Differential Privacy for Growing Databases

Rachel Cummings*  Sara Krehbiel†  Kevin A. Lai‡  Uthaipon (Tao) Tantipongpipat§

March 20, 2018

Abstract

We study the design of differentially private algorithms for adaptive analysis of dynamically growing databases, where a database accumulates new data entries while the analysis is ongoing. We provide a collection of tools for machine learning and other types of data analysis that guarantee differential privacy and accuracy as the underlying databases grow arbitrarily large. We give both a general technique and a specific algorithm for adaptive analysis of dynamically growing databases. Our general technique is illustrated by two algorithms that schedule black box access to some algorithm that operates on a fixed database to generically transform private and accurate algorithms for static databases into private and accurate algorithms for dynamically growing databases. These results show that almost any private and accurate algorithm can be rerun at appropriate points of data growth with minimal loss of accuracy, even when data growth is unbounded. Our specific algorithm directly adapts the private multiplicative weights algorithm of [HR10] to the dynamic setting, maintaining the accuracy guarantee of the static setting through unbounded data growth. Along the way, we develop extensions of several other differentially private algorithms to the dynamic setting, which may be of independent interest for future work on the design of differentially private algorithms for growing databases.

1 Introduction

Technological advances are continuously driving down the cost of data collection and storage. Data collection devices such as smartphones and wearable health monitors have become ubiquitous, resulting in continuous accumulation of new data. This means that the statistical properties of a database may evolve dramatically over time, and earlier analysis of a database may grow stale. For example, tasks like identifying trending news topics critically rely on dynamic data and dynamic analysis. To harness the value of growing databases and keep up with data analysis needs, guarantees of machine learning algorithms and other statistical tools must apply not just to fixed databases but also to dynamic databases.

Learning algorithms must deal with highly personal data in contexts such as wearable health data, browsing behavior, and GPS location data. In these settings, privacy concerns are particularly important. Analysis of sensitive data without formal privacy guarantees has led to numerous privacy violations in practice, for tasks such as recommender systems [CKN+11], targeted advertising [Kor10], data anonymization [NS08], and deep learning [HAPC17].

*School of Industrial and Systems Engineering, Georgia Institute of Technology. Email: rachelc@gatech.edu. Supported in part by a Mozilla Research Grant.
†Department of Math and Computer Science, University of Richmond. Email: krehbiel@richmond.edu. Supported in part by a Mozilla Research Grant.
‡School of Computer Science, Georgia Institute of Technology. Email: kevinlai@gatech.edu. Supported in part by National Science Foundation grant IIS-1453304.
§School of Computer Science, Georgia Institute of Technology. Email: uthaipon3@gatech.edu. Supported in part by National Science Foundation grants CCF-24067E5 and CCF-1740776 (Georgia Institute of Technology TRIAD), and by a Georgia Institute of Technology ARC fellowship. Part of this work was completed while the author was a student at University of Richmond.
In the last decade, a growing literature on differential privacy has developed to address these concerns (see, e.g., DR14). First defined by DMNS06, differential privacy gives a mathematically rigorous worst-case bound on the maximum amount of information that can be learned about any one individual’s data from the output of an algorithm. The theoretical computer science community has been prolific in designing differentially private algorithms that provide accuracy guarantees for a wide variety of machine learning problems (see JLE14 for a survey). Differentially private algorithms have also begun to be implemented in practice by major organizations such as Apple, Google, Uber, and the United Status Census Bureau. However, the vast majority of differentially private algorithms are designed only for static databases, and are ill-equipped to handle new environments with growing data.

This paper presents a collection of tools for machine learning and other types of data analysis that guarantee differential privacy and accuracy as the underlying databases grow arbitrarily large. We give both a general technique and a specific algorithm for adaptive analysis of dynamically growing databases. Our general technique is illustrated by two algorithms that schedule black box access to some algorithm that operates on a fixed database, to generically transform private and accurate algorithms for static databases into private and accurate algorithms for dynamically growing databases. Our specific algorithm directly adapts the private multiplicative weights algorithm of HR10 to the dynamic setting, maintaining the accuracy guarantee of the static setting through unbounded data growth.

1.1 Our Results

Here we outline our two sets of results for adaptive analysis of dynamically growing databases. Throughout the paper, we refer to the setting in which a database is fixed for the life of the analysis as the static setting, and we refer to the setting in which a database is accumulating new data entries while the analysis is ongoing as the dynamic setting.

Our first set of results consists of two methods for generically transforming a black box algorithm that is private and accurate in the static setting into an algorithm that is private and accurate in the dynamic setting. BBScheduler reruns the black box algorithm every time the database increases in size by a small multiplicative factor, and it provides privacy and accuracy guarantees that are independent of the total number of queries (Theorem 10). BBScheduler calls each successive run of the black box algorithm with an exponentially shrinking privacy parameter to achieve any desired total privacy loss. The time-independent accuracy guarantee arises from the calibration of the decreasing per-run privacy parameters with the increasing database size. We instantiate this scheduler using the SMALLDB algorithm [BLR08] for answering k linear queries on a database of size n over a universe of size N. With our scheduler we can answer an infinite amount of queries from a linear query class of size k on a growing database with starting size n over universe of size N. The static and dynamic settings have the following respective accuracy guarantees (Theorem 7, [BLR08]; Theorem 9):

\[ \alpha_s = O \left( \frac{\log N \log k}{\epsilon n} \right)^{1/3} \]

\[ \alpha_d = O \left( \frac{\log N \log k}{\epsilon n} \right)^{1/5} \]

Our second transformation, BBImprover, runs the black box every time new entries are added to the database, and it yields accuracy guarantees that improve as more data accumulate. This algorithm is well-suited for problems where data points are sampled from a distribution, where one would expect the accuracy guarantees of static analysis to improve with the size of the sample. We apply this scheduler to private empirical risk minimization (ERM) algorithms to output classifiers with generalization error that improves as the training database grows (Theorem 12).

The following informal theorem statement summarizes our results for BBScheduler (Theorem 11) and BBImprover (Theorem 13). These results show that almost any private and accurate algorithm can be rerun at appropriate points of data growth with minimal loss of accuracy. Throughout the paper, we use \( n \) to denote the starting size of the database. The \( \tilde{O} \) below hides \( \text{poly}(n) \) terms, and we suppress dependence on parameters other than \( \epsilon \) and \( n \), (e.g., data universe size \( N \), number of queries \( k \), failure probability \( \beta \)). Section 8 also provides an improved \((\epsilon, \delta)\)-private version of BBScheduler.
Theorem 1 (Informal). Let \( \mathcal{M} \) be an \( \epsilon \)-differentially private algorithm that is \((\alpha, \beta)\)-accurate for an input query stream for some \( \alpha = \tilde{O}\left(\left(\frac{1}{\epsilon^p}\right)^{\frac{1}{2p+1}}\right) \) and constant \( p \). Then

1. BBScheduler running \( \mathcal{M} \) is \( \epsilon \)-differentially private and \((\alpha, \beta)\)-accurate for \( \alpha = \tilde{O}\left(\left(\frac{1}{\epsilon^p}\right)^{\frac{1}{2p+1}}\right) \).

2. BBImprover running \( \mathcal{M} \) is \((\epsilon, \delta)\)-differentially private and is \((\{\alpha_t\}_{t \geq n}, \beta)\)-accurate for \( \alpha_t = \tilde{O}\left(\left(\frac{\sqrt{\log(1/\delta)}}{\epsilon \sqrt{t}}\right)^p\right) \),

where \( \alpha_t \) bounds the error when the database is size \( t \geq n \).

Our second set of results opens the black box to increase accuracy and adaptivity by modifying the private multiplicative weights (PMW) algorithm [HR10], a broadly useful algorithm for privately answering an adaptive stream of \( k \) linear queries with accuracy \( \alpha = O(\log N \log k)^{1/3} \). Our modification for growing databases (PMWG) considers all available data when any query arrives, and it suffers asymptotically no additional accuracy cost relative to the static setting.

The static PMW algorithm answers an adaptive stream of queries while maintaining a public histogram reflecting the current estimate of the database given all previously answered queries. It categorizes incoming queries as either easy or hard, suffering significant privacy loss only for the hard queries. Hardness is determined with respect to the public histogram: upon receipt of a query for which the histogram provides a significantly different answer than the true database, PMW classifies this as a hard query, and it updates the histogram in a way that moves it closer to a correct answer on that query. The number of hard queries is bounded using a potential argument. Potential is defined as the relative entropy between the database and the public histogram. This quantity is initially bounded, it decreases by a substantial amount after every hard query, and it never increases.

The main challenge in adapting PMW to the dynamic setting is that we can no longer use this potential argument to bound the number of hard queries. This is because the relative entropy between the database and the public histogram can increase as new data arrive. In the worst case, PMW can learn the database with high accuracy (using many hard queries), and then adversarial data growth can change the composition of the database dramatically, allowing for even more hard queries on the new data than is possible in the static setting. Instead, we modify PMW so that the public histogram updates not only in response to hard queries but also in response to new data arrivals. By treating the new data as coming from a uniform distribution, these latter updates incur no additional privacy loss, and they mitigate the relative entropy increase due to new data. This modification allows us to maintain the accuracy guarantee of the static setting through unbounded data growth. The following informal theorem is a statement of our main result for PMWG (Theorem 13). As with the static PMW algorithm, we can improve the exponent in the bound to \( 1/2 \) if our goal is \((\epsilon, \delta)\)-privacy for \( \delta > 0 \) (Theorem 17).

Theorem 2 (Informal). PMWG is \( \epsilon \)-differentially private and \((\alpha, \beta)\)-accurate for any stream of up to \( \kappa \cdot \exp(\sqrt{t/n}) \) queries when the database is any size \( t \geq n \) for some \( \alpha = O\left(\left(\frac{\log N \log \kappa}{\epsilon^p}\right)^{1/3}\right) \).

Along the way, we develop extensions of several static differentially private algorithms to the dynamic setting. These algorithms are presented in Appendix [C] and may be of independent interest for future work on the design of differentially private algorithms for growing databases.

1.2 Related Work

Online Learning. Our setting of dynamically growing databases is most closely related to online learning, where a learner plays a game with an adversary over many rounds. On each round \( t \), the adversary first gives the learner some input, then the learner chooses an action \( a_t \) and receives loss function \( \mathcal{L}_t \) chosen by the adversary, and experiences loss \( \mathcal{L}_t(a_t) \). There is a vast literature on online learning, including several works on differentially private online learning [JKT12, ST13, AS17]. In those settings, a database is a sequence of loss functions, and neighboring databases differ on a single loss function. While online learning resembles the dynamic database setting, there are several key differences. Performance bounds in the online setting are
in terms of regret, which is a cumulative error term. On the other hand, we seek additive error bounds that hold for all of our answers. Such bounds are not possible in general for online learning, since the inputs are adversarial and the true answer is not known. In our case, we can achieve such bounds because even though queries are presented adversarially, we have access to the query’s true answer. Instead of a cumulative error bound, we manage a cumulative privacy budget.

Private Learning on a Static Database. There is a prominent body of work designing differentially private algorithms in the static setting for a wide variety of machine learning problems (see [JLE14] for a survey). These private and accurate algorithms can be used as black boxes in our schedulers BBScheduler and BBImprover. In this paper, we pay particular attention to the problem of private empirical risk minimization (ERM) as an instantiation for our algorithms. Private ERM has been previously studied by [CMS11, KST+12, BST14]: we compare our accuracy bounds in the dynamic setting to their static bounds in Table 1.

Private Adaptive Analysis of a Static Database. If we wish to answer multiple queries on the same database by independently perturbing each answer, then the noise added to each answer must scale linearly with the number of queries to maintain privacy, meaning only $O(n)$ queries can be answered with meaningful privacy and accuracy. If the queries are known in advance, however, [BLR08] showed how to answer exponentially many queries relative to the database size for fixed meaningful privacy and accuracy. If the queries are known in advance, however, [BLR08] showed how to answer exponentially many queries relative to the database size for fixed $\epsilon$ and $\alpha$. Later, Private Multiplicative Weights (PMW) [HR10] achieved a similar result in the interactive setting, where the analyst can adaptively decide which queries to ask based on previous outputs, with accuracy guarantees close to the sample error. A recent line of work [DFH15, CLN+16, BNS+16] showed deep connections between differential privacy and adaptive data analysis of a static database. Our results would allow analysts to apply these tools on dynamically growing databases.

Private Non-Adaptive Analysis of a Dynamic Database. Differential privacy for growing databases has been studied for a limited class of problems. Both [DNPR10] and [CSS11] adapted the notion of differential privacy to streaming environments in a setting where each entry in the database is a single bit, and bits arrive one per unit time. [DNPR10] and [CSS11] design differentially private algorithms for an analyst to maintain an approximately accurate count of the number of 1-bits seen thus far in the stream. This technique was later extended by [ST13] to maintain private sums of real vectors arriving online in a stream. We note that both of these settings correspond to only a single query repeatedly asked on a dynamic database, precluding meaningful adaptive analysis. To contrast, we consider adaptive analysis of dynamically growing databases, allowing the analyst exponentially many predicate queries to choose from as the database grows.

2 Preliminaries

All algorithms in this paper take as input databases over some fixed data universe $X$ of finite size $N$. Our algorithms and analyses represent a finite database $D \in X^n$ equivalently as a fractional histogram $x \in \Delta(X) \subseteq \mathbb{R}^N$, where $x^i$ is the fraction of the database of type $i \in [N]$. When we say a database $x \in \Delta(X)$ has size $n$, this means that for each $i \in [N]$ there exists some $n_i \in \mathbb{N}$ such that $x^i = n_i/n$.

If an algorithm operates over a single fixed database, we refer to this as the static setting. For the dynamic setting, we define a database stream to be a sequence of databases $X = \{x_t\}_{t \geq n}$ starting with a database $x_n$ of size $n$ at time $t = n$ and increasing by one data entry per time step so that $t$ always denotes both a time and the size of the database at that time. Our dynamic algorithms also take a parameter $n$, which denotes the starting size of the database.

We consider algorithms that answer real-valued queries $f : \mathbb{R}^N \to \mathbb{R}$ with particular focus on linear queries. A linear query assigns a weight to each entry depending on its type and averages these weights over the database. We can interpret a linear query as a vector $f \in [0,1]^N$ and write the answer to the query on database $x \in \Delta(X)$ as $(f \cdot x)$, $f(x)$, or $x(f)$, depending on context. For $f$ viewed as a vector, $f^i$ denotes the $i$th entry. We note that an important special case of linear queries are counting queries, which calculate the proportion of entries in a database satisfying some boolean predicate over $X$.

---

1To get the static bounds, we use Appendix D of [BST14], which converts bounds on expected excess empirical risk to high probability bounds.
Many of the algorithms we study allow queries to be chosen adaptively, i.e., the algorithm accepts a stream of queries \( F = \{ f_j \}_{j=1}^{k} \) where the choice of \( f_{j+1} \) can depend on the previous \( j \) queries and answers. For the dynamic setting, we doubly index a stream of queries as \( F = \{ f_t \}_{t \geq n} = \{ \{ f_{t,j} \}_{j=1}^{k} \}_{t \geq n} \) so that \( t \) denotes the size of the database at the time \( f_{t,j} \) is received, and \( j = 1, \ldots, \ell_t \) indexes the queries received when the database is size \( t \).

The algorithms studied produce outputs of various forms. To evaluate accuracy, we assume that an output \( y \) of an algorithm for query class \( F \) (possibly specified by an adaptively chosen query stream) can be interpreted as a function over \( F \), i.e., we write \( y(f) \) to denote the answer to \( f \in F \) based on the mechanism’s output. We seek to develop mechanisms that are accurate in the following sense.

**Definition 1** (Accuracy in the static setting). For \( \alpha, \beta > 0 \), an algorithm \( \mathcal{M} \) is \((\alpha, \beta)\)-accurate for query class \( F \) if for any input database \( x \in \Delta(\mathcal{X}) \), the algorithm outputs \( y \) such that \( |f(x) - y(f)| \leq \alpha \) for all \( f \in F \) with probability at least \( 1 - \beta \).

In the dynamic setting, accuracy must be with respect to the current database, and the bounds may be parametrized by time.

**Definition 2** (Accuracy in the dynamic setting). For \( \alpha_n, \alpha_{n+1}, \ldots > 0 \) and \( \beta > 0 \), an algorithm \( \mathcal{M} \) is \((\{\alpha_t\}_{t \geq n}, \beta)\)-accurate for query stream \( F = \{ f_t \}_{t \geq n} \) if for any input database stream \( X = \{ x_t \}_{t \geq n} \), the algorithm outputs \( y \) such that \( |f_t(x_t) - y(f_t)| \leq \alpha_t \) for all \( f_t, j \in F \) with probability at least \( 1 - \beta \).

### 2.1 Differential Privacy and Composition Theorems

Differential privacy in the static setting requires that an algorithm produce similar outputs on neighboring databases \( x \sim x' \), which differ by a single entry. In the dynamic setting, differential privacy requires similar outputs on neighboring database streams \( X, X' \) that satisfy that for some \( t \geq n \), \( x_\tau = x'_\tau \) for \( \tau = n, \ldots, t - 1 \) and \( x_t \sim x'_t \) for \( \tau \geq t \).

In the definition below, a pair of *neighboring inputs* refers to a pair of neighboring databases in the static setting or a pair of neighboring database streams in the dynamic setting.

**Definition 3** (Differential privacy [DMNS06]). For \( \epsilon, \delta > 0 \), an algorithm \( \mathcal{M} \) is \((\epsilon, \delta)\)-differentially private if for any pair of neighboring inputs \( x, x' \) and any subset \( S \subseteq \text{Range}(\mathcal{M}) \),

\[
\Pr[\mathcal{M}(x) \in S] \leq e^\epsilon \cdot \Pr[\mathcal{M}(x') \in S] + \delta.
\]

When \( \delta = 0 \), we will say that \( \mathcal{M} \) is \( \epsilon \)-differentially private.

We note that in the dynamic setting, an element in \( \text{Range}(\mathcal{M}) \) is an entire (potentially infinite) transcript of outputs that may be produced by \( \mathcal{M} \).

Differential privacy is typically achieved by adding random noise that scales with the sensitivity of the computation being performed. The sensitivity of any real-valued query \( f : \Delta(\mathcal{X}) \rightarrow \mathbb{R} \) is the maximum change in the query’s answer due to the change of a single entry in the database, denoted \( \Delta_f = \max_{x \sim x'} |f(x) - f(x')| \). We note that a linear query on a database of size \( n \) has sensitivity \( 1/n \).

The following composition theorems quantify how the privacy guarantee degrades as additional computations are performed on a database.

**Theorem 3** (Basic composition [DMNS06]). Let \( \mathcal{M}_i \) be an \( \epsilon_i \)-differentially private algorithm for all \( i \in [k] \). Then the composition \( \mathcal{M} \) defined as \( \mathcal{M}(x) = (\mathcal{M}_i(x))_{i=1}^{k} \) is \( \epsilon \)-differentially private for \( \epsilon = \sum_{i=1}^{k} \epsilon_i \).

Theorem 3 is useful to combine many differentially private algorithms to still achieve \((\epsilon, 0)\)-differential privacy. Assuming the privacy loss in each mechanism is the same, the privacy loss from composing \( k \) mechanisms scales with \( k \). There is an advanced composition theorem due to [DRV10] that improves the privacy loss to roughly \( \sqrt{k} \) by relaxing from \((\epsilon, 0)\)-differential privacy to \((\epsilon, \delta)\)-differential privacy. However, advanced composition does not extend cleanly to the case where each \( \mathcal{M}_i \) has different \( \epsilon_i \). Instead we use a

\[ 2 \text{Note that this definition is equivalent to the definition of neighboring streams in [CPWV16].} \]
composition theorem based on concentrated differential privacy (CDP) of [BS16]. This gives us the flexibility to compose differentially private mechanisms with different $\epsilon_i$ to achieve $(\epsilon, \delta)$-differential privacy, where $\epsilon$ scales comparably to the $\sqrt{k}$ bound of advanced composition.

Theorem 4 (CDP composition, Corollary of [BS16]). Let $M_i$ be a $\epsilon_i$-differentially private algorithm for all $i \in [k]$. Then the composition of all $M_i$ is $(\epsilon, \delta)$-differentially private for $\epsilon = \frac{1}{k} \left( \sum_{i=1}^{k} \epsilon_i^2 \right) + \sqrt{2 \left( \sum_{i=1}^{k} \epsilon_i^2 \right) \log(1/\delta)}$. In particular, for $\delta \leq e^{-1}$ and $\sum_{i=1}^{k} \epsilon_i^2 \leq 1$, we have $\epsilon \leq 2 \sqrt{ \left( \sum_{i=1}^{k} \epsilon_i^2 \right) \log(1/\delta)}$.

Proof. The statement follows from the three following propositions in [BS16]:

1. A mechanism that is $\epsilon$-DP is $\frac{1}{k} \epsilon^2$-zCDP.
2. Composition of $\rho_1$-zCDP and $\rho_2$-zCDP is a $(\rho_1 + \rho_2)$-zCDP mechanism.
3. A $\rho$-zCDP mechanism is $(\rho + 2 \sqrt{\rho \log(1/\delta)}, \delta)$-DP for any $\delta > 0$.

Theorem 4 shows that composing $\epsilon_i$-differentially private algorithms results in $(\epsilon, \delta)$-differential privacy, where the privacy $\epsilon$ scales with the $\ell_2$-norm of the vector $(\epsilon_i)_{i=1}^{k}$ and $\log^{1/2}(1/\delta)$.

2.2 Empirical Risk Minimization

Empirical risk minimization (ERM) is one of the most fundamental tasks in machine learning. The task is to find a good classifier $\theta \in C$ from a set of classifiers $C$, given a database $x$ of size $n$ sampled from some distribution $P$ over $X = \mathbb{R}^d$ and loss function $L : C \times \Delta(X) \rightarrow \mathbb{R}$. The loss $L$ of a classifier $\theta$ on a finite database $x_n = \{z_1, \ldots, z_n\}$ with respect to some $L : C \times X \rightarrow \mathbb{R}$ is defined as $L(\theta; x_n) = \frac{1}{n} \sum_{i=1}^{n} L(\theta; z_i)$. Common choices for $L$ include $0 - 1$ loss, hinge loss, and squared loss.

We seek to find a $\hat{\theta}$ with small excess empirical risk, defined as,

$$\hat{\mathcal{R}}_n(\hat{\theta}) = L(\hat{\theta}; x_n) - \min_{\theta \in C} L(\theta; x_n).$$

(2.1)

In convex ERM, we assume that $L(\cdot; x)$ is convex for all $x \in X$ and that $C$ is a convex set. We will also assume that $X \subseteq \mathbb{R}^d$. Convex ERM is convenient because finding a suitable $\hat{\theta}$ reduces to a convex optimization problem, for which there exist many fast algorithms. Some examples of ERM include finding a $d$-dimensional median and SVM.

ERM is useful due to its connections to the true risk, also known as the generalization error, defined as $\mathcal{R}(\theta) = E_{x \sim P}[L(\theta; x)]$. That is, the loss function will be low in expectation on a new data point sampled from $P$. We can also define the excess risk of a classifier $\theta$:

$$\text{ExcessRisk}(\theta) = \mathbb{E}_{x \sim P} \left[ L(\theta; x) \right] - \min_{\theta' \in C} \mathbb{E}_{x \sim P} \left[ L(\theta'; x) \right].$$

ERM finds classifiers with low excess empirical risk, which in turn often have low excess risk. The following theorem relates the two. For completeness, we first give some definitions relating to convex empirical risk minimization. A convex body $C$ is a set such that for all $x, y \in C$ and all $\lambda \in [0, 1]$, $\lambda x + (1 - \lambda) y \in C$. A vector $v$ is a subgradient of a function $L$ at $x_0$ if for all $x \in C$, $L(x) - L(x_0) \geq \langle v, x - x_0 \rangle$. A function $L : C \rightarrow \mathbb{R}$ is $G$-Lipschitz if for all pairs $x, y \in C$, $|L(x) - L(y)| \leq G \|x - y\|_2$. $L$ is $\Delta$-strongly convex on $C$ if for all $x \in C$ and all subgradients $z$ at $x$ and for all $y \in C$, we have $L(y) \geq L(x) + \langle z, y - x \rangle + \frac{\Delta}{2} \|y - x\|_2^2$. $L$ is $B$-smooth on $C$ if for all $x \in C$, for all subgradients $z$ at $x$ and for all $y \in C$, we have $L(y) \leq L(x) + \langle z, y - x \rangle + \frac{B}{2} \|y - x\|_2^2$.

We denote the diameter of a convex set $C$ by $\|C\|_2 = \text{arg max}_{x, y \in C} \|x - y\|_2$.

Theorem 5 ([SSSSS10]). For $G$-Lipschitz and $\Delta$-strongly convex loss functions, with probability at least $1 - \gamma$ over the randomness of sampling the data set $x_n$, the following holds:

$$\text{ExcessRisk}(\hat{\theta}) \leq \sqrt{\frac{2G^2}{\Delta} \hat{\mathcal{R}}_n(\hat{\theta}) + \frac{4G^2}{\gamma \Delta n}}.$$
Moreover, we can generalize this result to any convex and Lipschitz loss function $L$ by defining a regularized version of $L$, called $\tilde{L}$, such that $\tilde{L}(\theta; x) = L(\theta; x) + \frac{\Delta}{2} \| \theta \|^2$. Then $\tilde{L}$ is $(L + \|C\|^2)$-Lipschitz and $\Delta$-strongly convex. Also note that:

$$\text{ExcessRisk}_L(\theta) \leq \text{ExcessRisk}_{\tilde{L}}(\theta) + \frac{\Delta}{2} \| C \|^2.$$ 

Thus, ERM finds classifiers with low true risk in these settings. The following result for differentially private static ERM is due to [BST14] and provides a baseline for our work in the dynamic setting.

**Theorem 6** (Static ERM [BST14]). There exists an algorithm $\text{ERM}(x, C, L, \epsilon, \alpha, \beta, n)$ for $X \subseteq \mathbb{R}^d$ that is $\epsilon$-differentially private and $(\alpha, \beta)$-accurate for static ERM as long as $\|C\|^2 = 1$, $L$ is 1-Lipschitz, and for sufficiently large constant $C$,

$$\alpha \geq C \left( \frac{\log(1/\beta) \log \log(1/\beta)}{\epsilon n} \right)^{1/3}.$$

2.3 SmallDB

The SMALLDB algorithm [BLR08] is a differentially private algorithm for generating synthetic databases. For any input database $x$ of size $n$, class $F$ of linear queries, and accuracy parameter $\alpha$, the algorithm samples a database $y$ of size $\frac{\log|F|}{\alpha}$ with exponential bias towards databases that closely approximate $x$ on all the queries in $F$. The main strength of SMALLDB is its ability to accurately answer exponentially many linear queries while still preserving privacy, captured in the following guarantee.

**Theorem 7** (Static SMALLDB [BLR08]). The algorithm $\text{SMALLDB}(x, F, \epsilon, \alpha, \beta, n)$ is $\epsilon$-differentially private, and it is $(\alpha, \beta)$-accurate for linear query class $F$ of size $|F| = k$ as long as for sufficiently large constant $C$,

$$\alpha \geq C \left( \frac{\log N \log k + \log(1/\beta)}{\epsilon n} \right)^{1/3}.$$

This bound on $\alpha$ shows that for a fixed accuracy goal, the privacy parameter can decrease proportionally as the size of the input database size increases.

2.4 Private Multiplicative Weights

The static private multiplicative weights (PMW) algorithm [HR10] answers an adaptive stream of linear queries while maintaining a public histogram $y$, which reflects the current estimate of the static database $x$ given all previously answered queries. Critical to the performance of the algorithm is that it uses the public histogram to categorize incoming queries as either easy or hard, and it updates the histogram after hard queries in a way that moves it closer to a correct answer on that query. The number of hard queries is bounded using a potential argument, where the potential function is defined as the relative entropy between the database and the public histogram, i.e., $\text{RE}(x||y) = \sum_{i \in [N]} x^i \log(x^i/y^i)$. This quantity is initially bounded, it decreases by a substantial amount after every hard query, and it never increases. The following guarantee illustrates that this technique allows for non-trivial accuracy for exponentially many adaptively chosen linear queries.

**Theorem 8** (Static PMW [HR10]). The algorithm $\text{PMW}(x, F, \epsilon, \delta, \alpha, \beta, n)$ is $(\epsilon, \delta)$-differentially private, and it is $(\alpha, \beta)$-accurate for $k$ adaptively chosen linear queries $F$ as long as for sufficiently large constant $C$,

$$\alpha \geq \begin{cases} 
C \left( \frac{\log N \log(k/\delta)}{\epsilon n} \right)^{1/3} & \text{for } \delta = 0 \\
C \left( \frac{\log^{1/2} N \log(k/\delta) \log(1/\delta)}{\epsilon n} \right)^{1/2} & \text{for } \delta > 0
\end{cases}$$

The bounds cited here are from the updated version in http://mrtz.org/papers/HR10mult.pdf
This result is nearly tight in that any $\epsilon$-differentially private algorithm that answers $k$ adaptively chosen linear queries on a database of size $n$ must have error $\alpha \geq \Omega(\log(N/n) \log k) \frac{1}{\epsilon t}$ [HR10]. PMW runs in time linear in the data universe size $N$. If the incoming data entries are drawn from a distribution that satisfies a mild smoothness condition, a compact representation of the data universe can significantly reduce the runtime [HR10]. The same idea applies to our modification of PMW for the dynamic setting presented in Section 3, but we only present the inefficient and fully general algorithm.

3 Extending Accuracy Guarantees to Growing Databases

In this section, we give two schemes for answering a stream of queries on a growing database, given black box access to a differentially private algorithm for the static setting. Our results extend the privacy and accuracy guarantees of these static algorithms to the dynamic setting, even when data growth is unbounded. We also instantiate our results with important mechanisms for machine learning that are private in the static setting.

In Section 3.1, we provide an algorithm BBSCHEDULER for scheduling repeated runs of a static algorithm. BBSCHEDULER is differentially private and provides $(\alpha, \beta)$-accurate answers to all queries, for $\alpha$ that does not change as the database grows or as more queries are asked. In Section 3.2, we provide a second algorithm BBIMPROVER that allows the accuracy guarantee to improve as more data accumulate. This result is well-suited for problems where data points are sampled from a distribution, where one would expect the accuracy guarantees of static analysis to improve with the size of the sample. This algorithm is differentially private and $(\{\alpha_i\}, \beta)$-accurate, where $\alpha_i$ is diminishing inverse polynomially in $t$ (i.e., approaching perfect accuracy as the database grows large). We also instantiate our results with important mechanisms for machine learning that are private in the static setting.

For ease of presentation, we restrict our results to accuracy of real-valued queries, but the algorithms we propose could be applied to settings with more general notions of accuracy or to settings where the black box algorithm itself can change across time steps, adding to the adaptivity of this scheme.

3.1 Fixed Accuracy as Data Accumulate

In this section, we give results for using a private and accurate algorithm for the static setting as a black box to solve the analogous problem in the dynamic setting. Our general purpose algorithm BBSCHEDULER treats a static algorithm as a black box endowed with privacy and accuracy guarantees, and it reruns the black box whenever the database grows by a small multiplicative factor. For concreteness, we first show in Section 3.1.1 how our results apply to the case of the well-known SMALLDB algorithm, described in Section 2.3. Then in Section 3.1.2, we present the more general algorithm.

3.1.1 Application: SMALLDB for Growing Databases

Before presenting our result in full generality, we instantiate it on SMALLDB for concreteness, and show how to extend SMALLDB to the dynamic setting. Recall from Section 2.3 that the static SMALLDB algorithm takes in a database $x$, a class of linear queries $F$, and privacy parameter $\epsilon$, and accuracy parameters $\alpha$, $\beta$. The algorithm is $\epsilon$-differentially private and outputs a smaller database $y$ of size $\frac{\log |F|}{\alpha}$, from which all queries in $F$ can be answered with $(\alpha, \beta)$-accuracy.

In the dynamic setting, we receive a database stream $X = \{x_t\}_{t \geq n}$, a stream of queries $F = \{\{f_{t,j}\}_{j=1}^{\ell_t}\}_{t \geq n}$ from some class of linear queries $F$, parameters $\epsilon, \alpha, \beta$, and starting database size $n$. We still require $\epsilon$-differential privacy and $(\alpha, \beta)$-accuracy on the entire stream of queries, for $\alpha$ that remains fixed as the database grows.

We design the SMALLDBG algorithm that works by running SMALLDB at times $\{t_i\}_{i=0}^{\infty}$, where $t_i = (1 + \gamma)^i n$ for some $\gamma < 1$ chosen by the algorithm. We will label the time interval from $t_i$ to $t_{i+1} - 1$

\footnote{For simplicity, we will assume that $(1 + \gamma)^i n$ is integral for all $i$. We can replace $(1 + \gamma)^i n$ with $\lceil (1 + \gamma)^i n \rceil$ and achieve the same bounds up to a small sub-constant additive factor.}
as the \(i\)th epoch. At the start of the \(i\)th epoch, we call SMALLDB on the current database with privacy parameter \(\epsilon_i\), and output a synthetic database \(y_i\) that will be used to answer queries received during epoch \(i\). SMALLDBG provides the following guarantee:

**Theorem 9.** SMALLDBG\((X, F, \epsilon, \alpha, \beta, n)\) is \(\epsilon\)-differentially private and can answer all queries in query stream \(F\) from query class \(F\) of size \(|F| = k\) with \((\alpha, \beta)\)-accuracy for sufficiently large constant \(C\) and

\[
\alpha \geq C \left( \frac{\log N \log k \log(1/\beta)}{en} \right)^{1/5}.
\]

Note that there is no bound on the number of queries or on the database growth. The algorithm can provide answers to an arbitrary number of linear queries at any time.

There are two key technical properties that allow this result to hold. First, each data point added to a database of size \(t\) can only change a linear query by roughly \(1/t\). Thus, using synthetic database \(y_i\) from time \(t_i\) for queries before time \((1 + \gamma)t_i\) will incur extra additive error of at most \(\gamma\). Second, since the \(t_i\)'s grow by a multiplicative factor each time, the epochs become exponentially far apart and the total privacy loss (due to composition of multiple calls of SMALLDB) is not too large.

### 3.1.2 A General Black Box Scheduler

The results for SMALLDB are an instantiation of a more general result that extends the privacy and accuracy guarantees of any static algorithm to the dynamic setting. Our general purpose algorithm BBSCHEDULER treats a static algorithm as a black box endowed with privacy and accuracy guarantees, and reruns the black box whenever the database grows by a factor of \((1 + \gamma)\). Due to the generality of this approach, BBSCHEDULER can be applied to any algorithm that satisfies \(\epsilon\)-differential privacy and \((\alpha, \beta)\)-accuracy, as specified in Definition 4.

**Definition 4** \((p, g)\)-black box. An algorithm \(M(x, \epsilon, \alpha, \beta, n)\) is a \((p, g)\)-black box for a class of linear queries \(F\) if it is \((\epsilon, 0)\)-differentially private and with probability \(1 - \beta\) it outputs \(y : F \rightarrow \mathbb{R}\) such that \(|y(f) - x_n(f)| \leq \alpha\) for every \(f \in F\) when \(\alpha \geq g \left( \frac{\log(1/\beta)}{en} \right)^p\) for some \(g\) that is independent of \(\epsilon, n, \beta\).

The parameter \(g\) is intended to capture dependence on domain-specific parameters that affect the accuracy guarantee. For example, SMALLDB is a \((1/3, (64 \log N \log k)^{1/3})\)-black box for an arbitrary set \(F\) of \(k\) linear queries, and its output \(y\) is a synthetic database of size \(\log k/\alpha^2\).

Our generic algorithm BBSCHEDULER (Algorithm 1) will run the black box \(M\) at times \(\{t_i\}_{i=0}^{\infty}\) for \(t_i = (1 + \gamma)^n t_i\) with \(\gamma < 1\) that depends on \(p \) and \(g\). The \(i\)th \(M\) call will have parameters \(\epsilon_i\) and \(\beta_i\), and will use \(y_i\) to answer queries received during the \(i\)th epoch, from \(t_i\) to \(t_{i+1} - 1\).

We now state our main result for BBSCHEDULER:

**Theorem 10.** Let \(M\) be a \((p, g)\)-black box for query class \(F\). Then for any database stream \(X\) and stream of linear queries \(F\) over \(F\), BBSCHEDULER\((X, F, M, \epsilon, \delta, \beta, n, p, g)\) is \((\epsilon, \delta)\)-differentially private for \(\epsilon < 1\) and \((\alpha, \beta)\)-accurate for sufficiently large constant \(C\) and

\[
\alpha \geq \begin{cases} 
Cg^{1/\sqrt{\delta+1}} \left( \frac{\log(1/\beta)}{en} \right)^{\frac{1}{\sqrt{\delta+1}}} & \text{if } \delta = 0 \\
Cp^{1/\sqrt{\delta+1}} \left( \frac{N \log(1/\beta) \log(1/\beta)}{en} \right)^{\frac{1}{\sqrt{\delta+1}}} & \text{if } \delta > 0
\end{cases}
\]

---

5Note that SMALLDBG will still give similar guarantees if the query class changes over time, provided that the black box SMALLDB at time \(t_i\) uses the correct query class \(F_i\) for times \(t_i\) to \(t_{i+1} - 1\). We could think of this as SMALLDBG receiving a SMALLDBG\((F_i, \epsilon_i, \alpha, \beta)\) as its black box in epoch \(i\).

6With a more careful analysis, one can show that the numerator in this accuracy bound can be taken to be \(16 \log N \log |F| + 4 \log \frac{1}{\beta}\) to match the form of the bound in Theorem 4.
Algorithm 1 \textsc{BBScheduler}(X, F, M, \epsilon, \delta, n, p, g)

\begin{algorithmic}
\If{$\delta = 0$}
    \State Let $\gamma \leftarrow g^{\frac{1}{2p+1}} \left( \frac{\log \frac{1}{\epsilon}}{\epsilon n} \right)^{\frac{p}{2p+1}}$
\Else
    \State Let $\gamma \leftarrow g^{\frac{1}{2p+1}} \left( \frac{\log \frac{1}{\epsilon}}{\epsilon n} \right)^{\frac{p}{2p+1}}$
\EndIf
\State Let $i \leftarrow -1$
\For{$t \leftarrow n, n+1, \ldots$}
    \If{$t = (1 + \gamma)^i + 1$}
        \State $i \leftarrow i + 1$
    \If{$\delta = 0$}
        \State Let $\epsilon_i \leftarrow \frac{\gamma^2(i+1)^2}{i+\gamma} \epsilon$
    \Else
        \State Let $\epsilon_i \leftarrow \frac{\gamma^{i+1}}{(1+\gamma)^{i+1}} \epsilon$
    \EndIf
    \State Let $\beta_i \leftarrow \left( \frac{\beta}{1+\gamma} \right)^{i+1}$
    \State Let $\alpha_i \leftarrow g \left( \frac{\log \frac{1}{\epsilon}}{\epsilon(1+\gamma) n} \right)^{i+1}$
    \State Let $y_i \leftarrow M(x_t, \epsilon_i, \alpha_i, \beta_i)$
    \EndIf
\EndFor
\For{$j \leftarrow 1, \ldots, \ell_t$}
    \State Output $y_i(f_{t,j})$
\EndFor
\EndFor
\end{algorithmic}

Note that this algorithm can provide two different types accuracy bounds. If we desire $(\epsilon, 0)$-differential privacy, then the accuracy bounds are slightly weaker, while if we allow $(\epsilon, \delta)$-differential privacy, we can get improved accuracy bounds at the cost of a small loss in privacy. The only differences are how the algorithm sets $\gamma$ and $\epsilon_i$. For a complete proof of Theorem 10, see Appendix A. We present a proof sketch below.

\textit{Proof sketch of Theorem 10} \textsc{BBScheduler} inherits its privacy guarantee from the black box $M$ and the composition properties of differential privacy. When $\delta = 0$, we use Theorem 3 (Basic Composition). When $\delta > 0$, we use Theorem 4 (CDP Composition). These two cases require different settings of $\gamma$ and $\epsilon_i$ for their respective composition theorems to yield the desired privacy guarantee.

To prove the accuracy of \textsc{BBScheduler} we require the following lemma, which bounds the additive error introduced by answering queries that arrive mid-epoch using the slightly outdated database from the end of the previous epoch.

\textbf{Lemma 11.} For any linear query $f$ and databases $x_t$ and $x_\tau$ from a database stream $X$, where $\tau \in [t, (1+\gamma)t]$ for some $\gamma \in (0, 1)$,

$$|x_\tau(f) - x_t(f)| \leq \frac{\gamma}{1+\gamma}.$$

We combine this lemma with the algorithm’s choice of $\gamma_i$ to show that with probability at least $1 - \beta_i$, all mid-epoch queries are answered $\alpha$-accurately with respect to the current database. The final step is to bound the overall failure probability of the algorithm. Taking a union bound over the failure probabilities in each epoch, we complete the proof by showing that $\sum_{i=0}^{\infty} \beta_i \leq \beta$. \hfill $\square$
3.2 Improving Accuracy as Data Accumulate

In the previous section, our accuracy bounds stayed fixed as the database size increased. However, in some applications it is more natural for accuracy bounds to improve as the database grows. For instance, in empirical risk minimization (defined in Section 2.2) the database can be thought of as a set of training examples. As the database grows, we expect to be able to find classifiers with shrinking empirical risk, which implies shrinking generalization error. More generally, when database entries are random samples from a distribution, one would expect accuracy of analysis to improve with more samples.

In this section, we extend our black box scheduler framework to allow for accuracy guarantees that improve as data accumulate. Accuracy improvements over BBSCHEDULER are typically only seen once the database is sufficiently large. We first instantiate our result for empirical risk minimization in Section 3.2.1 and then present the general result in Section 3.2.2.

3.2.1 Application: Empirical Risk Minimization for Growing Databases

In the static setting, an algorithm for empirical risk minimization (ERM) takes in a database $x$ of size $n$, and outputs a classifier from some set $C$ that minimizes a loss function $L$ on the sample data. Increasing the size of the training sample will improve accuracy of the classifier, as measured by excess empirical risk ($\hat{R}(\theta) = \min_{\theta \in C} L(\theta; x) - \min_{\theta \in C} L(\theta; x)$). Given the importance of ERM, it is no surprise that a number of previous works have considered differentially private ERM in the static setting [CMS11, KST12, BST14].

For ERM in the dynamic setting, we want a classifier $y_t$ at every time $t \geq n$ that achieves low empirical risk on the current database, and we want the empirical risk of our classifiers to improve over time, as in the static case. Note that the dynamic variant of the problem is strictly harder because we must produce classifiers at every time step, rather than waiting for sufficiently many new samples to arrive. Releasing classifiers at every time step degrades privacy, and thus requires more noise to be added to preserve the same overall privacy guarantee. Nonetheless, we will compare our private growing algorithm, which simultaneously provides accuracy bounds for every time step from $n$ to infinity, to private static algorithms, which are only run once.

In ERMG, our algorithm for ERM in the dynamic setting, the sole query of interest is the loss function $L$ evaluated on the current database. At each time $t$, ERMG receives a single query $f_t$, where $f_t$ evaluated on the database is $x_t(f_t) = \min_{\theta \in C} L(\theta; x_t)$. The black box outputs $y_t$, which is a classifier from $C$ that can be used to evaluate the single query $y_t(f_t) = L(y_t; x_t)$. Our accuracy guarantee at time $t$ is the difference between $y_t(f_t)$ and $x_t(f_t)$:

$$\alpha_t = L(y_t; x_t) - \min_{\theta \in C} L(\theta; x_t).$$

This expression is identical to the excess empirical risk $\hat{R}(\theta)$ defined in Equation (2.1). Thus accurate answers to queries are equivalent to minimizing empirical risk. Our accuracy bounds are stated in Theorem 12.

**Theorem 12.** Let $c > 0$, and $L$ be a convex loss function that is 1-Lipschitz over some set $C$ with $||C||_2 = 1$. Then for any stream of databases $X$ with points in $\mathbb{R}^d$, ERMG $(X, L, C, \epsilon, \delta, \beta, n)$ is $(\epsilon, \delta)$-differentially private and with probability at least $1 - \beta$ produces classifiers $y_t$ for all $t \geq n$ that for sufficiently large constant $C$ have excess empirical risk bounded by

$$\hat{R}(y_t) \leq C \frac{d \log(1/\beta) \sqrt{\log(1/\delta)}}{\sqrt{c \delta t^{1-c}}}.$$

If $L$ is also $\Delta$-strongly convex,

$$\hat{R}(y_t) \leq C \frac{d^2 \log^2(1/\beta) \log(1/\delta)}{\sqrt{\epsilon \Delta^2 t^{1-c} \delta}}.$$

The results in Theorem 12 all come from instantiating (the more general) Theorem 13 stated in the next section, and the proof is in Appendix A.2. We use the static $(\epsilon, 0)$-differentially private algorithms of
As black boxes. The differing assumptions on \( \mathcal{L} \) allow us to use different \((p, g)\)-black boxes with different input parameters in each case. We compare our growing bounds to these static bounds in Table 1. Since ERMG provides \((\epsilon, \delta)\)-differential privacy, we also include static \((\epsilon, \delta)\)-differential privacy bounds for comparison in Table 1. The static bounds are optimal in \( d, t, \) and \( \epsilon \) up to log factors.

| Assumptions | Static \((\epsilon, 0)\)-DP \cite{BST14} | Static \((\epsilon, \delta)\)-DP \cite{BST14} | Dynamic \((\epsilon, \delta)\)-DP (our results) |
|-------------|---------------------------------|---------------------------------|---------------------------------|
| 1-Lipschitz and \( ||\mathcal{C}||_2 = 1 \) | \( \frac{d \log \frac{1}{\epsilon t}}{\epsilon t} \) | \( \frac{\sqrt{d} \log^2 (t/\delta) \log \frac{1}{\epsilon t}}{\epsilon t} \) | \( \frac{d \sqrt{\log (1/\delta) \log \frac{1}{\epsilon t}}^{1-\epsilon}}{\sqrt{\epsilon t}^{3-\epsilon}} \) |
| ... and \( \Delta \)-strongly convex (implies \( \Delta \leq 2 \)) | \( \frac{d^2 (\log t) \log^2 \frac{1}{\epsilon t}}{\Delta^2 t^2} \) | \( \frac{d \log^2 (t/\delta) \log^2 \frac{1}{\epsilon t}}{\Delta^2 t^2} \) | \( \frac{d^2 \log^2 (1/\delta) \log^2 \frac{1}{\epsilon t}}{\Delta^2 t^2} \) |

Table 1: Comparison of excess empirical risk upper bounds in the static case versus the dynamic case for a database of size \( t \) under differing assumptions on the loss function \( \mathcal{L} \). Database entries are sampled from \( \mathbb{R}^d \), and \( c \) is any positive constant. We ignore leading multiplicative constants and factors of \( \log \log \frac{1}{\beta} \) in the static bounds. As in \cite{BST14}, we assume \( \delta < 1/t \) for simplicity.

Note that the bounds we get for the growing setting have the same dependence on \( \epsilon, \beta, \) and \( \Delta \) and better dependence on \( \delta \). The dependence on \( t \) in our bound is roughly the square root of that in the static bounds. Compared to the static \((\epsilon, 0)\)-differential privacy bounds, our dependence on \( d \) is the same, while the dependence is squared relative to the static \((\epsilon, \delta)\)-differential privacy bounds.

Given that the growing setting is strictly harder than the static setting, it is somewhat surprising that we have no loss in most of the parameters, and only minimal loss in the size of the database \( t \). Thus, for ERM, performance in the static setting largely carries over to the growing setting.

### 3.2.2 A General Black Box Scheduler for Improving Accuracy

In this section we describe the general BBIMPROVER algorithm, which achieves accuracy guarantees in the dynamic setting that improve as the database size grows. The algorithm takes in a private and accurate black box \( \mathcal{M} \), which it re-runs on the current database at every time step. We require the following more general definition of black box to state the privacy and accuracy guarantees of BBIMPROVER.

**Definition 5** (Definition of \((p, p', p'', g)\)-black box). An algorithm \( \mathcal{M}(x, \epsilon, \alpha, \beta, n) \) is a \((p, p', p'', g)\)-black box for a class of linear queries \( \mathcal{F} \) if it is \((\epsilon, 0)\)-differentially private and with probability \( 1 - \beta \) it outputs some \( y : \mathcal{F} \to \mathbb{R} \) such that \( |y(f) - x_n(f)| \leq \alpha \) for every \( f \in \mathcal{F} \) when \( \alpha \geq g \left( \frac{1}{en} \right)^p \log^{p'} n \log^p (1/\beta) \) for some \( g \) that is independent of \( \epsilon, n, \beta \).

The algorithm BBIMPROVER (Algorithm 2) will run the black box \( \mathcal{M} \) after each new data point arrives, starting at time \( t = n \), using time-dependent parameters \( \epsilon_t, \alpha_t, \beta_t \). The output \( y_t \) will be used to answer all queries that arrive at time \( t \).

The following theorem is our main result for BBIMPROVER, which states that the algorithm is differentially private and \((\alpha_t, \beta)\)-accurate for \( \alpha_t \) that decreases inverse polynomially in \( t \). The complete proof is given in Appendix A.

**Theorem 13.** Let \( c > 0 \) and let \( \mathcal{M} \) be a \((p, p', p'', g)\)-black box for query class \( \mathcal{F} \). Then for any database stream \( X \) and stream of linear queries \( F \) over \( \mathcal{F} \), BBIMPROVER\((X, F, \mathcal{M}, \epsilon, \delta, \beta, n, p, p', p'', g, c)\) is \((\epsilon, \delta)\)-

---

\(^7\)To get the static bounds, we use Appendix D of \cite{BST14}, which converts bounds on expected excess empirical risk to high probability bounds.
Algorithm 2 BBImprover$(X, F, \mathcal{M}, \epsilon, \delta, \alpha, \beta, n, p, p', p'', g, c)$

for $t \leftarrow n, n + 1, \ldots$ do
    Let $\epsilon_t \leftarrow \frac{\epsilon}{3\sqrt{\log(1/\delta) t^{2}}}^{1/2}$
    Let $\beta_t \leftarrow \frac{\beta}{2^t}$
    Let $\alpha_t \leftarrow g\left(\frac{1}{t^2}\right)^p \log^{p''} n \log^{p'} \frac{1}{\beta}$
    Let $y_t \leftarrow \mathcal{M}(x_t, \epsilon_t, \alpha_t, \beta_t)$
    for $j \leftarrow 1, \ldots, \ell_t$ do
        Output $y_t(f_{t,j})$
    end for
end for

differentially private for $\epsilon < 1$ and $(\alpha_t)_{t \geq n}$-accurate for sufficiently large constant $C$ and

$$\alpha_t = C g \log^{p'} \left(\frac{1}{\beta} \left(\frac{\sqrt{\log(1/\delta)}}{\sqrt{\epsilon t^{1/2}}}\right)^p\right).$$

The free parameter $c$ in Theorem 13 can be any positive constant, and should be set to an arbitrarily small constant for the algorithm to achieve the best asymptotic performance.

BBImprover does not incur accuracy loss from ignoring new data points mid-epoch as in BBScheduler because it runs $\mathcal{M}$ at every time step. However, this also means that privacy loss will accumulate much faster than in BBScheduler because more computations are being composed. To combat this and achieve overall privacy loss $\epsilon$, each run of $\mathcal{M}$ will have increasingly strict (i.e., smaller) privacy parameter $\epsilon_t$. The additional noise needed to preserve privacy will overpower the improvements in accuracy until the database grows sufficiently large, when the accuracy of BBImprover will surpass the comparable fixed accuracy guarantee of BBScheduler. For any $p > 0$, the guarantees of BBImprover are stronger when $t \gg n^2$. This suggests that an analyst’s choice of algorithm should depend on her starting database size and expectations of data growth.

4 Private Multiplicative Weights for Growing Databases

In this section, we show how to modify the private multiplicative weights (PMW) algorithm for adaptive linear queries [HR10] to handle continuous data growth. The first black box process BBScheduler in the previous section shows that any algorithm can be rerun with appropriate privacy parameters at appropriate points of data growth with minimal loss of accuracy with respect to the intra-epoch data. However, in some settings it may be undesirable to ignore new data for long periods of time, even if the overall accuracy loss is small. Although BBImprover runs the black box algorithm at every step for eventual tighter accuracy bounds, these bounds are inferior until the database grows substantially. We now show how to open the black box and apply these scheduling techniques with a modification of PMW that considers all available data when a query arrives, achieving tight bounds on accuracy as soon as analysis begins and continuing through infinite data growth.

The static PMW algorithm answers an adaptive stream of queries while maintaining a public histogram reflecting the current estimate of the database given all previously answered queries. Critical to the performance of the algorithm is that it categorizes incoming queries as either easy or hard, suffering significant privacy loss only for the hard queries. Hardness is determined with respect to the public histogram: upon receipt of a query for which the histogram provides a significantly different answer than the true database, PMW classifies this as a hard query, and it updates the histogram in a way that moves it closer to a correct answer on that query. The number of hard queries is bounded using a potential argument. Potential is defined as the relative entropy between the database and the public histogram. This quantity is initially bounded, decreases by a substantial amount after every hard query, and never increases.
If we run static PMW on a growing database, the previous potential argument fails because the relative entropy between the database and the public histogram can increase as new data arrive. In the worst case, PMW can learn the database with high accuracy (using many hard queries), and then adversarial data growth can change the composition of the database dramatically, increasing the number of possible hard queries well beyond the bound for the static case. Instead, we modify PMW so that the public histogram updates not only in response to hard queries but also in response to new data arrivals. By treating the new data as coming from a uniform distribution, these latter updates incur no additional privacy loss, and they mitigate the relative entropy increase due to new data. In fact, this modification allows us to suffer only constant loss in accuracy per query relative to the static setting, while maintaining this accuracy through unbounded data growth and accumulating additional query budget during growth.

4.1 $(\epsilon, 0)$-Differentially Private PMWG

Our formal algorithm for PMW for growing databases (PMWG) is given as Algorithm 3 below. We give an overview here to motivate our main results. The algorithm takes as input a database stream $X = \{x_t\}_{t \geq n}$ and an adaptively chosen query stream $F = \{\{f_{t,j}\}_{j=1}^{\ell_t}\}_{t \geq n}$. It also accepts privacy and accuracy parameters $\epsilon, \delta, \alpha$. In this section we restrict to the case where $\delta = 0$; in Section 4.2 we allow $\delta > 0$.

The algorithm maintains a fractional histogram $y$ over $X$, where $y_{t,j}$ denotes the histogram after the $j$th query at time $t$ has been processed. This histogram is initialized to uniform, i.e., $y_{t,0} = 1/|N|$ for all $i \in [N]$. As with static PMW, when a query is deemed hard, our algorithm performs a multiplicative weights update of $y$ with learning rate $\alpha/6$. As an extension of the static case, we also update the weights of $y$ when a new data entry arrives to reflect a data-independent prior belief that data arrive from a uniform distribution. That is, for all $t > n, i \in [N]$,

$$y_{t,0} = \frac{t - 1}{\ell} y_{t-1,\ell} + \frac{1}{\ell N}.$$

It is important to note that a multiplicative weights update depends only on the noisy answer to a hard query as in the static case, and the uniform update only depends on the knowledge that a new entry arrived, so this histogram can be thought of as public.

As in static PMW, we determine hardness using a Numeric Sparse subroutine. We specify a hardness threshold of $T = \frac{\alpha}{6}$, and we additionally specify a function $\xi$ that varies with time and determines how much noise to add to the hardness quantities. Our most general result for $(\epsilon, 0)$-privacy (Theorem 22 in Appendix B.1) considers other noise functions, but for the results stated here, we let $\xi_t = \frac{\alpha}{\log(N)^{1/2}} \cdot t^{1/2}$ for appropriate constant $c$. A query’s hardness is determined by the subroutines after adding Laplace noise with parameter $4/\xi_t$. We present and analyze the required growing database modifications to Numeric Sparse and its subroutines Numeric Above Threshold and Above Threshold in Appendix C; these algorithms may be of independent interest for future work in the design of private algorithms for growing databases.

We now present our main result for PMWG, Theorem 14. We sketch its proof here and give the full proof in Appendix B.1. Whereas the accuracy results for static PMW are parameterized by the total allowed queries $k$, our noise scaling means our algorithm can accommodate more and more queries as new data continue to arrive. Our accuracy result is with respect to a query stream respecting a query budget that increases at each time $t$ by a quantity increasing exponentially with $\sqrt{t}$. This budget is parameterized by time-independent $\kappa \geq 1$, which is somewhat analogous to the total query budget $k$ in static PMW. This theorem tells us that PMWG can accommodate poly($\kappa$) queries on the original database. Since $\kappa$ degrades accuracy logarithmically, this means we can accurately answer exponentially many queries before any new data arrive. In particular, our accuracy bounds are tight with respect to the static setting, and we maintain this accuracy through unbounded data growth, subject to a generous query budget specified by the theorem’s bound on $\sum_{t=n}^{\infty} \ell_t$.

**Theorem 14.** The algorithm $\text{PMWG}(X, F, \epsilon, 0, \alpha, n)$ is $(\epsilon, 0)$-differentially private, and for any time-independent $\kappa \geq 1$ and $\beta > 0$ it is $(\alpha, \beta)$-accurate for any query stream $F$ such that $\sum_{t=n}^{\infty} \ell_t \leq \kappa \sum_{r=n}^{\infty} \exp\left(\frac{\alpha^2 \sqrt{\beta}}{c \log(N)^{1/2}}\right)$. 

14
Algorithm 3 PMWG($X, F, \epsilon, \delta, \alpha, n$)

if $\delta = 0$ then
    Let $\xi_t \leftarrow \frac{\alpha^2 n^{1/2}}{162 \log(Nn)} t^{1/2}$ for $t \geq n$
else
    Let $\xi_t \leftarrow \frac{\alpha n^{1/2}}{48 \log^{1/2}(Nn) \log^{1/2}(1/\delta)} t^{1/2}$ for $t \geq n$
end if

Start NSG($X, \cdot, 2\alpha/3, \{\xi_t\}_{t \geq n}$) $\triangleright$ Initialize subroutine

Let $y_{i,0} \leftarrow 1/N$ for $i \in [N]$ $\triangleright$ Public histogram

Let $h_t \leftarrow 0$ for $t \geq n$ $\triangleright$ Hard query counters

Let $b_t \leftarrow \log N + \log(t-1) + \log \frac{1}{t}$ for $t \geq n + 1$ $\triangleright$ Hard query bounds

for each incoming query $f_{t, j}$ do
    if last query was at time $t' < t$ then
        Let $y_{i,t} \leftarrow \frac{L}{t} y_{i,t'} + \frac{t-t'}{t} \frac{1}{N}$ for $i \in [N]$ $\triangleright$ Uniform update
    end if

    Let $f'_{t,2j-1} \leftarrow f_{t,j} - f_{t,j}(y_{t,j-1})$, $f'_{t,2j} \leftarrow f_{t,j}(y_{t,j-1}) - f_{t,j}$ $\triangleright$ Check hardness

    Receive $a'_{t,2j-1}$, $a'_{t,2j}$ from NSG on $f'_{t,2j-1}, f'_{t,2j}$

    if $a'_{t,2j-1} = \bot$ and $a'_{t,2j} = \bot$ then
        Let $y_{t,j} \leftarrow y_{t,j-1}$
        Output $a_{t,j} \leftarrow f_{t,j}(y_{t,j})$ $\triangleright$ Compute easy query answer
    end else

    Let $h_t \leftarrow h_t + 1$
    if $\sum_{\tau=n}^{t} h_{\tau} > \frac{36}{\alpha} \left( \log N + \sum_{\tau=n+1}^{t} b_{\tau} \right)$ then
        return $\bot$ $\triangleright$ Hard query budget exceeded
    end if

    if $a'_{t,2j-1} \in \mathbb{R}$ then
        Output $a_{t,j} \leftarrow f_{t,j}(y_{t,j-1}) + a'_{t,2j-1}$
    else
        Output $a_{t,j} \leftarrow f_{t,j}(y_{t,j-1}) - a'_{t,2j}$
    end if

    if $a_{t,j} < f_{t,j}(y_{t,j-1})$ then
        Let $r_{t,j} \leftarrow f_{t,j}$
    else
        Let $r_{t,j} \leftarrow 1 - f_{t,j}$
    end if

    Let $y'_{t,j} \leftarrow \exp(-\frac{\alpha}{6} r_{t,j}) y_{t,j-1}$ for $i \in [N]$ $\triangleright$ MW update

    Let $y_{i,t} \leftarrow \frac{y'_{i,t}}{\sum_{i' \in [N]} y_{i',t}}$ for $i \in [N]$ $\triangleright$ MW update
end for
for all \( t \geq n \) and sufficiently large constant \( C \), as long as \( N \geq 3, n \geq 21 \) and
\[
\alpha \geq C \left( \frac{\log(Nn) \log(kn/\beta)}{ne} \right)^{1/3}.
\]

**Proof sketch.** The proof hinges on showing that we do not have to answer too many hard queries, even as the composition of the database changes with new data, potentially increasing the relative entropy between the database and the public histogram. We first show that our new public histogram update rule bounds this relative entropy increase (Lemma 15), and then show that the bound on the number of hard queries does not suffer too much relative to the static setting (Corollary 16).

**Lemma 15.** Let \( x, y, \bar{x}, \bar{y} \in \Delta(\mathcal{X}) \) respectively be databases of size \( t, t, t+1, t+1 \), where \( \bar{x} \) is obtained by adding one entry to \( x \) and \( \bar{y}_i = \frac{t}{t+1} y_i + \frac{1}{(t+1)N} \) for \( i \in [N] \). Then,
\[
\text{RE}(\bar{x}||\bar{y}) - \text{RE}(x||y) \leq \frac{\log N}{t+1} + \frac{\log t}{t+1} + \log(\frac{t+1}{t}).
\]

The corollary below comes from a straightforward modification of the proof on the bound on hard queries in static PMW using the result above.

**Corollary 16.** For a particular run of PMWG, if the Numeric Sparse subroutine returns \( \alpha/3 \)-accurate answers for each query, then the total number of hard queries answered by any time \( t \geq n \) is given by
\[
\sum_{\tau=n}^{t} h_{\tau} \leq \frac{36}{\alpha^2} \left( \frac{\log N}{\tau} + \frac{\log(N-1)}{\tau} + \log(\frac{\tau}{\tau-1}) \right).
\]

With this corollary, we separately prove privacy and accuracy (Theorems 18 and 19) in terms of the noise function \( \xi \), which yield our desired result when instantiated with the \( \xi \) specified by Algorithm 3. As with static PMW, the only privacy leaked is by the Numeric Sparse subroutine, which is differentially private (Theorem 23). The privacy loss depends in the usual ways on the noise parameter, the query sensitivity, and the number of hard queries, although in our setting both the noise parameter and query sensitivity are changing over time.

After the proof Theorem 14 in Appendix B.1, Theorem 22 gives results for a generalization of PMWG specified by Equation (B.5). This generalization leaves a free parameter in the noise function \( \xi \) used by the subroutine, allowing one to trade off between accuracy and a query budget that increases more with time.

### 4.2 \((\epsilon, \delta)\)-Differentially Private PMWG

We remark that we can tighten our accuracy bounds if we allow \((\epsilon, \delta)\)-differential privacy in a manner analogous to the \( \delta > 0 \) relaxation for static PMW. Using CDP composition [BS16] to bound the privacy loss at each time step in the algorithm, we get the following result. As with its \( \delta = 0 \) counterpart, Theorem 17 is also an instantiation of a more general result (Theorem 24) for a version of the algorithm with a free noise parameter, proven in Appendix B.2.

**Theorem 17.** The algorithm PMWG\((X, F, \epsilon, \delta, \alpha, n)\) is \((\epsilon, \delta)\) differentially private for any \( \epsilon \in (0, 1], \delta \in (0, e^{-1}] \), and for any time-independent \( \kappa \geq 1 \) and \( \beta \in (0, 2^{-15/2}) \) it is \((\alpha, \beta)\)-accurate for any query stream \( F \) such that \( \sum_{\tau=n}^{t} \ell_{\tau} \leq \kappa \sum_{\tau=n}^{t} \exp\left(\frac{\alpha^2 + \sqrt{\alpha}}{C \log^{1/2}(Nn) \log^{1/2}(1/\delta)}\right) \) for all \( t \geq n \) as long as \( N \geq 3, n \geq 17 \) and
\[
\alpha \geq C \left( \frac{\log^{1/2}(Nn) \log(kn/\beta) \log^{1/2}(1/\delta)}{ne} \right)^{1/2}.
\]
References

[AS17] Naman Agarwal and Karan Singh. The price of differential privacy for online learning. In Proceedings of the 34th International Conference on Machine Learning, ICML ’17, pages 32–40, 2017.

[BLR08] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC ’08, pages 609–618, 2008.

[BNS+16] Raef Bassily, Kobbi Nissim, Adam D. Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. Algorithmic stability for adaptive data analysis. In Proceedings of the 48th Annual ACM Symposium on Theory of Computing, STOC ’16, pages 1046–1059, 2016.

[BS16] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In Proceedings of the 13th Conference on Theory of Cryptography, TCC ’16, pages 635–658, 2016.

[BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Differentially private empirical risk minimization: Efficient algorithms and tight error bounds. In Proceedings of the 55th Annual Symposium on Foundations of Computer Science, FOCS ’14, pages 464–473, 2014.

[CKN+11] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. “You might also like:” Privacy risks of collaborative filtering. In Proceedings of the 2011 IEEE Symposium on Security and Privacy, SP ’11, pages 231–246, 2011.

[CLN+16] Rachel Cummings, Katrina Ligett, Kobbi Nissim, Aaron Roth, and Zhiwei Steven Wu. Adaptive learning with robust generalization guarantees. In Proceedings of the 29th Annual Conference on Learning Theory, COLT ’16, pages 772–814, 2016.

[CMS11] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. Journal of Machine Learning Research, 12:1069–1109, 2011.

[CPWV16] Rachel Cummings, David M. Pennock, and Jennifer Wortman Vaughan. The possibilities and limitations of private prediction markets. In Proceedings of the 17th ACM Conference on Economics and Computation, EC ’16, pages 143–160, 2016.

[CSS11] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. ACM Transactions on Information and System Security, 14(3):26, 2011.

[DFH+15] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. Science, 349(6248):636–638, 2015.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Proceedings of the 3rd Conference on Theory of Cryptography, TCC ’06, pages 265–284, 2006.

[DNPR10] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC ’10, pages 715–724, 2010.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, 9(34):211–407, 2014.
A Analysis of Blackbox Scheduler

In this appendix we present the proofs that were omitted in Section 3.

A.1 Fixed Accuracy as Data Accumulate

Theorem 10. Let $M$ be a $(p, g)$-black box for query class $F$. Then for any database stream $X$ and stream of linear queries $F$ over $F$, BBSCHEDULE($X, F, M, \epsilon, \delta, \beta, n, p, g$) is $(\epsilon, \delta)$-differentially private for $\epsilon < 1$ and $(\alpha, \beta)$-accurate for sufficiently large constant $C$ and

$$\alpha \geq \begin{cases} Cg \left( \frac{\log(1/\delta)}{\epsilon n} \right)^{\frac{1}{1+p}} & \text{if } \delta = 0 \\ Cg \left( \frac{\sqrt{\log(1/\delta) \log(1/\beta)}}{\epsilon n} \right)^{\frac{1}{1+p}} & \text{if } \delta > 0 \end{cases}$$
Proof. We begin with the privacy guarantees of BBScheduler. When \( \delta = 0 \), BBScheduler runs \( \mathcal{M} \) in each epoch \( i \) with privacy parameter \( \epsilon_i = \frac{\gamma^2(i+1)}{(1+\gamma)^{i+1}} \). Then by Basic Composition (Theorem 3), BBScheduler is \( (\sum_{i=0}^{\infty} \epsilon_i, 0) \)-differentially private, where

\[
\sum_{i=0}^{\infty} \epsilon_i = \frac{\gamma^2}{1+\gamma} \sum_{i=0}^{\infty} \frac{i+1}{(1+\gamma)^{i+1}} = \epsilon.
\]

The sum \( \sum_{i=0}^{\infty} \frac{i+1}{(1+\gamma)^{i+1}} \) converges to \( \frac{1+\gamma}{\gamma^2} \), so BBScheduler is \( (\epsilon, 0) \)-differentially private.

When \( \delta > 0 \), BBScheduler runs \( \mathcal{M} \) with privacy parameter \( \epsilon_i = \frac{\gamma^{1.5}i(i+1)}{(1+\gamma)^{i+1}} \) in each epoch \( i \). By Theorem 4, the total privacy loss is at most \( \frac{1}{2} \sum_{i=0}^{\infty} \epsilon_i^2 + \sqrt{2(\sum_{i=0}^{\infty} \epsilon_i^2) \log(1/\delta)} \). Note that

\[
\sum_{i=0}^{\infty} \epsilon_i^2 = \frac{\epsilon^2}{9 \log(1/\delta)} \frac{\gamma^2}{1+\gamma} \sum_{i=0}^{\infty} \frac{(i+1)^2}{(1+\gamma)^{2(i+1)}} = \frac{\epsilon^2}{9 \log(1/\delta)} \frac{\gamma^2}{1+\gamma} \frac{(1+\gamma)^2(\gamma^2 + 2\gamma + 2)}{\gamma^3(\gamma + 2)^3} \leq \frac{2\epsilon^2}{9 \log(1/\delta)}
\]

where we used the fact that \( \gamma \in (0, 1) \). Then the total privacy loss is at most

\[
\frac{\epsilon^2}{9 \log(1/\delta)} + \frac{2\epsilon}{3 \sqrt{9 \log(1/\delta)}} \sqrt{\log(1/\delta)} \leq \epsilon
\]

since \( \epsilon < 1 \).

To prove the accuracy of BBScheduler we require the following lemma, which bounds the additive error introduced by answering queries that arrive mid-epoch using the slightly outdated database from the end of the previous epoch.

**Lemma 11.** For any linear query \( f \) and databases \( x_i \) and \( x_r \) from a database stream \( X \), where \( \tau \in [t, (1+\gamma)t] \) for some \( \gamma \in (0, 1) \),

\[
|x_r(f) - x_t(f)| \leq \frac{\gamma}{1+\gamma}.
\]

**Proof.** The linear query \( x_t(f) \) can be written in the following form: \( x_t(f) = \frac{1}{\tau} \sum_{i=1}^{N} \tau x_i f_i - \frac{1}{\tau} \sum_{i=1}^{N} \tau x_i f_i \), then since \( x_r(f) - x_t(f) = \frac{1}{\tau} \sum_{i=1}^{N} \tau x_i f_i - \frac{1}{\tau} \sum_{i=1}^{N} \tau x_i f_i \), we have,

\[
x_r(f) - x_t(f) = \frac{(\tau - t) + tx_i}{\tau} - \frac{tx_i}{\tau} \leq \frac{(\tau - t) + tx_i}{\tau} - \frac{tx_i}{\tau} = 1 - \frac{t}{\tau},
\]

\[
x_r(f) - x_t(f) \geq \frac{tx_i}{\tau} - \frac{tx_i}{\tau} \geq t \left( \frac{1}{\tau} - \frac{1}{t} \right) = \frac{t}{\tau} - 1.
\]

The last inequality follows because \( \frac{1}{\tau} - \frac{1}{t} \) and \( tx_i \) are both positive. Thus, \( |x_r(f) - x_t(f)| \leq 1 - \frac{t}{\tau} \). Since \( \tau \in [t, (1+\gamma)t] \), then

\[
1 - \frac{t}{\tau} \leq 1 - \frac{t}{(1+\gamma)t} = 1 - \frac{1}{1+\gamma} = \frac{\gamma}{1+\gamma}.
\]

\[\Box\]

We now continue to prove the accuracy of BBScheduler. Let \( t_i = (1+\gamma)^i n \). Recall that epoch \( i \) is defined as the time interval where \( t \in \{ t_i, t_i + 1, \ldots, t_{i+1} - 1 \} \). Let \( F_i \) denote the set of all queries received during epoch \( i \). All queries \( f \in F_i \) will be answered using \( y_i \), which is computed on database \( x_{t_i} \).
We want to show that \( y_i(f) \) is close to \( x_i(f) \) for all \( f \in F_i \). Since \( y_i \) is the output of \( M(x_i, \epsilon_i, \alpha_i, \beta_i) \), we know that for \( f \in F_i \),
\[
|y_i(f) - x_i(f)| \leq \alpha_i.
\]
By the triangle inequality and Lemma 11 for any \( f \in F_i \),
\[
|y_i(f) - x_i(f)| \leq |y_i(f) - x_i(f)| + |x_i(f) - x_i(f)|
= \alpha_i + \frac{\gamma}{1 + \gamma}.
\]

When \( \delta = 0 \), we have
\[
\alpha_i = g \left( \log \frac{1}{\beta_i} \right)^p = g \left( \frac{(i + 1) \log \frac{1 + \beta}{\beta}}{\gamma^2 (i + 1)(1 + \gamma)^i n} \right)^p = g \left( \frac{(1 + \gamma)^2}{\gamma^2 \epsilon n \log 1 + \frac{1}{\beta}} \right)^p.
\]
Let \( Z = \log \frac{1 + \beta}{\epsilon n} \). Note that since \( \gamma < 1 \), we have \((1 + \gamma) \in (1, 2)\). Then (A.1) becomes
\[
(A.1) \leq \alpha_i + \gamma \leq g Z^p \left( \frac{1}{2} \right)^{-2p} + \gamma.
\]
Since we set \( \gamma = g Z^{1 - 2p} Z^{p^{1 - 2p}} \), we have:
\[
g Z^p \left( \frac{1}{2} \right)^{-2p} + \gamma = g Z^p \left( \frac{1}{2} \right)^{-2p} + g Z^{p^{1 - 2p}} + g Z^{p^{1 - 2p}} = C_1 g Z^{1 - 2p} Z^{p - 2p} + g Z^{p^{1 - 2p}} Z^{p^{1 - 2p}} = C_2 g Z^{p^{1 - 2p}} Z^{p^{1 - 2p}}
\]
where \( C_1 \) and \( C_2 \) are positive absolute constants.

When \( \delta > 0 \), BBScheduler uses a different setting of \( \gamma \) and \( \epsilon_i \). Let \( Z = \frac{3 \sqrt{\log(1/\delta)} \log \frac{1 + \beta}{\beta}}{\epsilon n} \). In this case, we have
\[
\alpha_i = g \left( \log \frac{1}{\beta_i} \right)^p = g \left( \frac{(i + 1) \log \frac{1 + \beta}{\beta}}{\gamma^2 (i + 1)(1 + \gamma)^i n} \right)^p = g Z^p \left( \frac{1 + \gamma}{\gamma} \right)^{1.5p} \leq g Z^p \left( \frac{1}{2} \right)^{1.5p}
\]
Since we set \( \gamma = g Z^{1.5p} Z^{1.5p} \), we have:
\[
(A.1) \leq g Z^p \left( \frac{1}{2} g Z^{1.5p} Z^{1.5p} \right)^{-1.5p} + g Z^{1.5p} Z^{1.5p} \leq C_1 g Z^{1.5p} Z^{1.5p} + g Z^{1.5p} Z^{1.5p} \leq C_2 g Z^{1.5p} Z^{1.5p}
\]
where \( C_1 \) and \( C_2 \) are positive absolute constants.

The final accuracy bound for any \( \delta \geq 0 \) follows by substitution and by noting that \( \log \frac{1 + \beta}{\beta} = O(\log \frac{1}{\beta}) \) since \( \beta \in (0, 1) \). Each of the \( \alpha_i \) bounds holds with probability \( 1 - \beta_i \), so by a union bound, all will hold simultaneously with probability \( 1 - \sum_{i=0}^{\infty} \beta_i \), where,
\[
\sum_{i=0}^{\infty} \beta_i = \frac{\beta}{2n^2} \leq \sum_{i=0}^{\infty} \frac{\beta}{2n^2} \leq \frac{\pi^2}{12} \leq \beta.
\]
Then with probability at least \( 1 - \beta \), all queries are answered with accuracy \( C g Z^{1.5p} Z^{1.5p} \) for \( \delta = 0 \) and \( C g Z^{1.5p} Z^{1.5p} \) for \( \delta > 0 \) for some positive absolute constant \( C \). \(\square\)
A.2 Improving Accuracy as Data Accumulate

**Theorem 13.** Let $c > 0$ and let $\mathcal{M}$ be a $(p, p', p'', g)$-black box for query class $\mathcal{F}$. Then for any database stream $X$ and stream of linear queries $F$ over $\mathcal{F}$, BBIMPROVER($X, F, \mathcal{M}, \epsilon, \delta, \beta, n, p, p', p'', g, c$) is $(\epsilon, \delta)$-differentially private for $\epsilon < 1$ and $\{(\alpha_t)_{t \geq n}, \beta_t\}$-accurate for sufficiently large constant $C$ and

$$
\alpha_t = C g \log (p'')^p \left( \frac{\sqrt{\log(1/\delta)}}{\sqrt{c \epsilon t^{2-2c}}} \right).
$$

**Proof.** We start with the privacy guarantee. BBIMPROVER runs $\mathcal{M}$ at each time $t$ with privacy parameters $\epsilon_t = \frac{\epsilon}{\sqrt{1+c}}$. By Theorem 4 the total privacy loss is at most $\frac{1}{2} \sum_{t=n}^\infty \epsilon_t + \sqrt{2(\sum_{t=n}^\infty \epsilon_t^2)} \log(1/\delta)$. Note:

$$
\sum_{t=n}^\infty \epsilon_t^2 = \frac{ce^2}{9 \log(1/\delta)} \sum_{t=n}^\infty \frac{1}{t^{1+2c}} \leq \frac{ce^2}{9 \log(1/\delta)} \frac{1}{cn^{2c}} \leq \frac{e^2}{9 \log(1/\delta)},
$$

since $n \geq 1$ and $c > 0$. Then the privacy loss is at most:

$$
\frac{1}{18 \log(1/\delta)} \epsilon^2 + \frac{2t}{3 \sqrt{\log(1/\delta)}} \sqrt{\log(1/\delta)} \leq \epsilon
$$

since $\epsilon < 1$.

We next prove the accuracy of BBIMPROVER. For each time $t$, $\mathcal{M}(x_t, \epsilon_t, \alpha_t, \beta_t)$ is $\left( g \left( \frac{1}{\epsilon_t \beta_t} \right)^p \log p'' \ t \log p' \ rac{1}{\beta_t} \right)$-accurate. We simply plug in $\epsilon_t$ and $\beta_t$ to get our accuracy bound at time $t$:

$$
\alpha_t = g \left( \frac{1}{\epsilon_t} \right)^p \log p'' \ t \log p' \ \frac{1}{\beta_t}
$$

$$
= g \left( \frac{3 \sqrt{\log(1/\delta)} t^{1+c} + \epsilon}{\sqrt{c \epsilon t^{2-c}}} \right)^p \log p'' \ t \log p' \ \frac{2t^2}{\beta}
$$

$$
\leq C_1 g \left( \frac{\sqrt{\log(1/\delta)}}{\sqrt{c \epsilon t^{2-c}}} \right)^p \log p'' \ t \log p' \ \frac{1}{\beta}
$$

$$
\leq C_2 g \left( \frac{\sqrt{\log(1/\delta)}}{\sqrt{c \epsilon t^{2-2c}}} \right)^p \log p'' \ \frac{1}{\beta},
$$

for positive constants $C_1$ and $C_2$. The last line holds since $\log p'' + p' t = o(t^c)$ for any positive constants $c, p'$, and $p''$.

The accuracy of $\mathcal{M}(x_t, \epsilon_t, \beta_t)$ at time $t$ holds with probability $1 - \beta_t$. By a union bound, all accuracy guarantees will be satisfied with probability $1 - \sum_{t=n}^\infty \beta_t$, where,

$$
\sum_{t=n}^\infty \beta_t = \frac{\beta}{2t^2} \leq \frac{\pi^2}{12} \beta \leq \beta.
$$

\qed

B Analysis of PMW for Growing Databases

We first present the modification of PMW for growing databases described in Section 4 and presented formally in Algorithm 3. We separately prove privacy and accuracy in terms of the internal noise function $\xi$, which depends on the parameters of the algorithm. We then instantiate these theorems with our particular choice of $\xi$ to prove Theorem 14 from the body, which gives our accuracy bound for the $(\epsilon, 0)$-differentially private version of the algorithm. In Equation (B.5), we describe how to generalize our algorithm by adding a parameter for the noise function, which allows us to trade accuracy for a larger query budget. Finally, we use CDP to give our $(\epsilon, \delta)$-results, yielding Theorem 17.
B.1 \((\epsilon, 0)\)-Differentially Private PMWG

\((\epsilon, 0)\)-privacy of PMWG follows from our analysis of the NSG subroutine (Appendix C.3) as well as Lemma 15 bounding the entropy increase due to new data and Corollary 16 bounding the number of hard queries received by any given time. We first prove this lemma and then use it with its corollary to prove privacy of PMWG in Theorem 18.

**Lemma 15.** Let \(x, y, \bar{x}, \bar{y} \in \Delta(\mathcal{X})\) respectively be databases of size \(t, t, t+1, t+1\), where \(\bar{x}\) is obtained by adding one entry to \(x\) and \(\bar{y}^i = \frac{\bar{y}^i}{t+1} y^i + \frac{1}{t+1} x^i\) for \(i \in [\mathcal{N}]\). Then,

\[
\text{RE}(\bar{x}||\bar{y}) - \text{RE}(x||y) \leq \frac{\log N}{t+1} + \frac{\log t}{t+1} + \log(t + 1).
\]

**Proof.** We partition indices \(i \in [\mathcal{N}]\) into two sets \(L, H\), where \(i \in L\) whenever \(y^i \leq \frac{1}{tN}\). For each \(S \subseteq [\mathcal{N}]\), use \(x_S\) to denote \(\sum_{i \in S} x^i\). Then, by \(\bar{y}^i \geq \frac{t}{t+1} y^i\) for all \(i\),

\[
\sum_{i \in L} (\bar{x}^i \log(1/\bar{y}^i) - x^i \log(1/y^i)) \leq \sum_{i \in L} (\bar{x}^i \log(N(t + 1)) - x^i \log(1/y^i)) \\
\quad \leq \sum_{i \in L} \left( \bar{x}^i \log(N(t + 1)) - x^i \log(tN) \right) \\
\quad = \sum_{i \in L} \left( \bar{x}^i \log(Nt) + \bar{x}^i \log\left(\frac{t+1}{t}\right) - x^i \log(tN) \right) \\
\quad = \sum_{i \in L} \left( \bar{x}^i - x^i \right) \log(Nt) + \bar{x}_L \log\left(\frac{t+1}{t}\right) \\
\quad \leq \sum_{i \in L} \max\{\bar{x}^i - x^i, 0\} \log(Nt) + \bar{x}_L \log\left(\frac{t+1}{t}\right)
\]

The last inequality is by ignoring the term \(i \in L\) with \(\bar{x}^i < x^i\). Next, we use \(\bar{y}^i \geq \frac{t}{t+1} y^i\) to get

\[
\sum_{i \in H} (\bar{x}^i \log(1/\bar{y}^i) - x^i \log(1/y^i)) \leq \sum_{i \in H} \left( \bar{x}^i \log\left(\frac{t+1}{t}\right) \log(1/y^i) - x^i \log(1/y^i) \right) \\
\quad = \sum_{i \in H} \left( \bar{x}^i \log(1/y^i) - x^i \log(1/y^i) \right) + \bar{x}^i \log\left(\frac{t+1}{t}\right) \\
\quad = \sum_{i \in H} \left[ (\bar{x}^i - x^i) \log(1/y^i) \right] + \bar{x}_H \log\left(\frac{t+1}{t}\right) \\
\quad \leq \sum_{i \in H} \max\{\bar{x}^i - x^i, 0\} \log(1/y^i) + \bar{x}_H \log\left(\frac{t+1}{t}\right) \\
\quad \leq \sum_{i \in H} \max\{\bar{x}^i - x^i, 0\} \log(Nt) + \bar{x}_H \log\left(\frac{t+1}{t}\right)
\]

The second inequality is by ignoring the term \(i \in H\) with \(\bar{x}^i < x^i\). Combining two bounds on \(L, H\) gives

\[
\sum_{i \in \mathcal{X}} (\bar{x}^i \log(1/\bar{y}^i) - x^i \log(1/y^i)) \leq \sum_{i \in \mathcal{X}} \max\{\bar{x}^i - x^i, 0\} \log(Nt) + \log\left(\frac{t+1}{t}\right)
\]

Since there is at most one index \(i \in [\mathcal{N}]\) such that \(\bar{x}^i - x^i \geq 0\) (the index of newly added data entry), and for that term we have \(\bar{x}^i - x^i = \frac{t}{t+1} + \frac{1}{t+1} x^i - x^i \leq \frac{t}{t+1}\), we have

\[
\sum_{i \in \mathcal{X}} (\bar{x}^i \log(1/\bar{y}^i) - x^i \log(1/y^i)) \leq \frac{1}{t+1} \log(Nt) + \log\left(\frac{t+1}{t}\right)
\]
The following corollary is an immediate extension of the analysis for static PMW using the above lemma:

**Corollary 16.** For a particular run of PMWG, if the Numeric Sparse subroutine returns $\alpha/3$-accurate answers for each query, then the total number of hard queries answered by any time $t \geq n$ is given by

$$
\sum_{\tau=n}^{t} h_\tau \leq \frac{36}{\alpha^2} \left( \log N + \sum_{\tau=n+1}^{t} \frac{\log(N)}{\tau} + \frac{\log(\tau-1)}{\tau} + \log\left(\frac{\tau}{\tau-1}\right) \right).
$$

**Theorem 18 (PMWG Privacy).** PMWG($X, F, \epsilon, 0, \alpha, n$) is $(\epsilon, 0)$-DP for $\xi$ as defined by the algorithm and

$$
\epsilon = \left(1 + \frac{81}{2\alpha^2} \log N\right) \xi_n \Delta_n + \frac{81}{2\alpha^2} \sum_{t=n+1}^{\infty} \left( \frac{\log(N)}{t} + \frac{\log(t-1)}{t} + \log\left(\frac{t}{t-1}\right) \right) \xi_t \Delta_t
$$

**(B.1)**

**Proof.** The uniform and MW updates to the public histogram do not use any information about $\xi$, and because $\xi_t \Delta_t$ is non-increasing, we may upper bound $\sum_{t=n}^{\infty} h_t \xi_t \Delta_t$ by setting $h_t = \frac{36}{\alpha^2} b_t$. Hence,

$$
\epsilon \leq \left(1 + \frac{81}{2\alpha^2} \log N\right) \xi_n \Delta_n + \frac{81}{2\alpha^2} \sum_{t=n+1}^{\infty} \left( \frac{\log(N)}{t} + \frac{\log(t-1)}{t} + \log\left(\frac{t}{t-1}\right) \right) \xi_t \Delta_t
$$

Next we show the accuracy of PMWG.

**Theorem 19 (PMWG Accuracy).** Let $k : \{n, n+1, \ldots\} \to \mathbb{R}$. On query stream $F$ such that $\sum_{\tau=n}^{t} k_{\tau} \leq \sum_{\tau=n}^{t} k_{\tau}$, PMWG($X, F, \epsilon, 0, \alpha, n$) outputs answers such that $|f_{t,j}(D_{t}) - a_{t,j}| \leq \alpha$ for every query $f_{t,j}$ except with probability

$$
\beta \leq \exp\left(-\frac{\alpha \xi_n}{24}\right) + 3 \sum_{t \geq n} k_t \exp\left(-\frac{\alpha \xi_t}{24}\right)
$$

**(B.2)**

**Proof.** By the exact same proof in [DR14]: PMWG’s $\alpha$-accuracy follows if NSG returns answers that are $\alpha/3$-accurate. Hence, we can take the $\beta$ from the NSG accuracy analysis in Theorem [51].

These privacy and accuracy results are in terms of noise scaling function $\xi$. The $\xi$ enforced by the algorithm is chosen to get good privacy and accuracy bounds simultaneously (Theorem [14]). Specifically, we need to ensure that the $\epsilon$ and $\beta$ bounds in (B.1) and (B.2) converge. The dominating term in the hard query bound $b_t$ in the algorithm is $\log(t-1)/t$, so to guarantee convergence of our $\epsilon$ bound, we need

$$
\sum_{t \geq n+1} \frac{\log(t-1)\xi_t \Delta_t}{t} = \sum_{t \geq n+1} \frac{\log(t-1)\xi_t}{t^2} \leq \int_{t=n}^{\infty} \frac{\log(t-1)\xi_t}{t^2} dt,
$$

so we may pick $\xi = O(t^{1-c})$ for any $c > 0$. To guarantee convergence of our $\beta$ bound and allow $\alpha$ to degrade logarithmically in the number of queries $\ell_t$, we want $\sum_{t \geq n} k_t \exp\left(-\frac{\alpha \xi_t}{24}\right)$ to converge exponentially quickly in $\alpha$. For example, we may pick query budget $k_t = O(\exp(\frac{\alpha \xi_t}{24}))$ and $\xi_t = \Omega(t^{c})$ for any $c > 0$. To get these conditions simultaneously, Algorithm [3] picks $\xi_t = ct^{1/2}$ for some time-independent quantity $c > 0$, and
we guarantee accuracy for query streams respective query budget $k_t = \kappa \exp(\frac{2\alpha^{1/2}}{48})$ for some constant $c, \kappa$. Other reasonable choices of $\xi_t, k_t$ are possible; we discuss these after proving our main result for Algorithm 3.

We prove the result below without suppressing constants in the query budget and the $\alpha$ bound, i.e., we prove that $\sum_{t=n}^{\tau} \ell_t \leq \kappa \sum_{t=n}^{\tau} \exp\left(\frac{\alpha^2 \sqrt[1/2]{\tau}}{2 \log(\sqrt{\tau})}\right)$ and $\alpha \geq \left(\frac{8262 \log(Nn) \log(192n/\beta)}{n \epsilon}\right)^{1/3}$ suffice for accuracy.

Note that with this choice of $\alpha$, the query budget grows exponentially with $\sqrt{\tau}$ since $\kappa \sum_{t=n}^{\tau} \exp\left(\frac{\alpha^2 \sqrt[1/2]{\tau}}{2 \log(\sqrt{\tau})}\right) \geq \kappa \sum_{t=n}^{\tau} \left(\frac{192n}{\beta}\right)^{\frac{22}{3}} \sqrt{\tau}$.

**Theorem 14.** The algorithm PMWG($X, F, \epsilon, 0, \alpha, n$) is ($\epsilon, 0$)-differentially private, and for any time-independent $\kappa \geq 1$ and $\beta > 0$ it is ($\alpha, \beta$)-accurate for any query stream $F$ such that $\sum_{t=n}^{\tau} \ell_t \leq \kappa \sum_{t=n}^{\tau} \exp\left(\frac{\alpha^2 \sqrt[1/2]{\tau}}{2 \log(\sqrt{\tau})}\right)$ for all $t \geq n$ and sufficiently large constant $C$, as long as $N \geq 3, n \geq 21$ and

$$\alpha \geq C \left(\frac{\log(Nn) \log(n/\beta)}{n \epsilon}\right)^{1/3}.$$

**Proof.** Our main result is an instantiation of the more general results in Theorems 18 and 19. In what follows, let $\xi_t = ct^{1/2}$ for some time-independent $c > 0$. Applying Theorem 18 the privacy loss of PMWG is

$$\epsilon' = \left(1 + \frac{81}{2\alpha^2} \log N\right) \frac{c}{\alpha^2} \sqrt[1/2]{n} \sum_{t=n+1}^{\infty} \left(\frac{\log(N)}{t} + \frac{\log(t-1)}{t} + \frac{\log(t)}{t^2} + \frac{1}{t^2} \right) \frac{c}{\alpha^2} \sqrt[1/2]{n} t^{-1/2}$$

$$\leq \left(\frac{81}{\alpha^2} \log N\right) \frac{c}{\alpha^2} \sqrt[1/2]{n} \sum_{t=n+1}^{\infty} \left(\frac{\log(N)}{t^3/2} + \frac{\log(t)}{t^{3/2}} + \frac{1}{t^3/2} \right) dt$$

$$= \frac{81c}{\alpha^2} \log N \frac{c}{\alpha^2} \sqrt[1/2]{n} \left[ \frac{2 \log N}{t^{1/2}} - \frac{2 \log(t)}{t^{1/2}} - \frac{2}{t^{1/2}} \right] \right]_{t=n}^{\infty}$$

$$= \frac{81c}{\alpha^2} \left(\frac{2 \log N}{n^{1/2}} + \frac{2 \log(n)}{n^{1/2}} + \frac{1}{(n-1)^{1/2}} \right)$$

$$\leq \frac{81c}{\alpha^2} \frac{2 \log(N + \log(n))}{n^{1/2}},$$

using $\log(t/(t-1))t^{-1/2} = \log(1 + \frac{1}{t-1})t^{-1/2} \leq \frac{1}{t-1}(t-1)^{-1/2}$ for the first inequality and $n \geq 21$ for the last inequality. Finally, setting $c = \frac{24 \sqrt[1/2]{n}}{2 \log(\sqrt{Nn})} \epsilon$ gives $\epsilon' = \epsilon$.

Applying Theorem 19 PMWG is ($\alpha, \beta'$)-accurate for

$$\beta' = \exp\left(-\frac{\alpha c \sqrt[1/2]{n}}{24}\right) + 3\kappa \sum_{t=n}^{\infty} \exp\left(-\frac{\alpha c t^{1/2}}{48}\right) \exp\left(-\frac{\alpha c t^{1/2}}{24}\right)$$

$$\leq \exp\left(-\frac{\alpha c t^{1/2}}{24}\right) + 3\kappa \left[ \exp\left(-\frac{\alpha c t^{1/2}}{48}\right) \right]_{t=n-1}^{\infty}$$

$$= \exp\left(-\frac{\alpha c t^{1/2}}{24}\right) + 3\kappa \left[ \frac{96 (\alpha c \sqrt[1/2]{n} + 48) \exp\left(-\frac{\alpha c \sqrt[1/2]{n}}{48}\right)}{\alpha^2 c^2} \right]_{t=n-1}^{\infty}$$

$$= \exp\left(-\frac{\alpha c t^{1/2}}{24}\right) + 288\kappa \frac{(\alpha c \sqrt[1/2]{n} - 1 + 48)}{\alpha^2 c^2} \frac{\exp\left(-\frac{\alpha c \sqrt[1/2]{n} - 1}{48}\right)}{\alpha^2 c^2}$$

To get $\beta' \leq \beta$, we can require $\exp\left(-\frac{\alpha c t^{1/2}}{24}\right) \leq \beta/2$ and $288\kappa \frac{(\alpha c \sqrt[1/2]{n} - 1 + 48)}{\alpha^2 c^2} \frac{\exp\left(-\frac{\alpha c \sqrt[1/2]{n} - 1}{48}\right)}{\alpha^2 c^2} \leq \beta/2$. The first is equivalent to

$$\frac{\alpha c t^{1/2}}{24} \geq \log(2/\beta) \iff \alpha c \geq \frac{24 \log(2/\beta)}{n^{1/2}} \iff \alpha \geq \left(\frac{3888 \log(Nn) \log(2/\beta)}{n \epsilon}\right)^{1/3} \quad (B.3)$$
We assume that $\alpha$ satisfies (B.3) before proceeding. Secondly, (B.3) gives $\alpha c \geq \frac{24 \log(2/\beta)}{n^{1/2}}$, which implies

$$\frac{(a c \sqrt{n-1} + 48)}{a^2 c^2} \leq \frac{n^{1/2} \sqrt{n-1}}{24 \log(2/\beta)} + \frac{n}{12 \log^2(2/\beta)} \leq \frac{n(\log(2/\beta) + 2)}{24 \log^2(2/\beta)} \leq \frac{n(2 + \log 2)}{24 \log^2(2)} < \frac{n}{3}$$

Hence, it’s enough to require $96 n \exp(-\frac{a c \sqrt{n-1}}{48}) \leq \beta/2$. This is equivalent to

$$\frac{a c \sqrt{n-1} - 1 + 48}{a^2 c^2} \leq \frac{126 \log(Nn)}{\alpha^2 q n^2} \iff \alpha \geq \left(\frac{8262 \log(Nn) \log(192 \kappa n/\beta)}{n^2 q n^2}\right)^{1/3}$$ (B.4)

(B.4) is a stronger bound than (B.3) for $\kappa \geq 1$.

B.1.1 $(\epsilon, 0)$-Differential Privacy for PMWG with Different Noise Functions

We can still achieve good privacy and accuracy for other choices of $\xi$ and query budget. In this section we consider a generalization of PMWG that include an additional parameter $p$. The only difference between this generalization and Algorithm 3 is that the modified version defines the noise function as

$$\xi_t = \begin{cases} \alpha^2 (1-p)^2 n^1 \epsilon t_p & \text{if } \delta = 0 \\ \frac{126 \log(Nn)}{\alpha (1-p) n^{1-p}} \epsilon t_p & \text{if } \delta > 0 \end{cases}$$ (B.5)

Theorem 22 shows that this choice yields an accuracy bound of

$$\alpha \geq \left(\frac{6426 \log(Nn) \log(144 \kappa n/\beta)}{(1-p)^2 n^2 q n^2}\right)^{1/3}.$$ 

This is comparable to accuracy of static PMW, $\alpha_{\text{static}} = \Theta\left(\frac{\log N \log(k/\beta)}{n^2 q n^2}\right)^{1/3}$, and it yields query budget

$$\kappa \sum_{t=n}^{t} \exp\left(\frac{\alpha^3 (1-p)^2 n^1 \epsilon t_{p_p}}{6048 \log(Nn)}\right) \geq \kappa \sum_{t=n}^{t} \left(\frac{144 \kappa n}{\beta}\right)^{\frac{126 \log(Nn)}{\alpha^2 q n^2}}.$$ (B.6)

These bounds tell us that as $p$ approaches 1, the query budget approaches exponential in $t$, but accuracy suffers proportionally to $(1-p)^{-2/3}$. Therefore, we suffer only a constant loss in accuracy as long as $n$ is bounded polynomially in $N$ and $k/\beta$. Note that, however, our query budget in the growing setting allows a generous additional number of queries to be asked upon each arrival of new data entry.

Before proving our general result (Theorem 22), we first present two useful lemmas. The first bounds privacy loss for $\xi$ of our more general form.

**Lemma 20.** Assuming $n \geq 5, N \geq 3$, the privacy loss associated with Algorithm 3 using $\xi_t$ such that $\xi_t \Delta_t = ct^{-q}$ for some constant $c$ independent of $t$ and $1 \geq q > 0$ is $\epsilon' \leq \frac{126c \log(Nn)}{\alpha^2 q n^2}$. 

---

25
Lemma 21. Let \( \epsilon \geq 0 \) and \( \beta < 1 \) that for any \( s, x \) We have

\[ \epsilon = \left( 1 + \frac{81}{2\alpha^2} \log N \right) \xi_n \Delta_n + \frac{81}{2\alpha^2} \sum_{t=n+1}^{\infty} \left( \frac{\log(N)}{t} + \frac{\log(t-1)}{t} + \log\left(\frac{t}{t-1}\right) \right) \xi_t \Delta_t \]

\[ = \left( 1 + \frac{81}{2\alpha^2} \log N \right) cn^{-q} + \frac{81c}{2\alpha^2} \sum_{t=n+1}^{\infty} \left( \frac{\log(N)}{t} + \frac{\log(t-1)}{t} + \log\left(\frac{t}{t-1}\right) \right) ct^{-q} \]

\[ \leq \left( \frac{81}{\alpha^2} \log N \right) cn^{-q} + \frac{81c}{2\alpha^2} \int_{t=n}^{\infty} \left( t^{-q-1} \log(N) + t^{-q-1} \log(t) + (t-1)^{-q-1} \right) dt \]

\[ = \frac{81c \log N}{\alpha^2} n^{-q} + \frac{81c}{2\alpha^2} \left[ \frac{\log(N)n^{-q} + \log(n)n^{-q} + 2n^{-q} + n^{-q}}{q^2} \right] \]

\[ \leq \frac{81c \log N}{\alpha^2} n^{-q} + \frac{81c}{2\alpha^2} \left( 2 \log N + \frac{\log(n) + \log(n) + 3}{q^2} \right) \]

\[ \leq \frac{81c n^{-q}}{2\alpha^2} \left( 3 \log(Nn) \right) \]

\[ \leq \frac{126c \log(Nn)}{\alpha^2 q^2 n^q} \]

where we use the fact that \( (n-1)^{-q} \leq 2n^{-q} \) and \( \log N \leq \frac{\log(N)}{q} \) for \( n \geq 2, 0 \leq q \leq 1 \), and that \( 3 \leq 2 \log n \) for \( n \geq 5 \).

\[ \square \]

We also know that for \( \xi_t = \Omega(t^p) \) for some \( p > 0 \), \( \int_{t=n-1}^{\infty} \exp\left(-\frac{c \xi_t}{48}\right) dt = P_{p,c\alpha}(n-1) \exp\left(-\frac{c \xi_{n-1}}{48}\right) \) for some polynomial \( P_{p,c\alpha}(n) \) dependent on \( p, c\alpha \). Stating the exact bound for this integral, however, involves approximating an upper incomplete gamma function. In the following lemma, which will be used to bound the probability of failure for our algorithm with these modified parameters, we therefore require \( p \geq 1/4 \), noting that we could better optimize constants with smaller \( p \).

Lemma 21. Let \( 1 \geq p \geq 1/4, n \geq 17 \) and \( c, \alpha \) be constants independent of \( t \) such that \( \log(n) \geq 24 \log(2/\beta) \) and \( \beta < 2^{-15/2} \). Then

\[ \int_{t=n-1}^{\infty} \exp\left(-\frac{c \alpha t^p}{48}\right) dt \leq \frac{6n e^{-\frac{c \alpha n^p}{51}}}{p} \]

Proof. We have

\[ \int_{t=n-1}^{\infty} \exp\left(-\frac{c \alpha t^p}{48}\right) dt = \frac{\Gamma\left(\frac{1}{p}, \frac{c \alpha}{48}(n-1)^p\right)}{p\left(\frac{c \alpha}{48}\right)^{\frac{1}{p}}} \]

where \( \Gamma(s, x) = \int_{x}^{\infty} t^{s-1} e^{-t} dt \) is an upper incomplete gamma function. We now use the bound in\footnote{Proposition 10 from the extended version http://www.maths.lancs.ac.uk/jameson/gammaintc.pdf} that for any \( a > 1, e^x > 2^a \), we have \( \Gamma(a, x) \leq 2^a x^{a-1} e^{-x} \). With \( n \geq 17, \frac{c \alpha}{48}(n-1)^p \geq \frac{\log(2/\beta)}{17} \). Choosing \( \beta < 2^{-15/2} \) gives \( \exp\left(\frac{8\log(2/\beta)}{17}\right) \geq 2^4 \geq 2^{1/2} \), so we can apply the bound

\[ \Gamma\left(\frac{1}{p}, \frac{c \alpha}{48}(n-1)^p\right) \leq \Gamma\left(\frac{1}{p}, \frac{\log(n)}{51}\right) \leq 2^{1/p} \left(\frac{\log(n)}{51}\right)^{\frac{1}{p}-1} e^{-\frac{c \alpha n^p}{51}} \]
Therefore,
\[
\Gamma \left( \frac{1}{p}, \frac{\epsilon}{160} (n - 1)^p \right) \leq \frac{51(32/17)^{1/p} n e^{-\frac{\epsilon n}{32}}}{pc\alpha_48 (n - 1)} \leq \frac{17(32/17)^{1/p} n e^{-\frac{\epsilon n}{32}}}{8p \log(2/\beta)} \\
\leq \frac{6 n e^{-\frac{\epsilon n}{32}}}{p}
\]

where we use \( p \geq 1/4 \) and \( \beta < 2^{-15/2} \) to get the last inequality.

Finally, we give our overall privacy and accuracy guarantees of our modified algorithm.

**Theorem 22** (Generalized-Noise PMWG \( \epsilon \)-DP Result). Let \( p \in [1/4, 1) \). The algorithm PMWG(\( X, F, \epsilon, 0, \alpha, n, p \)) that generalizes Algorithm 3 as in B.5 is \( (\epsilon, 0) \)-differentially private, and for any time-independent \( \kappa \geq 1 \) and \( \beta \in (0, 2^{-15/2}) \) it is \( (\alpha, \beta') \)-accurate for any query stream \( F \) such that \( \sum_{t=n}^\tau \ell_t \leq \kappa \sum_{t=n}^\tau \exp(\frac{\alpha(1-p)^2n^{1-p}}{6048 \log(2Nn)}) \) for all \( t \geq n \) as long as \( \beta' < 2^{-15/2} \).

**Proof.** Applying Theorem 18 and Lemma 20, the privacy loss of PMWG is
\[
\epsilon' \leq \frac{126 c \log(Nn)}{\alpha^2 q^2 n^2}
\]
where \( q = 1 - p > 0 \). Setting \( c = \frac{\alpha^2(1-p)^2n^{1-p}}{126 \log(Nn)} \) gives \( \epsilon' = \epsilon \).

Applying Theorem 19, PMWG is \( (\alpha, \beta') \)-accurate for
\[
\beta' \leq \exp(-\frac{c \alpha n}{24}) + 3\kappa \int_{t=n-1}^\infty \exp(-\frac{c \alpha t}{48})dt
\]
Again, we require the first term to be at most \( \beta/2 \):
\[
\frac{c \alpha n}{24} \geq \log(2/\beta) \iff \frac{(1-p)^2n^{1-p}}{126 \log(Nn)} \geq 24 \log(2/\beta) \iff \alpha \geq \left( \frac{3024 \log(Nn) \log(2/\beta)}{(1-p)^2 n \epsilon} \right)^{1/3} \tag{B.8}
\]
We assume that \( \alpha \) satisfies this requirement before proceeding. Now we apply Lemma 21
\[
\int_{t=n-1}^\infty \exp(-\frac{c \alpha t}{48})dt \leq \frac{6 n e^{-\frac{\epsilon n}{32}}}{p} \leq 24 n e^{-\frac{\epsilon n}{32}}
\]
So it’s enough to require \( 3\kappa \cdot 24 n e^{-\frac{\epsilon n}{32}} \leq \beta/2 \). This is equivalent to
\[
\frac{c \alpha n}{51} \geq \log \left( \frac{144 \kappa n}{\beta} \right) \iff \frac{(1-p)^2n^{1-p}}{126 \log(Nn)} \geq 51 \log \left( \frac{144 \kappa n}{\beta} \right) \tag{B.9}
\]
\[
\iff \alpha \geq \left( \frac{6426 \log(Nn) \log(144 \kappa n/\beta)}{(1-p)^2 n \epsilon} \right)^{1/3} \tag{B.10}
\]
Clearly the second requirement is stronger than the first for \( \kappa \geq 1 \).
\[ \varepsilon' := \left( 1 + \frac{585}{16\alpha^2}\log N \right) (\xi_n \Delta_n)^2 + \frac{585}{16\alpha^2} \sum_{t=n+1}^{\infty} \left( \frac{\log(N)}{t} + \frac{\log(t-1)}{t} + \frac{\log(t)}{2(t-1)} \right) (\xi_t \Delta_t)^2 \]  

(B.11)

We now state how to upper bound this sum, which is in a similar form to Lemma 20.

**Lemma 23.** Let \( \xi_i \Delta_i = ct^{-q} \) for some constant \( c \) independent of \( t \) and \( 1 \geq q > 0 \). Then, for \( n \geq 3, N \geq 3 \), and \( \varepsilon' \) as in Equation (B.11) we have \( \sqrt{\varepsilon'} \leq \frac{12c^{n-2q}}{\alpha q} (\log(N))^{1/2} \).

**Proof.** By a similar calculation as in Lemma 20 but with \( c^2n^{-2q} \) in place of \( cn^{-q} \),

\[
\varepsilon' \leq \frac{c^2n^{-2q}}{\alpha^2} \left( \frac{585}{8} \log N \right) n^{-2q} + \frac{585c^2}{16\alpha^2} \left( \frac{2\log(N) + 2\log(n) + 5}{4q^2} \right) n^{-2q} \\
\leq \frac{c^2n^{-2q}}{\alpha^2} \left( \frac{585}{8} \log N \right) n^{-2q} + \frac{585}{64} \frac{2\log(N) + 10\log(n)}{q^2} \\
= \frac{c^2n^{-2q}}{\alpha^2} \left( \frac{2925 \log(N)}{32q^2} \right)
\]

where we use the bounds \( 5 \leq 8 \log(n) \) and \( \log N \leq \frac{\log N}{q^2} \). The result now follows. \( \square \)

With the bound by Lemma 23 and Theorem 4 we achieve an \((\epsilon, \delta)\)-differentially private version of PMWG. Note that Theorem 17 instantiates this result with \( p = 1/2 \).

**Theorem 24** (Generalized-Noise PMWG \((\epsilon, \delta)\)-DP Result). Let \( p \in [1/4, 1), \delta \in (0, e^{-1}) \). The algorithm PMWG\((X, F, \epsilon, \delta, \alpha, n, p)\) is \((\epsilon, \delta)\)-differentially private, and for any time-independent \( \kappa \geq 1 \) and \( \beta \in (0, 2^{-15/2}) \) it is \((\alpha, \beta)\)-accurate for any query stream \( F \) such that \( \sum_{t=n}^{\infty} \ell_t \leq \kappa \sum_{t=n}^{\infty} \exp \left( \frac{\alpha^2(1-p)cn^{-p}n^{-p}}{1152 \log(N)n/\log(1/\delta)} \right) \) for all \( t \geq n \) as long as \( N \geq 3, n \geq 17 \) and

\[
\alpha \geq \left( \frac{1224 \log^{1/2}(N)n/\log(1/\delta)}{(1-p)n/\alpha} \right)^{1/2}. 
\]

(B.12)

**Proof.** Applying Lemma 23 and Theorem 4 PMWG is \((\epsilon, \delta)\)-DP for

\[
\varepsilon' \leq \frac{24cn^{-q} \log^{1/2}(N)n/\log^{1/2}(1/\delta)}{\alpha q}
\]

where \( q = 1 - p > 0 \). Setting \( c = \frac{\alpha(1-p)n^{-p}}{24 \log^{1/2}(N)n/\log^{1/2}(1/\delta)} \) gives \( \varepsilon' = \epsilon \). The rest of the proof now follows the proof of Theorem 22 except that now \( \alpha cn^p = \frac{\alpha^2(1-p)n^{-p}n}{24 \log^{1/2}(N)n/\log^{1/2}(1/\delta)} \) instead of \( \alpha cn^p = \frac{\alpha^2(1-p)n^{-p}}{1152 \log(N)n/\log(1/\delta)} \). \( \square \)

Finally, we can make an observation analogous to that made at the beginning of Subsection B.1.1 for \((\epsilon, \delta)\)-differential privacy. Note that for \( \alpha \) as in Theorem 23 we have query budget

\[
\kappa \sum_{t=n}^{\infty} \exp \left( \frac{\alpha^2(1-p)cn^{-p}}{1152 \log^{1/2}(N)n/\log^{1/2}(1/\delta)} \right) \geq \kappa \sum_{t=n}^{\infty} \left( \frac{144 kn}{\beta} \right)^{\alpha n/(\log(1/\delta))} 
\]

(B.13)
As we let $p$ approach 1, we increase the query budget (lower bounded by $\Theta(\frac{\log^{1/2} N \log(k/\beta) \log(1/\delta)}{cn})^{1/2}$ as well) and suffer accuracy loss proportional to $(1 - p)^{-1/2}$ compared to $\Theta(\frac{\log^{1/2} N \log(k/\beta) \log(1/\delta)}{cn})^{1/2}$ accuracy of static $(\epsilon, \delta)$-private PMW, as long as $n$ is bounded polynomially in $N$ and $k/\beta$.

C Sparse Vector Algorithms for Growing Databases

In this section, we consider three primitive algorithms that apply the sparse vector technique in the static setting: above threshold, numeric above threshold, and numeric sparse. We modify these algorithms for the dynamic setting of growing databases, and we call our modifications ATG, NATG, and NSG, respectively. The main difference in analyzing privacy and accuracy for dynamic algorithms is that the results now depend on a changing database size. In this section, the database has starting size $t_0$ and it grows by one entry each time step, starting at time $t_0$.

C.1 Above Threshold for Growing Databases

We first consider the simplest of these algorithms: ATG. This algorithm simply compares a noisy answer to a query from stream $F = \{\{f_{t,i}\}_{i=1}^{t}\}_{t=t_0}^{\infty}$ applied to the current database $D_t$ to a noisy threshold $\hat{T}_t \approx T$ and aborts the first time a noisy answer exceeds the noisy threshold. The noise added is determined by a noise function $\xi$:

$$
\{t_0, t_0 + 1, \ldots\} \rightarrow \mathbb{R}.
$$

**Algorithm 4 ATG($D, F, T, \xi$)**

```plaintext
for each incoming query $f_{t,i}$ do
  if $i = 1$ then
    Let $\hat{T}_t \leftarrow T + \text{Lap}(\frac{2}{\xi_t})$ \hspace{1cm} \triangleright \text{Noisy threshold for queries at time } t
  end if
  Let $\nu_{t,i} \leftarrow \text{Lap}(\frac{4}{\xi_t})$ \hspace{1cm} \triangleright \text{Noise for query } f_{t,i}
  if $f_{t,i}(D_t) + \nu_{t,i} \geq \hat{T}_t$ then
    Output $a_{t,i} \leftarrow \top$ and halt \hspace{1cm} \triangleright \text{Abort on first query above threshold}
  else
    Output $a_{t,i} \leftarrow \bot$
  end if
end for
```

**Theorem 25** (Privacy of ATG). Let $\xi, \Delta : \{t_0, t_0 + 1, \ldots\} \rightarrow \mathbb{R}^+$ be such that both $\Delta, \xi, \Delta$ are non-increasing and $\xi$ is non-decreasing. Let $F = \{\{f_{t,i}\}_{i=1}^{t}\}_{t=t_0}^{\infty}$ be a stream of queries with sensitivity $\Delta_{f_{t,i}} \leq \Delta_t$ for all $(t, i)$. Then for any $a$ in the range of ATG and any neighboring database streams $D, D'$ of starting size $t_0$,

$$
\Pr[ATG(D, F, T, \xi) = a] \leq \exp(\xi_{t_0} \Delta_{t_0}) \Pr[ATG(D', F, T, \xi) = a]
$$

**Proof.** We follow the proof as in [DR14], but modify it slightly. Let $A, A'$ represent the random variables of output of ATG running on $D, D'$, respectively. Suppose ATG halts at the last query $f_{t',i'}$. Let $a$ denote this output, i.e. $a_{t,i} = \bot$ for all $(t, i) < (t', i')$ and $a_{t',i'} = \top$. Define

$$
H(D) = \max_{(t,i) < (t',i')} \xi_t \cdot (f_{t,i}(D_t) + \nu_{t,i} - T)
$$

29
Fix \( \nu_{t,i} \) for all \((t,i) < (t',i')\), so that \( H(D) \) is a deterministic quantity. Then,

\[
\Pr_{\eta, \nu_{t',i'}} [A = a] = \Pr_{\eta, \nu_{t',i'}} \left[ f_{t,i}(D_t) + \nu_{t,i} < \hat{T}_t, \forall (t,i) < (t',i') \text{ and } f_{t',i'}(D_{t'}) + \nu_{t',i'} \geq \hat{T}_{t'} \right]
\]

\[
= \Pr_{\eta, \nu_{t',i'}} [H(D) < \eta \text{ and } \xi_{t'} \cdot (f_{t',i'}(D_{t'}) + \nu_{t',i'} - T) \geq \eta]
\]

\[
= \Pr_{\eta, \nu_{t',i'}} [\eta \in (H(D), \xi_{t'} \cdot (f_{t',i'}(D_{t'}) + \nu_{t',i'} - T)]
\]

\[
= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Pr_{\eta_{t',i'}} [\nu_{t',i'} = v] \cdot \Pr_{\eta} [\eta = \eta_0] \mathbb{1} [\eta_0 \in (H(D), \xi_{t'} \cdot (f_{t',i'}(D_{t'}) + v - T)] dv \, d\eta_0
\]

\[:= *\]

Apply a change of variables as follows:

\[
\hat{v} = v + \frac{H(D) - H(D')}{\xi_{t'}}
\]

\[
\hat{\eta}_0 = \eta_0 + H(D) - H(D')
\]

We have \( |H(D) - H(D')| \leq \max_{t \geq t_0} \xi_t \Delta_t = \xi_0 \Delta_{t_0} \) (by \( \xi_t \Delta_t \) being non-increasing), and \( |f_{t',i'}(D_{t'}) - f_{t',i'}(D_{t'})| \leq \Delta_{t'} \). Therefore,

\[
|\hat{v} - v| \leq \frac{\xi_0 \Delta_{t_0}}{\xi_{t'}} + \Delta_{t'}, |\hat{\eta}_0 - \eta_0| \leq \xi_0 \Delta_{t_0}
\]

Apply this change of variable to get

\[*
= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Pr_{\nu_{t',i'}} [\nu_{t',i'} = \hat{v}] \Pr_{\eta} [\eta = \hat{\eta}_0] \mathbb{1} [\eta_0 + H(D) - H(D') \in (H(D), \xi_{t'} \cdot (f_{t',i'}(D_{t'}) + \hat{v} - T)] dv \, d\eta_0
\]

\[
= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Pr_{\nu_{t',i'}} [\nu_{t',i'} = \hat{v}] \Pr_{\eta} [\eta = \hat{\eta}_0] \mathbb{1} [\eta_0 \in (H(D'), \xi_{t'} \cdot (f_{t',i'}(D_{t'}) + \hat{v} - T) + H(D') - H(D)] dv \, d\eta_0
\]

\[
= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Pr_{\nu_{t',i'}} [\nu_{t',i'} = \hat{v}] \Pr_{\eta} [\eta = \hat{\eta}_0] \mathbb{1} [\eta_0 \in (H(D'), \xi_{t'} \cdot (f_{t',i'}(D_{t'}) + v - T)] dv \, d\eta_0
\]

\[
\leq \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left( \frac{\xi_{t'}}{4} \left( \frac{\xi_0 \Delta_{t_0}}{\xi_{t'}} + \Delta_{t'} \right) \right) \Pr_{\nu_{t',i'}} [\nu_{t',i'} = v] \exp \left( \frac{\xi_0 \Delta_{t_0}}{2} \right) \Pr_{\eta} [\eta = \eta_0] \mathbb{1} [\eta_0 \in (H(D'), \xi_{t'} \cdot (f_{t',i'}(D_{t'}) + v - T)] dv \, d\eta_0
\]

\[
= \exp \left( \frac{\xi_{t'}}{4} \left( \frac{\xi_0 \Delta_{t_0}}{\xi_{t'}} + \Delta_{t'} \right) + \frac{\xi_0 \Delta_{t_0}}{2} \right) \cdot \Pr_{\eta_{t',i'}} [H(D') < \eta \text{ and } \xi_{t'} \cdot (f_{t',i'}(D_{t'}) + \nu_{t',i'} - T) \geq \eta] \mathbb{1} [\eta_0 \in (H(D'), \xi_{t'} \cdot (f_{t',i'}(D_{t'}) + v - T)] dv \, d\eta_0
\]

\[
= \exp \left( \frac{\xi_0 \Delta_{t_0}}{4} + \frac{\xi_{t'} \Delta_{t'}}{4} + \frac{\xi_0 \Delta_{t_0}}{2} \right) \Pr_{\eta_{t',i'}} [A' = a]
\]

\[
\leq \exp (\xi_0 \Delta_{t_0}) \Pr_{\eta_{t',i'}} [A' = a]
\]

The first inequality comes from the bounds on \( |\hat{v} - v|, |\hat{\eta}_0 - \eta_0| \) and the pdf of Laplace distribution. The last inequality is by \( \xi_t \Delta_t \) being non-increasing.
Next, we define accuracy for threshold algorithms and then give an accuracy guarantee for ATG.

**Definition 6** (Accuracy of Threshold Answers). For $\alpha, \beta > 0$, an algorithm that produces threshold answers is $(\alpha, \beta)$-accurate for (finite) query stream $F = \{f_{t,i}\}_{i=1}^{t} \leq t'$ and threshold $T$ if for any input database stream $D = \{D_t\}_{t \geq t_0}$ with probability at least $1 - \beta$ the algorithm outputs answers $a_{t,i} \in \{\bot, T\}$ such that it does not halt before receiving the final query $f_{t',i'}$ and for all $a_{t,i} = T$,

$$f_{t,i}(D_t) \geq T - \alpha$$

and for all $a_{t,i} = \bot$,

$$f_{t,i}(D_t) \leq T + \alpha$$

**Theorem 26** (Accuracy of ATG). Let $\xi, \Delta : \{t_0, t_0 + 1, \ldots\} \rightarrow \mathbb{R}^+$ be such that both $\Delta, \xi \cdot \Delta$ are non-increasing and $\xi$ is non-decreasing. For $\alpha > 0$, ATG$(D, F, T, \xi)$ is $(\alpha, \beta)$-accurate for threshold $T$ and finite query stream $F = \{f_{t,i}\}_{i=1}^{t} \leq t'$ with sensitivity $\Delta_{f_{t,i}} \leq \Delta_t$ for all $(t, i)$ and $f_{t,i}(D_t) < T - \alpha$ for all $(t, i) \neq (t', i')$ and $f_{t',i'}(D_{t'}) \geq T + \alpha$ as long as

$$\beta \geq \sum_{t=0}^{t'} \ell_t \exp(-\frac{\alpha \xi t}{8}) + \exp(-\frac{\alpha \xi t_0}{8})$$

**Proof.** First, we want to show that $a_{t,i} = \bot$ if and only if $(t, i) < (t', i')$ with high probability. This is true if we can show that for all $(t, i)$,

$$|\nu_{t,i} - \eta| \leq \alpha$$

(C.1)

Because if so, we have that for all $(t, i) < (t', i')$, $f_{t,i}(D_t) + \nu_{t,i} < (T - \alpha) + (\alpha + \frac{\eta}{8}) = \tilde{T}_t$, so ATG will output $\bot$, and that $f_{t',i'}(D_{t'}) + \nu_{t',i'} \geq (T + \alpha) + (\frac{\eta}{8} - \alpha) = \hat{T}_t$, so ATG will output $a_{t',i'} = T$. To show (C.1), it is sufficient to require $|\nu_{t,i}| \leq \alpha/2$ and $|\eta| \leq \alpha/2$ for all $(t, i)$. The first requirement is false with probability $\exp(-\frac{\alpha \xi t}{8}) = \exp(-\frac{\alpha \xi t_0}{8})$ for each $(t, i)$. The second requirement is equivalent to $|\eta| \leq \alpha/2$ for a single time step $t_0$, which is false with probability $\exp(-\frac{\alpha \xi t_0}{8})$. By union bound, (C.1) is true except probability at most $\sum_{t=0}^{t'} \ell_t \exp(-\frac{\alpha \xi t}{8}) + \sum_{t=0}^{t'} \ell_t \exp(-\frac{\alpha \xi t_0}{8})$.

**C.2 Numeric Above Threshold for Growing Databases**

Next we analyze privacy and accuracy for NATG, which extends ATG by outputting a noisy answer to the first above threshold query.

**Algorithm 5** NATG$(D, F, T, \xi)$

```plaintext
for each incoming query $f_{t,i}$ do
    if $i = 1$ then
        Let $\tilde{T}_t \leftarrow T + \text{Lap}(\frac{\xi}{2})$  \Comment{Noisy threshold for queries at time $t$}
    end if
    Let $\nu_{t,i} \leftarrow \text{Lap}(\frac{\xi}{2})$ \Comment{Noise for query $f_{t,i}$}
    if $f_{t,i}(D_t) + \nu_{t,i} \geq \tilde{T}_t$ then
        Output $a_{t,i} \leftarrow f_{t,i}(D_t) + \text{Lap}(\frac{\xi}{2})$ and halt \Comment{Abort on first query above threshold}
    else
        Output $a_{t,i} \leftarrow \bot$
    end if
end for
```

31
Theorem 27 (Privacy of NATG). Let $\xi, \Delta : \{t_0, t_0 + 1, \ldots\} \to \mathbb{R}^+$ be such that both $\Delta, \xi \cdot \Delta$ are non-increasing and $\xi$ is non-decreasing. Let $F = \{\{f_{t,i}\}_{i=1}^{t}\}_{t \geq t_0}$ be a stream of queries with sensitivity $\Delta_{f_{t,i}} \leq \Delta_t$ for all $(t, i)$. Then for any $a$ in the range of NATG halting at time $t'$ and any neighboring database streams $D, D'$ of starting size $t_0$,

$$\Pr[NATG(D, F, T, \xi) = a] \leq \exp\left(\xi_{t_0} \Delta_{t_0} + \frac{\xi_{t'} \Delta_{t'}}{8}\right) \Pr[NATG(D', F, T, \xi) = a]$$

Proof. NATG privacy loss is the sum of ATG privacy loss and the loss by Laplace noise $\text{Lap}(\frac{\xi}{\Delta'})$ added to the numeric answer $a_{t',i'}$. The first is $\Delta_{t_0} \xi_{t_0}$ by Theorem 25 and the latter is $\xi_{t'} \Delta_{t'}/8$ because $f_{t,i}$ has sensitivity at most $1/\Delta_{t'}$. \qed

Next, we define accuracy for numeric threshold algorithms and give an accuracy guarantee for NATG.

Definition 7 (Accuracy of Threshold and Numeric Answers). For $\alpha, \beta > 0$, an algorithm that produces threshold and numeric answers is $(\alpha, \beta)$-accurate for (finite) query stream $F = \{\{f_{t,i}\}_{i=1}^{t}\}_{t_0 \leq t \leq t'}$ and threshold $T$ if for any input database stream $D = \{D_t\}_{t_0 \geq t}$, with probability $1 - \beta$ the algorithm outputs answers $a_{t,i} \in \mathbb{R} \cup \perp$ such that it does not halt before receiving the final query $f_{t',i'}$ and for all $a_{t,i} \in \mathbb{R}$,

$$|f_{t,i}(D_t) - a_{t,i}| \leq \alpha \text{ and } f_{t,i}(D_t) \geq T - \alpha$$

and for all $a_{t,i} = \perp$,

$$f_{t,i}(D_t) \leq T + \alpha$$

Theorem 28 (Accuracy of NATG). Let $\xi, \Delta : \{t_0, t_0 + 1, \ldots\} \to \mathbb{R}^+$ be such that both $\Delta, \xi \cdot \Delta$ are non-increasing and $\xi$ is non-decreasing. For $\alpha > 0$, NATG($D, F, T, \xi$) is $(\alpha, \beta)$-accurate for threshold $T$ and finite query stream $F = \{\{f_{t,i}\}_{i=1}^{t}\}_{t_0 \leq t \leq t'}$ with sensitivity $\Delta_{f_{t,i}} \leq \Delta_t$ for all $(t, i)$ and $f_{t,i}(D_t) < T - \alpha$ for all $(t, i) \neq (t', i')$ and $f_{t',i'}(D_{t'}) \geq T + \alpha$ as long as

$$\beta \geq \sum_{t=t_0}^{t'} \ell_t \exp(-\frac{\alpha \xi_t}{8}) + \exp(-\frac{\alpha \xi_{t_0}}{8}) + \exp(-\frac{\alpha \xi_{t'}}{8})$$

Proof. We apply Theorem 20 so that except with at most probability $\sum_{t=t_0}^{t'} \ell_t \exp(-\frac{\alpha \xi_t}{8}) + \exp(-\frac{\alpha \xi_{t_0}}{8})$, NATG is accurate for threshold answers, i.e. $f_{t,i}(D_t) \geq T - \alpha$ and $f_{t,i}(D_t) \leq T + \alpha$ hold for $a_{t,i} \in \mathbb{R}$ and $a_{t,i} = \perp$, respectively.

It’s left to show that $|f_{t',i'}(D_{t'}) - a_{t',i'}| \leq \alpha$. This is true except with probability $\exp(-\frac{\alpha \xi_{t'}}{8})$ by Laplace mechanism. Therefore,

$$\beta \leq \sum_{t=t_0}^{t'} \ell_t \exp(-\frac{\alpha \xi_t}{8}) + \exp(-\frac{\alpha \xi_{t_0}}{8}) + \exp(-\frac{\alpha \xi_{t'}}{8})$$

\qed

C.3 Numeric Sparse for Growing Databases

We now compose NATG multiple times into numeric sparse for growing databases (NSG). Note that we may run NATG infinitely many times as long as there is input coming online. Any query that causes NATG to output a number and halt is called hard; any other query is called easy.

Analysis of privacy of NSG can be done by simply summing up the privacy loss stated for NATG. Since privacy for NATG changes over time, this privacy loss is then dependent on the time that NATG starts and ends, i.e. when hard queries come.
Algorithm 6 NSG(D, F, T, ξ)

for each incoming query \( f_{t,i} \) do
    if no NATG subroutine is currently running then
        Initialize a NATG subroutine with the same arguments
    end if
    Output the NATG subroutine’s output for \( f_{t,i} \).
end for

Theorem 29 (Privacy of NSG). Let \( \xi, \Delta : \{t_0, t_0 + 1, \ldots \} \rightarrow \mathbb{R}^+ \) be such that both \( \Delta, \xi \cdot \Delta \) are non-increasing and \( \xi \) is non-decreasing. Let \( F = \{\{f_{t,i}\}_{i=1}^{t_t}\}_{t \geq t_0} \) be a stream of queries with sensitivity \( \Delta_{f_{t,i}} \leq \Delta_t \) for all \((t, i)\). Then for any \( a \) in the range of NSG with \( h_t \) hard queries arriving at time \( t \) for each \( t \geq t_0 \) and any neighboring database streams \( D, D' \) of starting size \( t_0 \),

\[
\Pr [\text{NSG}(D, F, T, \xi) = a] \leq \exp \left( \xi_{t_0} \Delta_{t_0} + \frac{9}{8} \sum_{t \geq t_0} h_t \xi_t \Delta_t \right) \Pr [\text{NSG}(D', F, T, \xi) = a]
\]

Proof. Let \( t_{0j}, t_{1j} \) be the start and end time of \( j \)th round of NATG in NSG. Then by Theorem 27, the privacy loss is at most

\[
\epsilon := \sum_j \left( \xi_{t_{0j}} \Delta_{t_{0j}} + \frac{\xi_{t_{1j}} \Delta_{t_{1j}}}{8} \right)
\]

The start time \( t_{0j} \) of round \( j \) is at least the end time of last round \( t_{1j-1} \) for each \( j \geq 2 \), and \( t_{1j} \geq n \), so

\[
\epsilon \leq \xi_n \Delta_n + \sum_j \left( \xi_{t_{1j}} \Delta_{t_{1j}} + \frac{\xi_{t_{1j}} \Delta_{t_{1j}}}{8} \right)
\]

\[
= \xi_n \Delta_n + \frac{9}{8} \sum_{t=n}^{\infty} h_t \xi_t \Delta_t
\]

The last equality is by the fact that the end times of NATG are exactly when hard queries come.

Now we apply Theorem 29 to a more specific setting of linear queries.

Corollary 30. For adaptively chosen linear query stream \( F \), NSG(D, F, T, ξ) for database streams of starting size \( n \) and noise function \( \xi_t = t^p \) for some \( p \in [0, 1] \), is \((n^p + \frac{9}{8} \sum_{t=n}^{\infty} h_t t^{p-1}, 0)\)-DP.

Proof. From the fact that \( \xi_t \) is non-decreasing, that linear queries have sensitivity \( \Delta_t = 1/t \), and that \( \xi_t \Delta_t = t^{p-1} \) is non-increasing, the conditions satisfy Theorem 29’s assumption, so we can apply Theorem 29.

Corollary 30 suggests that in order to bound privacy loss, we need to bound the number of hard queries, especially those arriving early in time.

Now we state the accuracy for NSG. Note that we use the same Definition 7 from NATG, since NSG also produces threshold and numeric answers.

Theorem 31 (Accuracy of NSG). Let \( \xi, \Delta : \{t_0, t_0 + 1, \ldots \} \rightarrow \mathbb{R}^+ \) be such that both \( \Delta, \xi \cdot \Delta \) are non-increasing and \( \xi \) is non-decreasing. Let \( k : \{t_0, t_0 + 1, \ldots \} \rightarrow \mathbb{R} \) be a query budget. For \( \alpha > 0 \), NATG(D, F, T, ξ) is \((\alpha, \beta)\)-accurate for threshold \( T \) and finite query stream \( F = \{\{f_{t,i}\}_{i=1}^{t_t}\}_{t_0 \leq t \leq t'} \) with

\[
\]
sensitivity $\Delta_{t,i} \leq \Delta_t$ for all $(t, i)$ and respecting cumulative query budget for $\sum_{t=t_0}^t \ell_t \leq \sum_{t=t_0}^t k_t$ for all $t \geq t_0$ as long as

$$\beta \geq \exp\left(-\frac{\alpha \xi_n}{8}\right) + \sum_{t=n}^\infty (\ell_t + 2h_t) \exp\left(-\frac{\alpha \xi_t}{8}\right),$$

(C.2)

where $h_t = |\{i : f_{t,i}(D_t) \geq T - \alpha\}|$ for $t \geq t_0$. This implies $(\alpha, \beta)$-accuracy for

$$\beta \geq \exp\left(-\frac{\alpha \xi_n}{8}\right) + 3 \sum_{t=n}^\infty k_t \exp\left(-\frac{\alpha \xi_t}{8}\right).$$

(C.3)

Proof. We need to show that except with probability at most $\beta$:

1. For each $a_{t,i} = \perp$, $f_{t,i}(D_t) \leq T + \alpha$
2. For each $a_{t,i} \in \mathbb{R}$, $f_{t,i}(D_t) \geq T - \alpha$
3. For each $a_{t,i} \in \mathbb{R}$, $|f_{t,i}(D_t) - a_{t,i}| \leq \alpha$

Suppose the $j$th round of NATG starts and ends at time $t_{0,j}, t_{1,j}$, and answers $\ell^j_t$ queries at time $t$. The set of three conditions is equivalent to requiring that $j$th NATG round satisfies Definition 7 with respect to threshold $T$ and stream of queries that $j$th round of NATG answers. By Theorem 28 and union bound, the rest of the proof is a calculation: we can take $\beta$ to be

$$\beta = \sum_j \left(\sum_{t=n}^{\infty} \ell^j_t \exp\left(-\frac{\alpha \xi_t}{8}\right) + \exp\left(-\frac{\alpha \xi_{t-1}^j}{8}\right) + \exp\left(-\frac{\alpha \xi_{t-1}^j}{8}\right)\right)$$

$$\leq \sum_j \left(\sum_{t=n}^{\infty} k^j_t \exp\left(-\frac{\alpha \xi_t}{8}\right) + \sum_j \left(\exp\left(-\frac{\alpha \xi_{t-1}^j}{8}\right) + \exp\left(-\frac{\alpha \xi_{t-1}^j}{8}\right)\right)\right)$$

$$\leq \left(\sum_{t=n}^{\infty} \ell_t \exp\left(-\frac{\alpha \xi_t}{8}\right) + \exp\left(-\frac{\alpha \xi_n}{8}\right) + \sum_j 2 \exp\left(-\frac{\alpha \xi_t}{8}\right)\right)$$

The first inequality is by the fact that start time of the next round is after the end of the current round, and we let $t_{1,j} = n$ for convenience. By condition (2), NATG can only halt on queries $f_{t,i}$ such that $f_{t,i}(D_t) \geq T - \alpha$, so there are at most $h_t$ rounds of NATG halting at time $t$. Therefore,

$$2 \sum_j \exp\left(-\frac{\alpha \xi_{t}^j}{8}\right) \leq 2 \sum_{t=n}^{\infty} h_t \exp\left(-\frac{\alpha \xi_t}{8}\right)$$

which finishes the proof for (C.2). Inequality (C.3) follows from the observations that $\ell_t \geq h_t$ and that $\exp\left(-\frac{\alpha \xi_t}{8}\right)$ is a non-increasing function of $t$. 

\[\square\]