least $1 - \frac{1}{m^c} - \frac{1}{\text{poly}(m)(e^\varepsilon - 1)/K + 1}$,

$$(1-\alpha)L_p(\mathfrak{S}) - \frac{12c\log m}{\varepsilon} - \frac{1}{K\,\text{poly}(m)} \leq \widehat{L_p}(\mathfrak{S}) \leq (1+\alpha)L_p(\mathfrak{S}) + \frac{12c\log m}{\varepsilon} + \frac{1}{K\,\text{poly}(m)},$$

where $L_p(\mathfrak{S})$ denotes the L_p norm of the frequency vector induced by the stream $\mathfrak{S}$.

(3) (Space) The algorithm $\mathcal{A}$ uses space $\mathcal{O}\left(\frac{p}{\alpha^2\eta^2} n^{1-2/p}\right) \cdot \text{poly}\left(\log n, \log \frac{1}{\alpha\eta}\right)$ bits.

When $p = 2$, we use the following well-known AMS algorithm [AMS99]:

Theorem 5.7. *[AMS99] For an accuracy parameter $\alpha > 0$ there is a turnstile streaming algorithm for $(\alpha, 0)$-approximation of L_2 with probability at least $1-\delta$ that uses $\mathcal{O}\left(\frac{1}{\alpha^2} \log m \log \frac{1}{\delta}\right)$ bits of space, for a stream of length m.*

We note that [Tet22] give an ε-DP algorithm using the AMS algorithm as well, but the accuracy guarantee only holds with constant probability. By applying our main framework from Theorem 1.3 and Theorem 1.5 to the AMS algorithm in Theorem 5.7 and noting the global sensitivity bounds in Lemma 5.4, we have the following guarantee:

Corollary 5.8. *For any integer $K > 0$ and for any $\alpha > 0$, $c > 0$, $\varepsilon > 0$, and $\rho := \mathcal{O}\left(\frac{\alpha\eta}{\log m}\right)$ on a stream $\mathfrak{S}$ of length $m = \mathrm{poly}(n)$, where $\eta = \min(\varepsilon, 1)$, there exists a turnstile streaming algorithm $\mathcal{A}$ algorithm that:*

(1) (Privacy) The algorithm $\mathcal{A}$ is ε-differentially private where ε is constant.

(2) (Accuracy) The algorithm $\mathcal{A}$ outputs $\widehat{L_2}(\mathfrak{S})$ such that with probability at least $1 - \frac{1}{m^c} - \frac{1}{\mathrm{poly}(m)(e^\varepsilon - 1)/K + 1}$,

$$(1-\alpha)L_2(\mathfrak{S}) - \frac{12c\log m}{\varepsilon} - \frac{1}{K\,\mathrm{poly}(m)} \le \widehat{L_2}(\mathfrak{S}) \le (1+\alpha)L_2(\mathfrak{S}) + \frac{12c\log m}{\varepsilon} + \frac{1}{K\,\mathrm{poly}(m)},$$

where $L_2(\mathfrak{S})$ is the L_2-norm of the stream $\mathfrak{S}$.

(3) (Space) The algorithm $\mathcal{A}$ uses space $\mathcal{O}\left(\frac{1}{\alpha^2\eta^2}\log^4 n\right)$ bits.

For $p \in (0, 2)$, we use the following turnstile streaming algorithm of [KNPW11]:

Theorem 5.9. *[KNPW11] For $p \in (0, 2)$, and accuracy parameter $\alpha > 0$ there is a turnstile streaming algorithm for $(\alpha, 0)$-approximation of L_p with probability at least $\frac{2}{3}$ that uses $\mathcal{O}\left(\frac{1}{\alpha^2}\log m + \log\log n\right)$ bits of space.*

Hence by applying the median trick to Theorem 5.9 to boost the success probability, and then applying our main framework from Theorem 1.3, and finally noting the global sensitivity bounds in Lemma 5.4, we have the following guarantee:

Corollary 5.10. *For any integer $K > 0$, $p \in (0, 2)$, and for any $\alpha > 0$, $c > 0$, $\varepsilon > 0$, and $\rho := \mathcal{O}\left(\frac{\alpha\eta}{\log m}\right)$ on a stream $\mathfrak{S}$ of length $m = \mathrm{poly}(n)$, where $\eta = \min(\varepsilon, 1)$, there exists a turnstile streaming algorithm $\mathcal{A}$ algorithm that:*

(1) (Privacy) The algorithm $\mathcal{A}$ is ε-differentially private where ε is constant.

(2) (Accuracy) The algorithm $\mathcal{A}$ outputs $\widehat{X}$ such that with probability at least $1 - \frac{1}{m^c} - \frac{1}{\mathrm{poly}(m)(e^\varepsilon - 1)/K + 1}$,

$$(1-\alpha)X - \frac{12c\log m}{\varepsilon} - \frac{1}{K\,\mathrm{poly}(m)} \le \widehat{X} \le (1+\alpha)X + \frac{12c\log m}{\varepsilon} + \frac{1}{K\,\mathrm{poly}(m)},$$

where X is the L_p norm of the stream $\mathfrak{S}$ for $p \in [1, 2)$ and X is the F_p moment of the stream $\mathfrak{S}$ for $p \in (0, 1)$.

(3) (Space) The algorithm $\mathcal{A}$ uses space $\mathcal{O}\left(\frac{1}{\alpha^2\eta^2}\log^4 n\right)$ bits.

For $p = 0$, we use the following turnstile streaming algorithm of [KNW10]:

Theorem 5.11. *[KNW10] For any accuracy parameter $\alpha > 0$ there is a turnstile streaming algorithm for $(\alpha, 0)$-approximation of L_0 with probability at least $\frac{2}{3}$ that uses $\mathcal{O}\left(\frac{1}{\alpha^2}\log n \log\frac{1}{\alpha} + \log\log m\right)$ bits of space.*

Then by applying the median trick to Theorem 5.9 to boost the probability of success to $1-\delta$, and then applying our main framework from Theorem 1.3, and finally noting the global sensitivity bounds in Lemma 5.4, we have the following guarantee:

Corollary 5.12. *For any integer $K > 0$, and for any $\alpha > 0$, $c > 0$, $\varepsilon > 0$, and $\rho := \mathcal{O}\left(\frac{\alpha\eta}{\log m}\right)$ on a stream $\mathfrak{S}$ of length $m = \text{poly}(n)$, where $\eta = \min(\varepsilon, 1)$, there exists a turnstile streaming algorithm $\mathcal{A}$ algorithm that:*

(1) (Privacy) The algorithm $\mathcal{A}$ is ε-differentially private where ε is constant.

(2) (Accuracy) The algorithm $\mathcal{A}$ outputs $\widehat{L_0}(\mathfrak{S})$ such that with probability at least $1 - \frac{1}{m^c} - \frac{1}{\text{poly}(m)(e^\varepsilon-1)/K+1}$,

$$(1-\alpha)L_0(\mathfrak{S}) - \frac{12c\log m}{\varepsilon} - \frac{1}{K\,\text{poly}(m)} \le \widehat{L_0}(\mathfrak{S}) \le (1+\alpha)L_0(\mathfrak{S}) + \frac{12c\log m}{\varepsilon} + \frac{1}{K\,\text{poly}(m)},$$

where $L_0(\mathfrak{S})$ is the number of distinct elements in the stream $\mathfrak{S}$.

(3) (Space) The algorithm $\mathcal{A}$ uses space $\tilde{\mathcal{O}}\left(\frac{1}{\alpha^2\eta^2}\log^4 n\right)$ bits.

5.2 Sliding Windows

In this section, we introduce the sliding window model and apply our DP framework to ensure privacy for many sliding window algorithms.

Definition 5.13 (Sliding window model)**.** *In the sliding window model, there exists a data stream of length m and a window parameter $W > 0$. The underlying dataset is then implicitly defined by only the W most recent updates in the stream.*

The sliding window model is a generalization of the insertion-only streaming model that captures the prioritization of recent data over outdated or expired data and thus there are a number of time sensitive applications applications [BBD+02, DM07, MM12, PGD15, ELVZ17], such as network monitoring [CM05, CG08] or event detection on social media [OMM+14], for which the sliding window model has better performance than the insertion-only streaming model. The sliding window model is especially appropriate for time-sensitive applications such as network monitoring [CM05, CG08] and has been subsequently studied in a number of additional settings [DM07, BOZ12, ELVZ17, BGL+18, BDM+20, BWZ21, EMMZ22, JWZ22].

For the purposes of differential privacy, we permit two neighboring streams to differ by a single update over the entire data stream. Observe that this notion includes the setting where two neighboring streams differ by a single update in the dataset induced by the sliding window, i.e., the last W updates of the data stream.

[BO10] presented the smooth histogram framework [BO10], which converts an existing insertion-only streaming algorithm into a sliding window algorithm. We remark that unfortunately, due to standard nomenclature in previously non-intersecting literature, the notion of smooth histogram does not use the same notion of smooth as smooth sensitivity. Hence, we first define the following notion of a smooth function:

Definition 5.14. *A function $f \geq 1$ is (ρ, ξ)-smooth if it has the following properties:*

Monotonicity *$f(A) \geq f(B)$ for $B \subseteq A$ (B is a suffix of A)*

Polynomial boundedness *There exists $c > 0$ such that $f(A) \leq n^c$.*

Smoothness *For any $\rho \in (0,1)$, there exists $\xi \in (0, \rho]$ so that if $B \subseteq A$ and $(1-\xi)f(A) \leq f(B)$, then $(1-\rho)f(A \cup C) \leq f(B \cup C)$ for any adjacent C.*

The smooth histogram framework has the following guarantees:

Theorem 5.15 (Smooth Histogram Framework, [BO10]). *Let $f : \mathcal{U}^m \to \mathbb{R}$ be a $(\rho, \beta(\rho))$-smooth function for any $\rho \in (0,1)$ on a stream $\mathfrak{S}$ of length $m = \mathrm{poly}(n)$, and suppose there exists an insertion-only streaming algorithm $\mathcal{A}$ that outputs a $(\alpha, 0)$-approximation of f using space $\mathcal{S}(\alpha, \delta, m, n)$ and update time $\mathcal{T}(\alpha, \delta, m, n)$, for any approximation parameter $\alpha \in (0,1)$ and failure probability δ. Then for any constant $c > 0$ and window parameter $W > 0$, there exists a sliding window algorithm $\mathcal{A}'$ such that for the dataset $\mathcal{W}$ induced by the last $\min(m, W)$ updates of the stream:*

(1) (Accuracy) The algorithm $\mathcal{A}'$ outputs $\widehat{f}(\mathcal{W})$ such that with probability at least $1 - \frac{1}{m^c}$,

$$(1-\alpha)f(\mathcal{W}) \leq \widehat{f}(\mathcal{W}) \leq (1+\alpha)f(\mathcal{W}).$$

(2) (Time/Space) The algorithm $\mathcal{A}'$ uses space $\mathcal{O}\left(\frac{1}{\beta(\alpha)}(\mathcal{S}(\beta(\alpha), \frac{\delta}{\mathrm{poly}(m,n)}, m, n) + \log m) \log m\right)$ and update time $\mathcal{O}\left(\frac{1}{\beta(\alpha)}(\mathcal{T}(\beta(\alpha), \frac{\delta}{\mathrm{poly}(m,n)}, m, n)) \log m\right)$.

The main takeaway for the smooth histogram framework is that it essentially provides a means to achieve a $(\rho, 0)$-approximation algorithm in the sliding window model for a smooth function for which there already exists a $(\rho, 0)$-approximation algorithm in the insertion-only streaming model. For many additional problems, either there are no known $(\rho, 0)$-approximation algorithms for all $\rho > 0$, e.g., [ELVZ17] or the output is not a scalar quantity, e.g., [BOZ12, BDM+20, BWZ21, EMMZ22, JWZ22].

By Theorem 5.15 and Theorem 1.3, we have the following guarantee:

Theorem 5.16 (DP Smooth Histogram Framework). *Let $\alpha > 0, c > 0, \varepsilon > 0$ and a $(\rho, \beta(\rho))$-smooth function $f : \mathcal{U}^m \to \mathbb{R}$ for $\rho := \mathcal{O}\left(\frac{\alpha\eta}{\log m}\right)$ on a stream $\mathfrak{S}$ of length $m = \mathrm{poly}(n)$, where $\eta = \min(\varepsilon, 1)$ and suppose there exists an insertion-only streaming algorithm $\mathcal{A}$ that outputs a $(\alpha, 0)$-approximation of f using space $\mathcal{S}(\alpha, \delta, m, n)$ and update time $\mathcal{T}(\alpha, \delta, m, n)$, and failure probability δ. Then there exists a sliding window algorithm $\mathcal{A}'$ such that for the dataset $\mathcal{W}$ induced by the last $\min(m, W)$ updates of the stream:*

(1) (Privacy) The algorithm $\mathcal{A}'$ is (ε, δ)-differentially private where ε is constant and $\delta = \frac{1}{m^c}$.

(2) (Accuracy) The algorithm $\mathcal{A}'$ outputs $\widehat{f}(\mathcal{W})$ such that with probability at least $1 - \frac{1}{m^c}$,

$$(1-\alpha)f(\mathcal{W}) - \frac{6c\Delta_f \log m}{\varepsilon} \leq \widehat{f}(\mathcal{W}) \leq (1+\alpha)f(\mathcal{W}) + \frac{6c\Delta_f \log m}{\varepsilon}.$$

(3) (Time/Space) The algorithm $\mathcal{A}'$ uses space $\mathcal{O}\left(\frac{1}{\beta(\rho)}(\mathcal{S}(\beta(\rho), \frac{\delta}{\text{poly}(m,n)}, m, n) + \log m) \log m\right)$ and update time $\mathcal{O}\left(\frac{1}{\beta(\rho)}(\mathcal{T}(\beta(\rho), \frac{\delta}{\text{poly}(m,n)}, m, n)) \log m\right)$.

Observe that if we tried to apply the non-private smooth histogram framework to a DP insertion-only streaming algorithm, this might preserve privacy (by post-processing), but may significantly increase the error (in terms of accuracy).

Length of the longest increasing sub-sequence. The longest increasing subsequence problem LIS is to find an increasing subsequence of maximum length of a sequence whose elements are from the universe $[n]$ and sequentially defined by the stream. The *LIS-length* problem is to output the length of such a subsequence. [BO07] showed that the *LIS-length* function is (ρ, ρ)-smooth.

Theorem 5.17. *[BO07] For any $\rho \in (0,1)$, LIS-length is a (ρ, ρ)-smooth function.*

We now recall the following insertion-only streaming algorithm for estimating the length of the longest increasing subsequence

Theorem 5.18. *[SW07] Let $\mathfrak{S}$ be a stream $u_1, u_2, \ldots$ such that $u_i \in [n]$ and let k be an upper bound on the LIS-length in the stream, then there exists a 1-pass streaming algorithm that computes LIS-length with probability at least $\frac{2}{3}$, using space $\mathcal{O}\left(k^2 \log \frac{n}{k}\right)$.*

We can therefore apply Theorem 5.16 and Theorem 1.5 to the algorithm of [SW07] to obtain Corollary 5.20. We note that while the longest increasing subsequence may have large global sensitivity, the length of the longest increasing subsequence only has global sensitivity 1, i.e., $\Delta_{LIS\text{-}length} = 1$.

Lemma 5.19. *For the length of the longest increasing subsequence, the global sensitivity is at most 1.*

Proof. Let S, S' be two sequences so that S' is formed by deleting a term of S. Since any subsequence of S' is a subsequence of S and any subsequence of S can be transformed into a subsequence of S' by deleting at most one term, then the lengths of the longest subsequences in S and S' differ by at most 1. Hence by the triangle inequality, the global sensitivity of the length of the longest increasing subsequence is at most 2. ☐

Corollary 5.20. *Let $\mathfrak{S}$ be a stream $u_1, u_2, \ldots$ such that $u_i \in [n]$ of length $m = \text{poly}(n)$ and let k be an upper bound on the LIS-length in the stream. For some $K > 0$, and for any $\alpha > 0$, $c > 0$, $\varepsilon > 0$, and $\rho := \mathcal{O}\left(\frac{\alpha\eta}{\log m}\right)$ on a stream of length m, where $\eta = \min(\varepsilon, 1)$, there exists a sliding window algorithm $\mathcal{A}$ such that:*

(1) (Privacy) The algorithm $\mathcal{A}$ is ε-differentially private where ε is constant.

(2) (Accuracy) The algorithm $\mathcal{A}$ outputs $\widehat{LIS}(\mathcal{W})$ such that with probability at least $1 - \frac{1}{m^c} - \frac{1}{m^c(e^\varepsilon - 1)/(Kk)+1}$,

$$(1-\alpha)LIS(\mathcal{W}) - \frac{12c \log m}{\varepsilon} - \frac{1}{Kk} \le \widehat{LIS}(\mathcal{W}) \le (1+\alpha)LIS(\mathcal{W}) + \frac{12c \log m}{\varepsilon} + \frac{1}{Kk},$$

where $LIS(\mathcal{W})$ denotes the length of the longest increasing subsequence of the last $\min(m, W)$ updates of $\mathfrak{S}$, for a window parameter $W > 0$.

(3) (Time/Space) The algorithm $\mathcal{A}$ uses an $\mathcal{O}\left(\frac{k^2}{\alpha\eta} \log^4 n\right)$ space.

Distinct elements. We first give a simple proof to show that the function computing the number of distinct elements is (ρ, ρ)-smooth below.

Lemma 5.21. *For any $\rho \in (0,1)$, the number of distinct elements is (ρ, ρ)-smooth.*

Proof. Let f be the number of distinct elements on an input stream. Given a stream A, it is easy to see that f preserves the following properties of smoothness: (1) $f(A) \geq 0$, (2) $f(A) \geq f(B)$ where $B \subseteq A$, (3) $f(A) \leq \text{poly}(n)$.

It remains to show that if $B \subseteq A$ and $(1-\rho)f(A) \leq f(B)$ then $(1-\rho)f(A \cup C) \leq f(B \cup C)$ for any adjacent C. Note that $(1-\rho)f(A \cup C) \leq (1-\rho)(f(A)+f(C)) \leq f(B)+f(C) \leq f(B \cup C)$, where the first step is true because A and C are adjacent streams and the number of distinct elements in their union is therefore at most the sum of distinct elements in both. $\square$

We apply our general framework to the following insertion-only streaming algorithm given by [Bla20]:

Theorem 5.22. *[Bla20] Given an accuracy parameter $\alpha > 0$ and a failure probability $\delta \in (0,1)$, there exists a streaming algorithm that outputs a $(\alpha, 0)$-approximation to the number of distinct elements with probability $1-\delta$ using $\mathcal{O}\left(\frac{1}{\alpha^2}\log\frac{1}{\delta} + \log n\right)$ bits.*

Corollary 5.23. *For any integer $K > 0$ and for any $\alpha > 0$, $c > 0$, $\varepsilon > 0$, and $\rho := \mathcal{O}\left(\frac{\alpha\eta}{\log m}\right)$ on a stream $\mathfrak{S}$ of length $m = \text{poly}(n)$, where $\eta = \min(\varepsilon, 1)$, there exists a sliding window algorithm $\mathcal{A}$ such that for the dataset W induced by the stream:*

(1) (Privacy) The algorithm $\mathcal{A}$ is (ε, δ)-differentially private where ε is constant and $\delta = \frac{1}{m^c}$.

(2) (Accuracy) The algorithm $\mathcal{A}$ outputs $\widehat{\mathsf{dist}}(\mathcal{W})$ such that with probability at least $1 - \frac{1}{m^c} - \frac{1}{m^{c-1}(e^\varepsilon - 1)/K+1}$,

$$(1-\alpha)\mathsf{dist}(\mathcal{W}) - \frac{12c\log m}{\varepsilon} - \frac{1}{Km} \leq \widehat{\mathsf{dist}}(\mathcal{W}) \leq (1+\alpha)\mathsf{dist}(\mathcal{W}) + \frac{12c\log m}{\varepsilon} + \frac{1}{Km},$$

where $\mathsf{dist}(\mathcal{W})$ denotes the number of distinct elements in the last $\min(m, W)$ updates of the stream $\mathfrak{S}$, for a window parameter $W > 0$.

(3) (Time/Space) The algorithm $\mathcal{A}$ uses $\mathcal{O}\left(\frac{1}{\alpha^3\eta^3}\log^5 n\right)$ bits.

We remark that the dependency on α and η can be improved to $\frac{1}{\alpha\eta}$ with a worse dependency on $\log n$ by using an algorithm of [BGL+18] that achieves a $(\alpha, 0)$-approximation to the number of distinct elements in the sliding window model, without using the smooth histogram framework.

L_p norm and F_p moment estimation. We remark that [BO10] showed that the L_p norm and F_p moment estimation problems are $(\alpha, \alpha^p/p)$-smooth for $p \geq 1$ and (α, α)-smooth for $0 < p \leq 1$. Therefore, we can apply the smooth histogram framework to the L_p norm and F_p moment estimation problem and then further apply our general framework to the smooth histogram framework. However, it turns out this approach gives a sub-optimal result, since there exist the following more efficient algorithms for L_p norm estimation in the sliding window model:

Lemma 5.24. *[WZ21] Given α and $p \in (0,2]$, there exists a one-pass algorithm in the sliding window model that outputs a $(\alpha, 0)$-approximation to the L_p norm/F_p moment with probability at least $1-\frac{1}{\text{poly}(n)}$. The algorithm uses $\mathcal{O}\left(\frac{1}{\alpha^2}\log^3 n \log^3 \frac{1}{\alpha}\right)$ space for $p = 2$ and $\mathcal{O}\left(\frac{1}{\alpha^2}\log^3 n(\log\log n)^2 \log^3 \frac{1}{\alpha}\right)$ space for $p \in (0,2)$.*

Corollary 5.25. *For any integer $K > 0$, $p \in (0,2]$, and for any $\alpha > 0$, $c > 0$, $\varepsilon > 0$, and $\rho := \mathcal{O}\left(\frac{\alpha\eta}{\log m}\right)$ on a stream $\mathfrak{S}$ of length $m = \text{poly}(n)$, where $\eta = \min(\varepsilon, 1)$. Then there exists a sliding window algorithm $\mathcal{A}$ such that:*

(1) (Privacy) The algorithm $\mathcal{A}$ is ε-differentially private where ε is constant.

(2) (Accuracy) The algorithm $\mathcal{A}$ outputs $\widehat{X}$ such that with probability at least $1-\frac{1}{m^c}-\frac{1}{\text{poly}(m)(e^{\varepsilon}-1)/K+1}$,

$$(1-\alpha)X - \frac{12c\log m}{\varepsilon} - \frac{1}{K\,\text{poly}(m)} \le \widehat{X} \le (1+\alpha)X + \frac{12c\log m}{\varepsilon} + \frac{1}{K\,\text{poly}(m)},$$

where X is the L_p norm of the frequency vector induced by the most recent $\min(m, W)$ updates of the stream for $p \in (1,2]$ and X is the F_p frequency moment of the same frequency vector for $p \in (0,1)$.

(3) (Time/Space) The algorithm uses $\tilde{\mathcal{O}}\left(\frac{1}{\alpha^2\eta^2}\log^5 n \log^3 \frac{1}{\alpha\eta}\right)$ space for $p = 2$ and $\tilde{\mathcal{O}}\left(\frac{1}{\alpha^2\eta^2}\log^5 n\right)$ space for $p \in (0,2)$.

6 Conclusion and Open Questions

In this work, we introduce a general framework for transforming a non-private approximation algorithm into a differentially private approximation algorithm. We show specific applications of our framework for sublinear time and sublinear space algorithms. Although our framework applies to a large variety of problems and settings, it does incur a small penalty in both runtime and space for achieving differential privacy. A natural question is whether these losses are necessary for a general black-box framework and what are sufficient conditions for achieving a black-box reduction.

It also seems possible that our framework could provide a method for achieving differentially private algorithms when the important resource is not runtime, number of queries, or space. For example, in distributed algorithms, it is often desired to achieve sublinear communication while in learning/testing, it is often desired to achieve sublinear query complexity. We believe that exploring the limits and capabilities of our framework in those settings would be a natural future direction of work.

References

[AdlVKK03] Noga Alon, Wenceslas Fernandez de la Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of max-csps. *J. Comput. Syst. Sci.*, 67(2):212–243, 2003.

[AE02] Gunnar Andersson and Lars Engebretsen. Property testers for dense constraint satisfaction programs on finite domains. *Random Struct. Algorithms*, 21(1):14–32, 2002.

[AFNS09] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. *SIAM J. Comput.*, 39(1):143–167, 2009.

[AGM12] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 459–467, 2012.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[ASZ18] Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. Differentially private testing of identity and closeness of discrete distributions. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6879–6891, 2018.

[BBD$^+$02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 1–16, 2002.

[BBDS12] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 410–419, 2012.

[BDM$^+$20] Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P. Woodruff, and Samson Zhou. Near optimal linear algebra in the online and sliding window models. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 517–528, 2020.

[BGK$^+$21] Zhiqi Bu, Sivakanth Gopi, Janardhan Kulkarni, Yin Tat Lee, Judy Hanwen Shen, and Uthaipon Tantipongpipat. Fast and memory efficient differentially private-sgd via JL projections. *CoRR*, abs/2102.03013, 2021.

[BGL$^+$18] Vladimir Braverman, Elena Grigorescu, Harry Lang, David P. Woodruff, and Samson Zhou. Nearly optimal distinct elements and heavy hitters on sliding windows. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 7:1–7:22, 2018.

[BGM22] Jeremiah Blocki, Elena Grigorescu, and Tamalika Mukherjee. Privately estimating graph parameters in sublinear time. In *49th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 26:1–26:19, 2022.

[BKM14] 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.

[Bla20] Jaroslaw Blasiok. Optimal streaming and tracking distinct elements with high probability. *ACM Trans. Algorithms*, 16(1):3:1–3:28, 2020.

[BMWZ22] Vladimir Braverman, Joel Manning, Zhiwei Steven Wu, and Samson Zhou. Private data stream analysis for universal symmetric norm estimation. In *3rd Annual Symposium on Foundations of Responsible Computing FORC*, 2022.

[BO07] Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Proceedings*, pages 283–293, 2007.

[BO10] Vladimir Braverman and Rafail Ostrovsky. Effective computations on sliding windows. *SIAM J. Comput.*, 39(6):2113–2131, 2010.

[BOZ12] Vladimir Braverman, Rafail Ostrovsky, and Carlo Zaniolo. Optimal sampling from sliding windows. *J. Comput. Syst. Sci.*, 78(1):260–272, 2012.

[BUV18] Mark Bun, Jonathan R. Ullman, and Salil P. Vadhan. Fingerprinting codes and the price of approximate differential privacy. *SIAM J. Comput.*, 47(5):1888–1938, 2018.

[BWZ21] Vladimir Braverman, Viska Wei, and Samson Zhou. Symmetric norm estimation and regression on sliding windows. In *Computing and Combinatorics - 27th International Conference, COCOON, Proceedings*, pages 528–539, 2021.

[CG08] Graham Cormode and Minos N. Garofalakis. Streaming in a connected world: querying and tracking distributed data streams. In *EDBT 2008, 11th International Conference on Extending Database Technology, Proceedings*, page 745, 2008.

[CM05] Graham Cormode and S. Muthukrishnan. What's new: finding significant differences in network data streams. *IEEE/ACM Transactions on Networking*, 13(6):1219–1232, 2005.

[CRT05] Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005.

[DL09] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC*, pages 371–380. ACM, 2009.

[DM07] Mayur Datar and Rajeev Motwani. The sliding-window computation model and results. In *Data Streams - Models and Algorithms*, pages 149–167. Springer, 2007.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. 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, Proceedings*, pages 265–284, 2006.

[DMNS16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51, 2016.

[Dwo06] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP, Proceedings, Part II*, pages 1–12, 2006.

[ELRS17] Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. *SIAM J. Comput.*, 46(5):1603–1646, 2017.

[ELVZ17] Alessandro Epasto, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghaddam. Submodular optimization over sliding windows. In *Proceedings of the 26th International Conference on World Wide Web, WWW*, pages 421–430, 2017.

[EMM$^+$23] Alessandro Epasto, Jieming Mao, Andres Muñoz Medina, Vahab Mirrokni, Sergei Vassilvitskii, and Peilin Zhong. Differentially private continual releases of streaming frequency moment estimations. In *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.

[EMMZ22] Alessandro Epasto, Mohammad Mahdian, Vahab S. Mirrokni, and Peilin Zhong. Improved sliding window algorithms for clustering and coverage via bucketing-based sketches. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 3005–3042, 2022.

[FN07] Eldar Fischer and Ilan Newman. Testing versus estimation of graph properties. *SIAM J. Comput.*, 37(2):482–501, 2007.

[FR21] Nimrod Fiat and Dana Ron. On efficient distance approximation for graph properties. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1618–1637. SIAM, 2021.

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

[GKMN21] Badih Ghazi, Ravi Kumar, Pasin Manurangsi, and Thao Nguyen. Robust and private learning of halfspaces. 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 1603–1611. PMLR, 2021.

[GLM$^+$10] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1106–1125, 2010.

[GMRS22] Arijit Ghosh, Gopinath Mishra, Rahul Raychaudhury, and Sayantan Sen. Tolerant bipartiteness testing in dense graphs. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 69:1–69:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[GR04] Oded Goldreich and Dana Ron. On estimating the average degree of a graph. *Electron. Colloquium Comput. Complex.*, 2004.

[GW18] Sumit Ganguly and David P. Woodruff. High probability frequency moment sketches. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 58:1–58:15, 2018.

[GXP$^+$20] Maoguo Gong, Yu Xie, Ke Pan, Kaiyuan Feng, and Alex Kai Qin. A survey on differentially private machine learning [review article]. *IEEE Comput. Intell. Mag.*, 15(2):49–64, 2020.

[HT10] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. 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 705–714. ACM, 2010.

[JWZ22] Rajesh Jayaram, David P. Woodruff, and Samson Zhou. Truly perfect samplers for data streams and sliding windows. In *PODS: International Conference on Management of Data*, pages 29–40, 2022.

[KNPW11] Daniel M. Kane, Jelani Nelson, Ely Porat, and David P. Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, pages 745–754, 2011.

[KNW10] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 41–52, 2010.

[MM12] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. *PVLDB*, 5(12):1699, 2012.

[MMNW11] Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 37–48, 2011.

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 75–84, 2007.

[OMM$^+$14] Miles Osborne, Sean Moran, Richard McCreadie, Alexander Von Lunen, Martin Sykora, Elizabeth Cano, Neil Ireson, Craig MacDonald, Iadh Ounis, Yulan He, Tom Jackson, Fabio Ciravegna, and Ann O'Brien. Real-time detection, tracking and monitoring of automatically discovered events in social media. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

[PGD15] Odysseas Papapetrou, Minos N. Garofalakis, and Antonios Deligiannakis. Sketching distributed sliding-window data streams. *VLDB J.*, 24(3):345–368, 2015.

[PRR06] Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006.

[SST20] Adam D. Smith, Shuang Song, and Abhradeep Thakurta. The flajolet-martin sketch itself preserves differential privacy: Private counting with minimal space. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

[SW07] Xiaoming Sun and David P. Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 336–345, 2007.

[Tet22] Jakub Tetek. Additive noise mechanisms for making randomized approximation algorithms differentially private. *CoRR*, abs/2211.03695, 2022.

[WPS22] Lun Wang, Iosif Pinelis, and Dawn Song. Differentially private fractional frequency moments estimation with polylogarithmic space. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

[WZ21] David P. Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1183–1196. IEEE, 2021.

[YYI12] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. Improved constant-time approximation algorithms for maximum matchings and other optimization problems. *SIAM J. Comput.*, 41(4):1074–1093, 2012.

A Proof of Approximate DP to Pure DP transformation

Theorem 1.5. *Let $M = \max_D f(D)$ and let parameter $K > 0$. If $\mathcal{A}'_f(D)$ is (ε, δ)-DP algorithm with accuracy guarantee $(1-\alpha)f(D) - \kappa \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D) + \kappa$ holding with probability $1-\eta$ then there exists an algorithm $\mathcal{A}''_f(D)$ which is ε-DP with accuracy guarantee $(1-\alpha)f(D) - \kappa - \frac{1}{KM} \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D) + \kappa + \frac{1}{KM}$ with probability at least $1 - \eta - p$ where $p = \frac{\delta K(M+1)}{e^{\varepsilon}-1+\delta K(M+1)}$.*

Proof. Note that WLOG we can assume that $\mathcal{A}_f(D)$ outputs a value between 0 and M since we can always truncate the output to this range — this operation preserves privacy by postprocessing and does not adversely affect accuracy. For some $K > 0$, define algorithm $\mathcal{A}''_f(D)$ as outputting $\frac{\lceil K\mathcal{A}_f(D) \rceil}{KM}$. Observe that $\mathcal{A}''_f$ is (ε, δ)-DP by postprocessing and the accuracy guarantee of $\mathcal{A}''_f$ is almost identical to that of $\mathcal{A}_f$ since by definition $|\mathcal{A}''_f(D) - \mathcal{A}_f(D)| < \frac{1}{KM}$. By post-processing we can ensure that the range $\mathcal{R}$ of $\mathcal{A}''_f(D)$ is small $|\mathcal{R}| = (M+1)K$ since $\mathcal{R} = \{\frac{i}{KM} : 0 \leq i \leq KM\}$. Thus, we can pick p such that $\delta \leq \frac{(e^{\varepsilon}-1)p}{|\mathcal{R}|(1-p)}$ and apply a folklore theorem (see Theorem 2.2) to transform our (ε, δ)-DP algorithm $\mathcal{A}''_f(D)$ to an ε-DP algorithm $\mathcal{A}'_f(D)$ in the following manner:

$$\mathcal{A}'_f(D) = \begin{cases} \mathcal{A}''_f(D) & \text{with probability } 1-p \\ \mathsf{random}(\mathcal{R}) & \text{with probability } p \end{cases}$$

By combining the accuracy guarantees of $\mathcal{A}_f$ and $\mathcal{A}''_f$ we see that with probability $1-\eta-p$, we have that $(1-\alpha)f(D)-\kappa-\frac{1}{KM} \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D)+\kappa+\frac{1}{KM}$ where $p=\frac{\delta K(M+1)}{e^{\varepsilon}-1+\delta K(M+1)}$ as claimed. ☐

Theorem 2.2. *[Approximate DP to Pure DP] Let $\mathcal{A}: \mathcal{D} \to \mathcal{R}$. If $\mathcal{A}$ is an (ε,δ)-DP algorithm such that $\delta \leq \frac{(e^{\varepsilon}-1)p}{|\mathcal{R}|(1-p)}$ then there is an algorithm $\mathcal{A}'$ such that $\mathcal{A}'$ is ε-DP defined in the following manner.*

$$\mathcal{A}'(D) = \begin{cases} \mathcal{A}(D) & \textit{with probability } 1-p \\ \mathsf{random}(\mathcal{R}) & \textit{with probability } p \end{cases}$$

where $\mathcal{R}$ is the range of $\mathcal{A}_f$.

Proof. Recall that we define $\mathcal{A}'$ as follows:

$$\mathcal{A}'(D) = \begin{cases} \mathcal{A}(D) & \text{with probability } 1-p \\ \mathsf{random}(\mathcal{R}) & \text{with probability } p \end{cases}$$

Let $D, D' \in \mathcal{D}$ be neighboring databases and fix output $y \in \mathcal{R}$. We first give a general claim regarding the probability of $\mathcal{A}'(D)=y$ in terms of the $\Pr[\mathcal{A}(D)=y]$.

Claim A.1. *For $D \in \mathcal{D}$,*

$$\Pr[\mathcal{A}'(D)=y] = \Pr[\mathcal{A}(D)=y]\,(1-p) + \frac{p}{|\mathcal{R}|}$$

Now we need to show that $\Pr[\mathcal{A}'(D)=y] \leq e^{\varepsilon}\Pr[\mathcal{A}'(D')=y]$.

$$\begin{aligned}
&\Pr[\mathcal{A}'(D)=y] \\
&= \Pr[\mathcal{A}(D)=y]\,(1-p) + \frac{p}{|\mathcal{R}|} \\
&\leq (1-p)\,(e^{\varepsilon}\Pr[\mathcal{A}(D')=y]+\delta) + \frac{p}{|\mathcal{R}|} \\
&\leq e^{\varepsilon}\Pr[\mathcal{A}(D')=y]\,(1-p) + \delta\,(1-p) + \frac{p}{|R|} && (4)\\
&= e^{\varepsilon}(\Pr[\mathcal{A}'(D')=y] - \frac{p}{|\mathcal{R}|}) + \delta\,(1-p) + \frac{p}{|\mathcal{R}|} && (5)\\
&\leq e^{\varepsilon}\Pr[\mathcal{A}'(D')=y] + \delta\,(1-p) + \frac{p}{|\mathcal{R}|}(1-e^{\varepsilon}) \\
&\leq e^{\varepsilon}\Pr[\mathcal{A}'(D')=y] && (6)
\end{aligned}$$

The transition 4 to 5 follows from the observation that $\Pr[\mathcal{A}'(D')=y] = (1-p)\Pr[\mathcal{A}(D')=y] + \frac{p}{|\mathcal{R}|}$ and therefore, $(1-p)\Pr[\mathcal{A}(D')=y] = \Pr[\mathcal{A}'(D')=y] - \frac{p}{|\mathcal{R}|}$. The last equation 6 follows because $\delta \leq \frac{(e^{\varepsilon}-1)p}{|\mathcal{R}|(1-p)}$ and thus

$$\delta\,(1-p) + \frac{p}{|\mathcal{R}|}(1-e^{\varepsilon}) \leq 0\ .$$

☐