

Differentially Private Synthetic High-dimensional Tabular Stream

Girish Kumar
gkum@ucdavis.edu
Department of Mathematics,
University of California
Davis, CA, USA

Thomas Strohmmer
strohmmer@math.ucdavis.edu
Department of Mathematics,
University of California
Davis, CA, USA

Roman Vershynin
rvershyn@uci.edu
Department of Mathematics,
University of California
Irvine, CA, USA

ABSTRACT

While differentially private synthetic data generation has been explored extensively in the literature, how to update this data in the future if the underlying private data changes is much less understood. We propose an algorithmic framework for streaming data that generates multiple synthetic datasets over time, tracking changes in the underlying private data. Our algorithm satisfies differential privacy for the entire input stream (continual differential privacy) and can be used for high-dimensional tabular data. Furthermore, we show the utility of our method via experiments on real-world datasets. The proposed algorithm builds upon a popular *select, measure, fit, and iterate* paradigm (used by offline synthetic data generation algorithms) and private *counters* for streams.

KEYWORDS

differential privacy, synthetic data, tabular, stream

1 INTRODUCTION

Data availability has become crucial to technological advancement in today’s world. Publicly available datasets have the additional advantage that people across various domains, affiliations, and geographic locations can contribute to the research on such datasets. However, in many critical domains such as healthcare, public policy, and market research, it is challenging to release curated datasets publicly without compromising user privacy. A potential solution explored in many previous works is the release of a synthetic dataset that mimics some relevant statistical properties of the private data. Differential privacy has emerged as a standard notion to provide a mathematical guarantee that the synthetic data preserves the privacy of individuals contributing to the private data. Achieving differential privacy is non-trivial and typically requires adding carefully calibrated noise to measurements of true data. Many existing works in the differential privacy literature have explored the task of generating synthetic data such as [1, 9, 16, 21–23, 25, 26].

Most of the research in differential privacy has focused on generating the synthetic dataset once, based on all private data available at the time. However, in many real-world scenarios, the private data may change over time resulting in a requirement to update the synthetic data with time as well. For example, consider the electronic health records of patients admitted to the hospital in the middle of a pandemic such as COVID-19. The availability of public data which can be updated over time will allow research developments in real time.

Motivated by this scenario, in this work, we are interested in developing an algorithm to generate a privacy-preserving synthetic high-dimensional *tabular stream*. A high-dimensional tabular data is where each record in the dataset consists of (say) p fixed attributes, where p is sufficiently large. A *stream* is a collection of such datasets

over time. Moreover, our algorithm is streaming (online) in the sense that we generate the updated synthetic dataset at each time, only using the private stream until that time. To preserve privacy, we will use the notion of *continual differential privacy*, which extends the concept of (offline) differential privacy to streaming algorithms. We discuss these terms and the setup rigorously in Section 3.

2 OVERVIEW AND RELATED WORK

The notion of differential privacy for streaming algorithms, such that the privacy guarantee spans the entire time horizon, was introduced in the seminal work of [4] and [8]. They also introduced the concept of *counters*, which are differentially private streaming algorithms that can efficiently count the occurrence of an event over an input stream. Counters have been used as building blocks of many streaming algorithms such as [5, 14, 24]. Our proposed method also uses counters as sub-routines.

However, differential privacy has not been explored as much for other complicated streaming tasks such as synthetic data generation. To the best of our knowledge [3], [12], and [17] are the only other works that explore differentially private synthetic data generation with streaming algorithms. [3] approach a different problem of generating synthetic streams for a fixed universe of users such that each user contributes at all times. Moreover, they assume that each record in data is a boolean. Both [12] and [17] provide an algorithm that works with a hierarchical decomposition of the data space. [12] provides superior theoretical guarantees of their proposed algorithm but does not provide any experimental evaluation. [17] limits the experiments to low-dimensional domains such as spatial streams. However, algorithms based on hierarchical decomposition typically do not scale well for high-dimensional data as the number of nodes in the tree grows exponentially with dimension leading to large time complexity and poor utility. To that end, this work is the first to provide a differentially private streaming algorithm for synthetic data generation that is tractable for real-world datasets and provides better utility than the trivial baseline. In the discussion that follows, we give an overview of our method and discuss how research in offline differential privacy has motivated it.

There is a plethora of research in differential privacy on offline algorithms generating synthetic tabular datasets such as [1, 10, 11, 16, 18–22, 27]. While they cannot be directly applied for our use case, they are the motivation behind our proposed method. First, similar to most of these works, we measure the quality of the synthetic stream using marginal queries. A marginal query counts the number of instances in a dataset where a particular combination of the values of some attributes occurs. For the previously mentioned data space of electronic health records, a marginal query may be - “how many patients are more than 50 years old, have been previously diagnosed with Asthma, and have tested positive for COVID-19”?

In this work, we are interested in a pre-defined set of marginal queries and we target that at any time the synthetic stream has almost the same value for any query as the true stream.

Second, similar to most of the offline algorithms mentioned earlier, we use the *select, measure, fit, and iterate* paradigm. In this approach, the algorithm iteratively selects a query (typically the worst-performing query), measures it, and fits the data-generating model according to this noisy measurement. An early work that used this paradigm is MWEM [11] which combined the Multiplicative Weights algorithm with the Exponential Mechanism (a differentially private selection algorithm) to generate synthetic data. The MW algorithm directly maintains an estimated distribution on the entire data space and adjusts the distribution to comply with the noisy measurements of the selected marginal queries. The algorithm thus quickly becomes intractable for reasonably high dimensions of data. MWEM is typically the best-performing algorithm for low data dimensionality [18]. In this work, we build upon MWEM+PGM [20], a scalable version of the MWEM approach that replaces modeling the data distribution from MW to a Probabilistic Graphical Model (PGM), and has been shown to perform well for a variety of real-world datasets [22].

A related task to our problem is generating answers to histogram queries for a stream. This problem has been explored in works such as [13] and [15]. However, they limit the data space to the set $\{0, 1\}^d$ (for some dimension d) and the histogram queries to the column sum. Moreover, while the accuracy scales logarithmically in time, it scales linearly with d . In contrast, our method outperforms a baseline streaming algorithm, which exhibits accuracy scaling poly-logarithmically with the number of queries. Thus, it is more practical for answering k -way marginal queries for high-dimensional datasets.

There exists a simple baseline differentially private streaming algorithm for our problem - run an independent instance of an offline algorithm (such as MWEM+PGM) on the differential data at any time to create the stream [17]. We show in our experiments that our proposed method outperforms such a baseline. In principle, our method has two key differences: (1) we use counters as a sub-routine to measure any query over time, and (2) at any time, we recycle some information about the queries from the synthetic stream generated so far. The contributions of this work are summarized below:

- (1) we present a novel framework that extends the *select, measure, fit, and iterate* paradigm from offline to streaming algorithms for synthetic data generation;
- (2) we give a theoretical accuracy guarantee for the baseline algorithm;
- (3) furthermore, we demonstrate using experiments that our proposed method outperforms the baseline on real-world high-dimensional streams.

3 PROBLEM SETUP AND USEFUL TOOLS

3.1 Notation

Let $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Let $[n]$ denote the set $\{1, 2, \dots, n\}$ for any $n \in \mathbb{N}$. Let $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_p$ denote a space of p -dimensional points such that $\mathcal{X}_i$ be a finite set of cardinality $|\mathcal{X}_i|$ for any $i \in [p]$. For example, if each data point results from a survey of p boolean

questions, then $\mathcal{X} = \{0, 1\}^p$. We refer to any mapping of the form $h : \mathcal{X} \rightarrow \mathbb{N}_0$ as a *dataset* such that $h(x)$ represents the number of times a point $x \in \mathcal{X}$ appears. We denote by $|h|$ the total number of points in the dataset, that is $|h| = \sum_{x \in \mathcal{X}} h(x)$.

In this work, we are interested in a dataset that changes with time, resulting in a data *stream*. Let $f : \mathcal{X} \times \mathbb{N} \rightarrow \mathbb{N}_0$ be an input stream where $f(x, t)$ denotes the number of instances of point x at time t . For example, consider that $\mathcal{X}$ is the set of all possible demographics in a country's voting population. Then we can represent the eligible voters in the country as a stream $f : \mathcal{X} \times \mathbb{N} \rightarrow \mathbb{N}_0$ where $f(x, t)$ denotes the number of individuals in the population with demographics $x \in \mathcal{X}$ that are eligible to vote at time t . We provide a streaming algorithm that generates a privacy-preserving synthetic data stream $g : \mathcal{X} \times \mathbb{N} \rightarrow \mathbb{N}_0$ such that g accurately represents the input stream f . We define the terms "streaming algorithm", "privacy-preserving", and "accurately" rigorously in the subsequent subsections.

For any $N \subseteq \mathbb{N}$, we will use the notation f_N to denote the restriction of the stream f to the time indices in the set N , that is $f_N : \mathcal{X} \times N \rightarrow \mathbb{N}_0$ such that $f_N(x, t) = f(x, t)$ for all $t \in N$ and $x \in \mathcal{X}$. Similarly, for any time $t \in \mathbb{N}$, $f_t : \mathcal{X} \rightarrow \mathbb{N}_0$ denotes a restriction of f to time t such that $f_t(x) = f(x, t)$ for all $x \in \mathcal{X}$.

We have assumed that $\mathcal{X}_i$ is a finite set for all $i \in [p]$. While this assumption may not hold in practice, we can *discretize* a continuous space by spending some of the privacy budget to create a differentially private histogram and mapping each point to the histogram bin [22].

3.2 Streaming algorithm

As a motivating example, consider an algorithm $\mathcal{A}$ that converts a given input data stream $f : \mathcal{X} \times \mathbb{N} \rightarrow \mathbb{N}_0$ into a synthetic data stream $g : \mathcal{X} \times \mathbb{N} \rightarrow \mathbb{N}_0$. The key idea of a streaming algorithm (Definition 3.1) is that at each time t , it can only see the part of the input stream $f(x, t')$ for all $x \in \mathcal{X}$ and $t' \leq t$, and at that time the algorithm outputs $g(x, t)$ for all $x \in \mathcal{X}$.

Definition 3.1 (Streaming algorithm). Given an input stream $f : \mathcal{X} \times \mathbb{N} \rightarrow \mathbb{N}_0$, an algorithm $\mathcal{A}$ with output stream $g : \mathcal{X} \times \mathbb{N} \rightarrow \mathbb{N}_0$ is said to be streaming if at any time $t \in \mathbb{N}$ it maps $f_{[t]}$ to g_t , that is, $g(t, x) := \mathcal{A}(f_{[t]})(t, x)$ for all $x \in \mathcal{X}$.

3.3 Differential Privacy

To rigorously define privacy, we use the notion of Differential Privacy [7] but for a streaming algorithm. Intuitively, differential privacy ensures that the output of an algorithm does not depend extensively on any particular user's data by ensuring robustness in the output probability distribution against change in a single data point. To this end, we need a concept of streams that differ in a single data point. Borrowing from [17], we first provide definitions of differential and neighboring streams and finally the extension of differential privacy to streaming algorithms.

Definition 3.2 (Differential stream). A *differential stream* for f is the stream ∇f defined as,

$$\nabla f(x, t) := f(x, t) - f(x, t - 1), \quad t \in \mathbb{N}, \quad (1)$$

where we set $f(x, 0) = 0$. The total change of f over all times and points is the quantity

$$\|f\|_{\nabla} := \sum_{x \in \mathcal{X}} \sum_{t \in \mathbb{N}} |\nabla f(x, t)|, \quad (2)$$

which defines a seminorm on the space of data streams.

Definition 3.3 (Neighboring streams). Two streams f and $\tilde{f}$ are said to be neighbors if

$$\|f - \tilde{f}\|_{\nabla} = 1, \quad (3)$$

that is if $\tilde{f}$ can be obtained from f by changing the count of a single data point at some particular time.

Definition 3.4 (Differential privacy (for streams)). A randomized streaming algorithm $\mathcal{A}$ that takes data streams as input is ϵ -differentially private if for any two neighboring streams f and $\tilde{f}$ that satisfy $\|f - \tilde{f}\| = 1$, the inequality

$$\mathbb{P}\{\mathcal{A}(\tilde{f}) \in S\} \leq e^{\epsilon} \cdot \mathbb{P}\{\mathcal{A}(f) \in S\} \quad (4)$$

holds for any measurable set of outputs S .

3.4 Accuracy over marginal queries

In this work, we measure the accuracy of our output stream using marginal queries. A marginal query is a low-dimensional counting query. For example, suppose we have a census-like dataset where some of the demographics are age, marital status, and income. An example marginal query on these attributes is - how many individuals in the dataset are age 30, never married, and have income more than \$100,000 per annum. We define marginal query more formally in Definition 3.5.

Definition 3.5 (k -way marginal query). A k -way marginal query $q : \mathcal{X} \rightarrow \{0, 1\}$ is a mapping defined by a tuple $(c_1, c_2, \dots, c_k)$ of k attribute indices and their corresponding values $(v_1, v_2, \dots, v_k)$ such that $v_i \in \mathcal{X}_{c_i}$ for all $i \in [k]$ and the mapping is defined as,

$$q(x) = \prod_{i=1}^k \left(\mathbb{1}_{\{x_{c_i} = v_i\}} \right), \quad (5)$$

for any $x \in \mathcal{X}$. With a slight abuse of notation, we extend the definition of the marginal query q from points to datasets as,

$$q(h) = \sum_{x \in \mathcal{X}} h(x) q(x), \quad (6)$$

for any dataset $h : \mathcal{X} \rightarrow \mathbb{N}_0$.

3.4.1 Accuracy. We first discuss the accuracy of offline algorithms and then extend it to streaming algorithms. The quality of an algorithm generating a synthetic dataset is typically measured by the aggregate performance of the resulting synthetic data over a predefined set of marginal queries (say) Q . We define the accuracy formally in Definition 3.6.

Definition 3.6 (Accuracy of an algorithm generating synthetic dataset). Let $\mathcal{A}$ be a randomized algorithm that maps an input dataset $f : \mathcal{X} \rightarrow \mathbb{N}_0$ to a synthetic dataset $g : \mathcal{X} \rightarrow \mathbb{N}_0$. Then, for any $\beta > 0$, $\mathcal{A}$ is said have an accuracy of (α, β) , with respect to a set of marginal queries Q , if

$$\mathbb{P} \left\{ \max_{q \in Q} |q(g) - q(f)| \geq \alpha \right\} \leq \beta, \quad (7)$$

with probability taken over the randomness of algorithm $\mathcal{A}$.

A natural extension of Definition 3.6, to streams can be created by restricting the stream to any time $t \in \mathbb{N}$ and looking at the accuracy of the dataset present at that time.

Definition 3.7 (Accuracy of a streaming algorithm). Let $\mathcal{A}$ be a randomized streaming algorithm that maps an input stream $f : \mathcal{X} \times \mathbb{N} \rightarrow \mathbb{N}_0$ to a synthetic stream $g : \mathcal{X} \times \mathbb{N} \rightarrow \mathbb{N}_0$. Then, at any time $t \in \mathbb{N}$ and for any $\beta > 0$, $\mathcal{A}$ is said have an accuracy of (α, β) , with respect to a set of marginal queries Q , if

$$\mathbb{P} \left\{ \max_{q \in Q} |q(g_t) - q(f_t)| \geq \alpha \right\} \leq \beta, \quad (8)$$

where the probability is taken over the randomness of the algorithm $\mathcal{A}$. Here, α may be a function of β and t .

3.5 Exponential Mechanism

Let $\mathcal{R}$ be a finite set. Let $u : \mathbb{N}_0^{\mathcal{X}} \times \mathcal{R} \rightarrow \mathbb{R}$ be a function such that $u(h, r)$ denotes the utility of an element $r \in \mathcal{R}$ for a dataset $h \in \mathbb{N}_0^{\mathcal{X}}$. Our task is to find an element in $\mathcal{R}$ with maximum utility while preserving differential privacy. Note that the term utility is very general and its exact definition is governed by the problem. As an example, suppose we want to find a mode of a given dataset $h \in \mathbb{N}_0^{\mathcal{X}}$. The mode is any point $x_* \in \mathcal{X}$ such that $x_* = \arg \max_{x \in \mathcal{X}} h(x)$. We can use the exponential mechanism in this case with the utility being the absolute difference between the frequency of any point from the maximum possible frequency in h . Thus in this case $\mathcal{R} = \mathcal{X}$, and $u(h, x; x_*) = |h(x) - h(x_*)|$ for all $x \in \mathcal{X}$.

Definition 3.8 (Exponential mechanism). Let Δ_u denote the sensitivity of the utility function defined as,

$$\Delta_u := \max_{h, \tilde{h} \in \mathbb{N}_0^{\mathcal{X}}} \max_{\substack{r \in \mathcal{R} \\ \|h - \tilde{h}\| = 1}} |u(h, r) - u(\tilde{h}, r)|. \quad (9)$$

Then, the exponential mechanism is defined as the algorithm $\mathcal{A} : \mathbb{N}_0^{\mathcal{X}} \rightarrow \mathcal{R}$ such that, for all $r \in \mathcal{R}$,

$$\mathbb{P} \{ \mathcal{A}(h) = r \} \propto \exp \left(-\frac{\epsilon}{2\Delta_u} u(h, r) \right).$$

THEOREM 3.9 (PRIVACY OF EXPONENTIAL MECHANISM). *The exponential mechanism, as defined in Definition 3.8, satisfies ϵ -differential privacy.*

THEOREM 3.10 (ACCURACY OF EXPONENTIAL MECHANISM). *For any $\beta > 0$, the exponential mechanism, as defined in Definition 3.8, satisfies*

$$\mathbb{P} \left\{ u(h, \mathcal{A}(h)) \leq u_{OPT} - \frac{2\Delta_u}{\epsilon} \ln \left(\frac{|\mathcal{R}|}{\beta} \right) \right\} \leq \beta,$$

where $u_{OPT} = \max_{r \in \mathcal{R}} u(h, r)$ denotes the maximum possible utility.

3.6 Counters

We borrow Definition 3.11 from [17]. Intuitively, it is an algorithm that estimates the sum of a stream with a certain accuracy.

Definition 3.11 (Counter). An (α, δ) -accurate counter C is a randomized streaming algorithm that estimates the sum of an input stream of values $f : \mathbb{N} \rightarrow \mathbb{R}$ and maps it to an output stream of values $g : \mathbb{N} \rightarrow \mathbb{R}$ such that at any time $t \in \mathbb{N}$,

$$\mathbb{P}\left\{\left|g(t) - \sum_{t' \leq t} f(t')\right| \leq \alpha(t, \delta)\right\} \geq 1 - \delta,$$

where the probability is over the randomness of C and δ is a small constant.

[4] and [8] first introduced differentially private counters that efficiently find the sum of a bit stream. The algorithms proposed in these works satisfy Definition 3.11. In this work we will be using the Simple II, Two-Level, and Hybrid Mechanism from [4], hereafter referred to as Simple, Block, and Binary Tree Counters respectively. As explained in [4], the key principle behind the design of these algorithms is dividing the time horizon into intervals and adding together the *noisy partial sums* from these intervals. For a fixed failure probability, the simple, block and binary tree counters are $O(\sqrt{t})$, $O(t^{1/4})$, and $O((\ln t)^{3/2})$ accurate respectively.

4 OFFLINE TABULAR SYNTHETIC DATASET GENERATION

Our streaming algorithm uses many ideas from offline algorithms in the literature for dataset generation. Let us first discuss these ideas. Consider the task of generating synthetic tabular data when the dataset is available at once (offline). Let $f : \mathcal{X} \rightarrow \mathbb{N}_0$ be a dataset. Assume we are interested in generating a synthetic dataset $g : \mathcal{X} \rightarrow \mathbb{N}_0$ that is accurate for marginal queries Q . A straightforward approach to generating the synthetic dataset would be: (1) generate a differentially private measurement for all queries using the Laplace Mechanism as

$$m = \left(q(f) + \text{Lap}\left(\frac{\Delta q}{\epsilon/|Q|}\right) \right)_{q \in Q};$$

(2) find a dataset that minimizes the maximum error over the query set by solving the following optimization problem,

$$\arg \min_{g \in \mathbb{N}_0^{\mathcal{X}}} \max_{q \in Q} |m_q - q(g)|. \quad (10)$$

There are however two key problems with this approach: (1) the size of the query set is typically polynomial in the dimension p which leads to a very small budget for answering an individual query, that is a large amount of noise is added in Laplace Mechanism, and (2) the optimization problem in equation (10) is a high-dimensional discrete optimization problem which is NP-Hard and cannot be solved in time polynomial in dimension p . Many existing algorithms thus circumvent the above two problems by: (1) measuring only a subset of the queries in Q which have the largest error, and (2) approximating the optimization problem in Equation 10. Let us first assume that we have a way to model the data distribution given the noisy measurements. Let $\mathcal{A}_{\text{Dataset}}$ be one such subroutine that takes as input noisy measurements of the queries in Q and an initialization dataset (say) $h \in \mathbb{N}_0^{\mathcal{X}}$, to provide a dataset $g \in \mathbb{N}_0^{\mathcal{X}}$ that complies with the measurements. In Section 4.1, we

Algorithm 1 Meta algorithm: generating differentially private synthetic tabular dataset

- 1: **Input:** Given dataset f , an ordered set of queries Q , privacy budget ϵ , a differentially private selection mechanism $\mathcal{A}_{\text{Select}}$, a subroutine $\mathcal{A}_{\text{Dataset}}$ to find a dataset given noisy query measurements, and the number of iterations k .
- 2: **Output:** A dataset $g \in \mathbb{N}_0^{\mathcal{X}}$.
- 3: Create a dataset $h_0 \in \mathbb{N}_0^{\mathcal{X}}$ with $h_0(x) = 1$ for all $x \in \mathcal{X}$.
- 4: Set $M \leftarrow \emptyset$ as a set of selected queries and their measurements.
- 5: **for** $i = 1, 2, \dots, k$ **do**
- 6: Set $e_i \leftarrow \left(|q(h_{i-1}) - q(f)| \right)_{q \in Q}$ as the error in queries.
- 7: **Select:** $l_i \leftarrow \mathcal{A}_{\text{Select}}(e_i, 2/\epsilon)$, an index of query.
- 8: **Measure:** $m_i \leftarrow q_{l_i}(f) + \text{Lap}\left(2\Delta_{q_{l_i}}/\epsilon\right)$, value of query.
- 9: Set $M \leftarrow M \cup \{(q_{l_i}, m_i)\}$; add selected query and its value.
- 10: **Optimize:** Dataset $h_i \leftarrow \mathcal{A}_{\text{Dataset}}(M, h_{i-1})$.

use $\mathcal{A}_{\text{Dataset}}$ as a black-box subroutine and discuss how to iteratively select and measure a subset of the queries. In Section 4.2, we then look at some of these methods for creating the dataset given a value of the queries.

4.1 The Select, Measure, Fit, and Iterate paradigm

Algorithm 1 is a meta-algorithm describing the *select, measure, fit, and iterate* paradigm. This paradigm is used in several existing methods and has been shown to achieve good empirical accuracy [18]. The algorithm has a fixed number of iterations k . It receives two subroutine algorithms $\mathcal{A}_{\text{Select}}$ and $\mathcal{A}_{\text{Dataset}}$ which can be treated as a black box for now. $\mathcal{A}_{\text{Select}}$ is a differentially private algorithm used for selection, whereas $\mathcal{A}_{\text{Dataset}}$ does not guarantee differential privacy and is used to create a dataset based on the values of the queries. Algorithm 1 iteratively produces a series of synthetic datasets $h_1, h_2, \dots, h_k$ that are, hopefully, more and more closer to the true data f as per the queries Q . In each iteration i , the following happens: (1) using the subroutine $\mathcal{A}_{\text{Select}}$, while upholding differential privacy, we select a query q_{l_i} that has the most error on the current synthetic dataset h_{i-1} ; (2) an approximation of the value of this query is generated as m_i using the Laplace Mechanism; and finally (3) the dataset is updated from h_{i-1} to h_i by using the sub-routine $\mathcal{A}_{\text{Dataset}}$.

4.2 Dataset complying with queries

In this subsection, we discuss some algorithms that can be used for $\mathcal{A}_{\text{Dataset}}$ in Algorithm 1.

4.2.1 Multiplicative Weights (MW). [11] first introduced the idea of Algorithm 1 and used the *Multiplicative Weights* (MW) algorithm as $\mathcal{A}_{\text{Dataset}}$ together with the Exponential Mechanism for $\mathcal{A}_{\text{Select}}$. With the MW algorithm, Step 10 of Algorithm 1 results in a dataset h_i that is $|f|$ times the distribution that satisfies

$$h_i(x) \propto h_{i-1}(x) \cdot \exp\left(q_{l_i}(x) \cdot \frac{m_i - q_{l_i}(h_{i-1})}{2|f|}\right).$$

MWEM solves a convex approximation of Equation (10) over the probability simplex in $\mathcal{X}$, we refer the reader to [18] for more details. The algorithm comes with a theoretical guarantee and works quite well for low-dimensional datasets. However, since it requires maintaining a probability distribution over $\mathcal{X}$, it becomes computationally intractable for many real-world datasets.

4.2.2 Probabilistic Graphical Model (PGM). An alternative to the MW algorithm as $\mathcal{A}_{Dataset}$ in Algorithm 1 is the Probabilistic Graphical Model (PGM) algorithm [21]. PGM further approximates the optimization problem by restricting the solution space from all possible distributions on $\mathcal{X}$ to distributions that can be represented as a graphical model of the form

$$P_\theta(x) = \frac{1}{Z} \exp\left(\sum_{C \in \mathcal{C}} \theta_C(x_C)\right),$$

for all $x \in \mathcal{X}$. Here, $\mathcal{C} \subseteq 2^{[d]}$ is a collection of subsets of $[d]$, θ_C is a function for each $C \in \mathcal{C}$, x_C is the restriction of $x \in \mathcal{X}$ on the column indices in C , and Z is a normalization constant. Thus the model P_θ is defined by low-dimensional functions θ_C , one for each $C \in \mathcal{C}$. PGM uses a proximal algorithm to solve the resulting convex optimization problem. PGM has been shown to perform very well in practice and we will be using it in our experiments.

5 BASELINE: STREAMING MWEM

Let us get to our problem of producing synthetic stream g for private stream f . In this section, we propose a baseline algorithm which can convert any offline algorithm $\mathcal{A}$ to a streaming algorithm (say) $\mathcal{A}^+$. The idea is very simple: given an input stream f , at any time t , $\mathcal{A}^+$ runs an independent instance of algorithm $\mathcal{A}$ on the differential dataset at time t , that is ∇f_t , and produces the differential synthetic dataset ∇g_t . It can be shown that if $\mathcal{A}$ satisfies (offline) ϵ -differential privacy, then $\mathcal{A}^+$ satisfies ϵ -differential privacy as per Definition 3.4. $\mathcal{A}$ can be any differentially private algorithm that generates a synthetic dataset. For our current discussion, we create our baseline streaming algorithm by fixing the offline mechanism $\mathcal{A}$ as the MWEM algorithm [11]. Let us refer to the streaming version of MWEM as *StreamingMWEM* and we present the complete algorithm in Algorithm 2.

THEOREM 5.1 (ACCURACY OF STREAMINGMWEM). *At any time $t \in \mathbb{N}$, *StreamingMWEM* (Algorithm 2) is*

$$\left(\mathcal{O}\left(|f_t|^{2/3} \left(\frac{(t \log t) \ln |\mathcal{X}| \ln |\mathcal{Q}|}{\epsilon \beta}\right)^{1/3}\right), \beta \right)$$

accurate with respect to the set of marginal queries $\mathcal{Q}$.

We provide the proof of Theorem 5.1 in Appendix A.

6 PROPOSED ALGORITHM

In this section, we present our proposed algorithm.

6.1 Outline

In a nutshell, our algorithm also follows the *select, measure, fit, and iterate* paradigm described in Algorithm 1. At any time $t \in \mathbb{N}$, the

Algorithm 2 Baseline algorithm: StreamingMWEM

- 1: **Input:** An input data stream f , an ordered set of marginal queries $\mathcal{Q}$, number of marginals to select at any time k , the privacy budget ϵ .
 - 2: **Output:** A synthetic stream g .
 - 3: Initialize $g(0, x) \leftarrow 1$ for all $x \in \mathcal{X}$.
 - 4: **for** $t = 1, 2, \dots$ **do**
 - 5: Set $I_{t,0} \leftarrow \emptyset$.
 - 6: **for** $l = 1, 2, \dots, k$ **do**
 - 7: Set $J_{t,l} \leftarrow [|\mathcal{Q}|] \setminus I_{t,l-1}$; as query indices not selected.
 - 8: Set $e_{t,l} \leftarrow \left(|q_i(\nabla f_t) - q_i(h_{t,l-1})| \right)_{i \in J_{t,l}}$.
 - 9: // Exponential Mechanism
 - 10: Sample a query index $\eta_{t,l}$ such that for any $i \in J_{t,l}$,

$$\mathbb{P}\{\eta_{t,l} = i\} \propto \exp\left(\frac{\epsilon}{2k}(e_{t,l})_i\right).$$
 - 11: Using j as a shorthand for $\eta_{t,l}$.
 - 12: Set $I_{t,l} \leftarrow I_{t,l-1} \cup \{j\}$.
 - 13: // Laplace Mechanism
 - 14: Sample $\delta_{t,l} \sim \text{Lap}\left(\frac{2k}{\epsilon}\right)$.
 - 15: Set $m(t, j) \leftarrow q_j(\nabla f_t) + \delta_{t,l}$.
 - 16: // Multiplicative weights
 - 17: Set $h_{t,l}$ as $|\nabla f_t|$ times the distribution that satisfies

$$h_{t,l}(x) \propto h_{t,l-1}(x) \cdot \exp\left(q_j(x) \cdot \frac{m_j - q_j(h_{t,l-1})}{2|\nabla f_t|}\right)$$
 - 18: Set $g_t \leftarrow g_{t-1} + \text{avg}_{l \in [k]} h_{t,l}$.
-

goal is to ensure that f_t and g_t are close to each other as evaluated using the queries in the enumerated set $\mathcal{Q}$. We start with a dataset $h_{t,0} = g_{t-1}$ and update it over k iterations from $h_{t,0}, h_{t,1}, \dots$, to $h_{t,k}$. At any iteration $l \in [k]$, we select a query index $\eta_{t,l} \in [|\mathcal{Q}|]$ for which our dataset $h_{t,l-1}$ has approximately the highest error when compared to f_t . We will discuss how exactly this selection is done soon, but for now, let us accept it as a black-box. At the end of the k iterations, g_t is set to some aggregate of the datasets $h_{t,1}, h_{t,2}, \dots$, and $h_{t,k}$. We present our proposed method as a meta-algorithm in Algorithm 3.

6.2 Measure

Let $m : \mathbb{N} \times [|\mathcal{Q}|] \rightarrow \mathbb{R}$ be a map such that $m(t, i)$ denotes our differentially private approximation of $q_i(f_t)$, that is the value of query $q_i \in \mathcal{Q}$ at time t . Since a single query may be selected at multiple time instances, we use a counter algorithm to measure the value of the query efficiently over time. We associate each query in $\mathcal{Q}$ with an instance of some counter Algorithm, say $\mathcal{A}_{Counter}$. Consider a query $q_i \in \mathcal{Q}$ and let C_i be its corresponding counter. We use the notation $C_i(t)$ to conveniently refer to the value of the counter C_i at time t . Let $N_i(t) \subseteq [t]$ be the time instances until time t when the query q_i was selected to be measured using the true data. Also, let $\bar{N}_i(t) := [t] \setminus N_i(t)$ be the time instances until

Algorithm 3 Main algorithm: streaming differentially private synthetic tabular stream

- 1: **Input:** An input data stream f , an ordered set of marginal queries Q , number of marginals to select at any time k , the privacy budget ϵ , a counter algorithm $\mathcal{A}_{Counter}$, and a subroutine $\mathcal{A}_{Dataset}$ to find a dataset given noisy query measurements.
 - 2: **Output:** A synthetic stream g .
 - 3: Initialize $C_1, C_2, \dots, C_{|Q|}$ as independent instances of the counter algorithm $\mathcal{A}_{Counter}$, one for each query in the set Q , with privacy budget $\epsilon/2k$.
 - 4: Initialize $g(0, x) \leftarrow 1$ for all $x \in \mathcal{X}$.
 - 5: Initialize $m(0, i) \leftarrow 0$ for all $i \in [|Q|]$; query measurements of selected queries
 - 6: Initialize $r(0, i) \leftarrow 0$ for all $i \in [|Q|]$; remainder of query value for times when the query is not selected.
 - 7: **for** $t = 1, 2, \dots$ **do**
 - 8: Set $I_{t,0} \leftarrow \emptyset$.
 - 9: **for** $l = 1, 2, \dots, k$ **do**
 - 10: Set $J_{t,l} \leftarrow [|Q|] \setminus I_{t,l-1}$; as query indices not selected.
 - 11: Set $e_{t,l} \leftarrow \left(|q_i(\nabla f_t + g_{t-1}) - q_i(h_{t,l-1})| \right)_{i \in J_{t,l}}$.
 - 12: $\eta_{t,l} \leftarrow \text{ExponentialMechanism}(e_{t,l}, \epsilon/2k)$.
 - 13: Using shorthand j for $\eta_{t,l}$.
 - 14: Set $I_{t,l} \leftarrow I_{t,l-1} \cup \{j\}$.
 - 15: Invoke counter subroutine C_j with input ∇f_t .
 - 16: Set $r(t, j) \leftarrow r(t-1, j)$.
 - 17: Set $m(t, j) \leftarrow C_j + r(t, j)$.
 - 18: Set $h_{t,l} \leftarrow \mathcal{A}_{Dataset} \left(\{(q_i, m(t, i))\}_{i \in I_{t,l}}, h_{t,l-1} \right)$.
 - 19: Set $g_t \leftarrow \text{avg}_{l \in [k]} h_{t,l}$.
 - 20: Set $C_i(t) \leftarrow C_i(t-1)$ for all $i \in [|Q|] \setminus I_{t,k}$.
 - 21: Set $r(t, i) \leftarrow q_i(g_t) - C_i(t)$ for all $i \in [|Q|] \setminus I_{t,k}$.
-

time t at which query q_i was not selected. Then the output $C_i(t)$ of the counter algorithm is based solely on the stream $\nabla f_{N_i(t)}$.

However, to generate the dataset g_t we need an approximate measurement of the value $q_i(f_t)$. In other words, we are missing the measurement of the query on times $\bar{N}_i(t)$ when the index i was not selected. At any such time $\tau \in \bar{N}_i(t)$, since q_i was not selected, we assume that the query value $q_i(g_\tau)$ on the synthetic dataset g_τ is close to the true value $q_i(f_\tau)$. We create a map $r : \mathbb{N} \times [|Q|] \rightarrow \mathbb{R}$ such that $r(t, i)$ denotes our differentially private approximation of the value of query q_i over times in $\bar{N}_i(t)$. Assuming $r(0, i) = 0$, we define $r(t, i)$ for any $t \in \mathbb{N}$ as,

$$r(t, i) = \begin{cases} q_i(g_t) - C_i(t), & t \in N_i(t), \\ r(t-1, i), & \text{otherwise.} \end{cases}$$

Finally, our differentially private approximation $m(t, i)$ of the query q_i at time t becomes

$$m(t, i) = C_i(t) + r(t, i).$$

6.3 Fit

At any time t and iteration l , Algorithm 3 uses the Algorithm $\mathcal{A}_{Dataset}$ as a subroutine to generate the synthetic dataset $h_{t,l}$ using the query

indices selected so far at time t , that is $\{\eta_{t,1}, \dots, \eta_{t,l}\}$, and their corresponding differentially private values $\{m(t, \eta_{t,1}), \dots, m(t, \eta_{t,l})\}$. $\mathcal{A}_{Dataset}$ can be any algorithm and is not required to satisfy differential privacy.

6.4 Select

We are finally ready to talk about query selection. During iteration l of time t , we want to select the query with maximum error over the synthetic dataset $h_{t,l-1}$ as compared to the true dataset f_t . However, accessing $q(f_t)$ results in high sensitivity. Indeed a simple change at some time $\tau \in \mathbb{N}$ can affect the selection at all times $t > \tau$.

To control the sensitivity, we follow a trick inspired by [17] and approximate f_t as $g_{t-1} + \nabla f_t$ for selection. For any query $q_i \in Q$, $l \in [k]$, and $t \in \mathbb{N}$, we define

$$e_{t,l} := \left(|q_i(\nabla f_t + g_{t-1}) - q_i(h_{t,l-1})| \right)_{i \in [|Q|]}.$$

Finally, we use the Exponential Mechanism as defined in Definition 3.8 for selecting a query index $\eta_{t,l}$ given the vector of query utilities $e_{t,l}$. Note that Algorithm 3 does not find the error for all queries but instead only for queries that have not been chosen so far at iteration l of time t (whose indices are in the set $J_{t,l}$).

THEOREM 6.1 (PRIVACY OF ALGORITHM 3). *If the Algorithms $\mathcal{A}_{Counter}$ and $\mathcal{A}_{Dataset}$ satisfy differential privacy, then Algorithm 3 is ϵ -differentially private.*

PROOF. Note that Algorithm 3 is an instance of the selective counting algorithm from [17]. Moreover, since we split the budget as $\epsilon/2$ for the selection (Exponential Mechanism) subroutine and the remaining $\epsilon/2$ for the counters at any time t , Algorithm 3 satisfies ϵ -differential privacy. $\square$

7 EXPERIMENTS AND RESULTS

We explore the performance of our algorithm on real-world datasets. We use the Probabilistic Graphical Model (PGM) [21] as the subroutine $\mathcal{A}_{Dataset}$ for both the baseline and proposed algorithm. We briefly introduced PGM in Section 4.2.2. Similar to [17], we found that Simple counter works the best for our use case as we do not have a very large time horizon for the stream. So, for the experiments in this section, we use the Simple counter as the subroutine $\mathcal{A}_{Counter}$. In the subsequent subsections, we discuss the details of over experiments.

7.1 Datasets

7.1.1 Eviction. The Eviction Dataset [6] contains eviction notices filed with the San Francisco Rent Board from January 1, 1997. The dataset has an attribute “File Date” which represents the date on which the eviction notice was filed with the Rent Board of Arbitration. We use the value of this attribute to construct our time index for the stream. We fix a synthetic data release frequency as weekly or bi-weekly, and based on the attribute File Date and this frequency, create the time index for our data. In the results, we refer to the corresponding streams as *Eviction-weekly* and *Eviction-bi-weekly* respectively. We limit the dataset to 3 location-based categorical attributes - “Eviction Notice Source Zipcode”, “Supervisor District”, and “Neighborhoods”, and all binary attributes such as - “Non Payment”, “Breach”, and “Illegal Use”. The data space

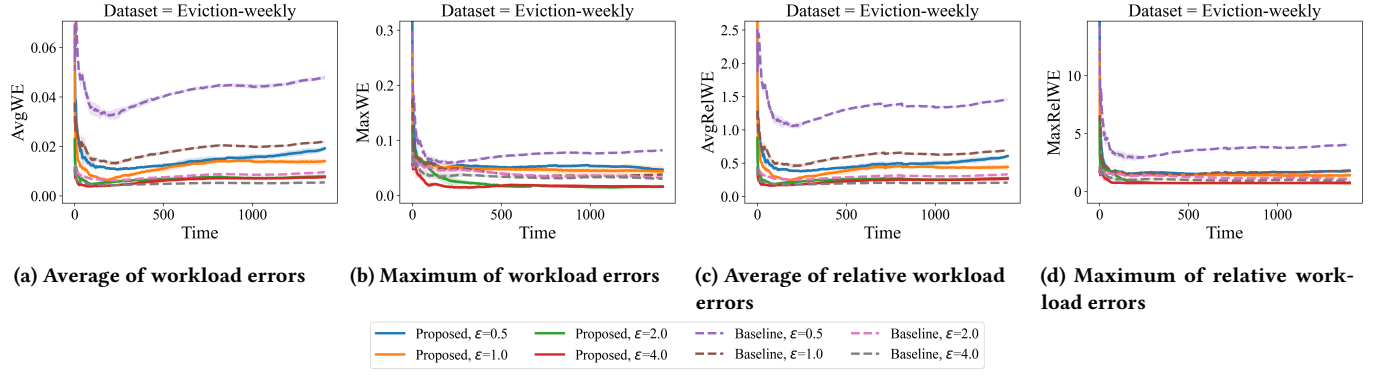


Figure 1: Metrics over time to compare the baseline and proposed method for the Eviction-weekly dataset.

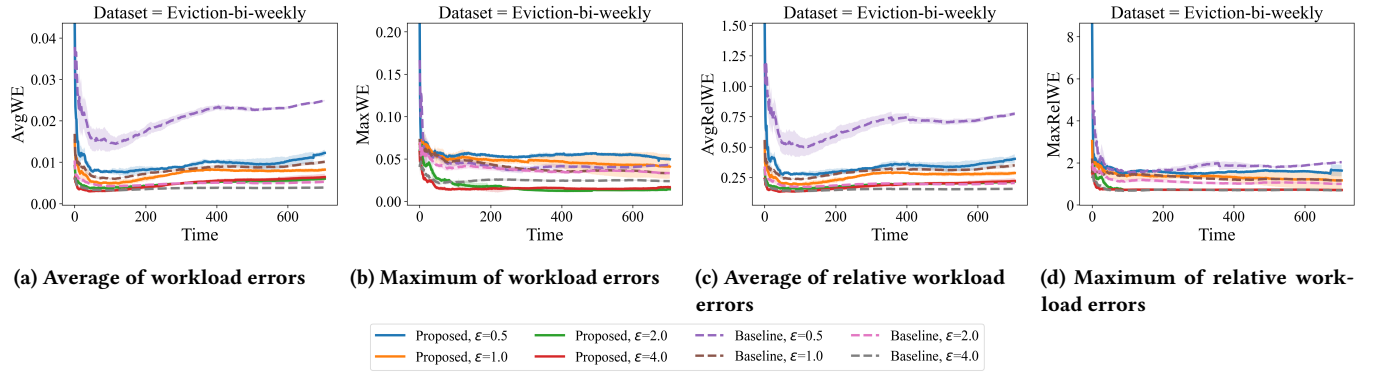


Figure 2: Metrics over time to compare the baseline and proposed method for the Eviction-bi-weekly dataset.

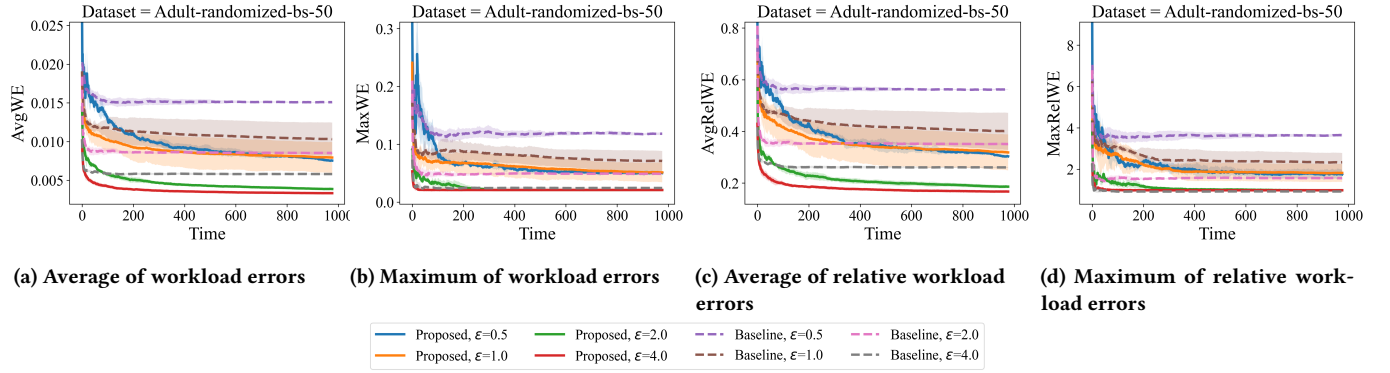


Figure 3: Metrics over time to compare the baseline and proposed method for the Adult-randomized-bs-50 dataset.

of the resulting dataset was 22 dimensional with 19 binary and 3 categorical attributes.

7.1.2 Adult. The *Adult* dataset [2] has been used extensively in previous research in this area and so we also use this dataset. We use a processed version of the dataset released in the source code provided by [18]. Note that there is no notion of time in this dataset. We artificially create time in two ways which results in the following two streams: (1) *Adult-randomized*: we fix a constant batch size say

B , that is the number of points that are added at each time, then at any time t , we simply add B points to our stream that are sampled uniformly at random from the *Adult* dataset without replacement; (2) *Adult-ordered*: the stream is also created by adding a fixed batch of B points, except the points are selected deterministically, where we first sort the entire dataset in increasing order and then add the next B points based on the indices at any time. We explore a small and large value of batch size B as 50 and 200 respectively. Note that

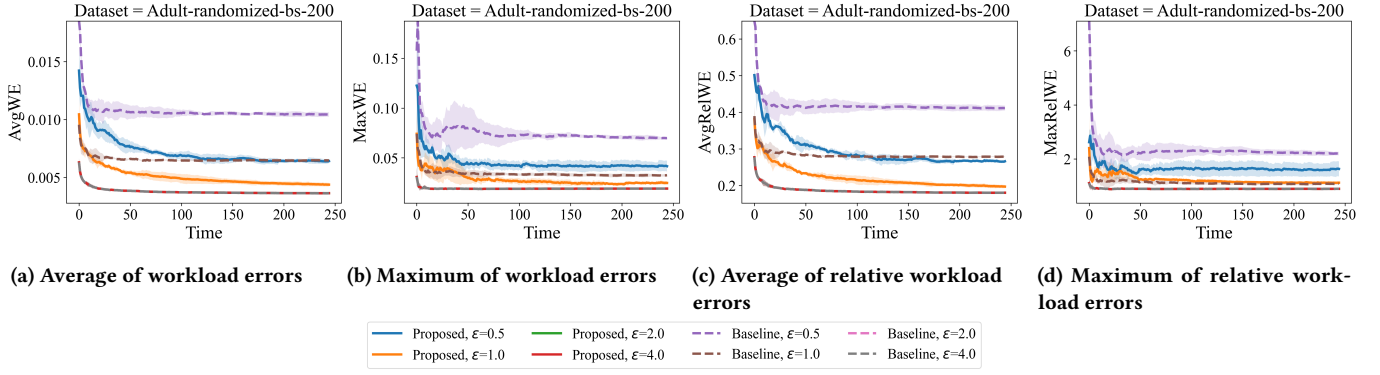


Figure 4: Metrics over time to compare the baseline and proposed method for the Adult-randomized-bs-200 dataset.

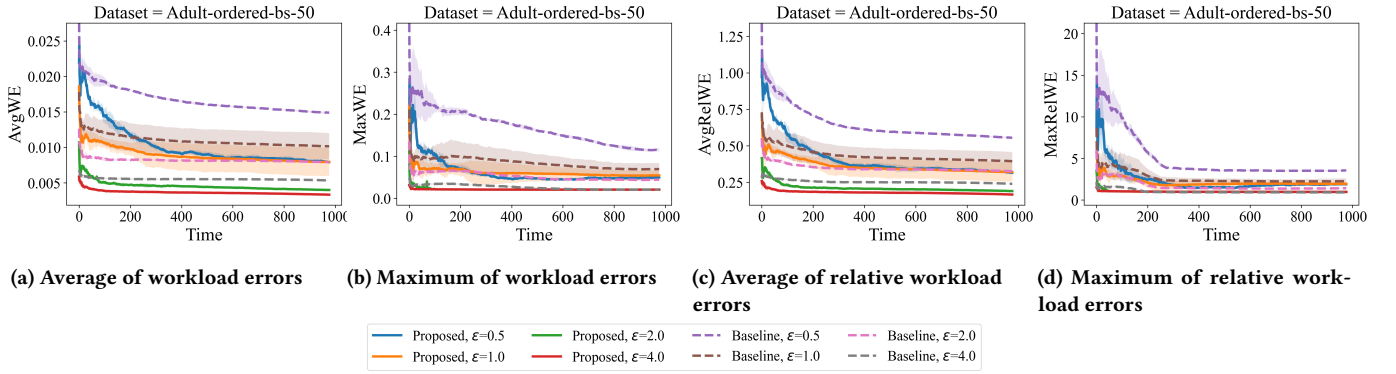


Figure 5: Metrics over time to compare the baseline and proposed method for the Adult-ordered-bs-50 dataset.

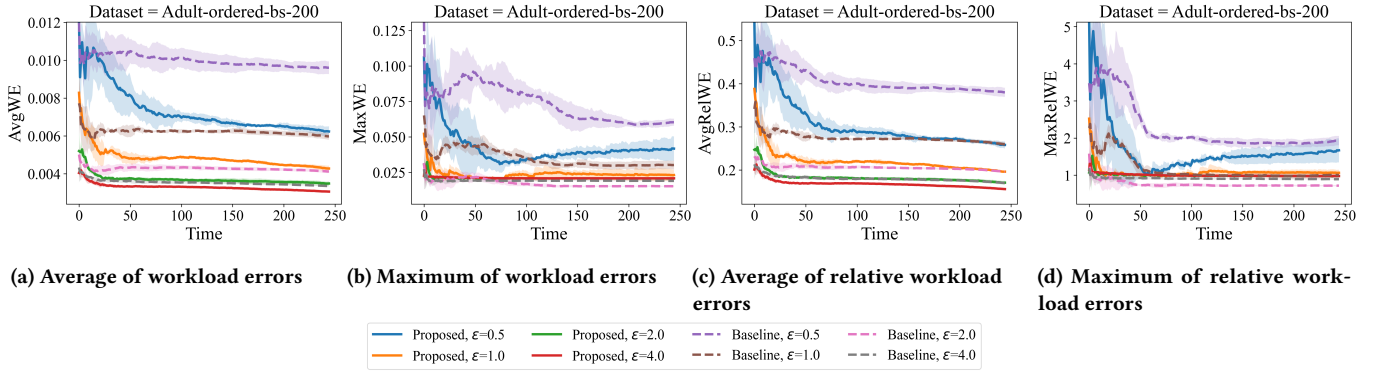


Figure 6: Metrics over time to compare the baseline and proposed method for the Adult-ordered-bs-200 dataset.

the data stream Adult-ordered is interesting in the sense that the query values may change very drastically over time.

7.2 Workload of queries

In the experiments, we aim to preserve all 2-way marginals on the space $\mathcal{X}$. However, instead of using the set of all 2-way marginals queries as Q , we use the set of all 2-way *workloads*.

Definition 7.1 (Workload). A k -way workload W is defined by a tuple of k column indices, $(c_1, c_2, \dots, c_k)$ such that $c_i \in [p]$ for all $i \in [k]$ and $c_1 < c_2 < \dots < c_k$. Let $columns(W) = (c_1, c_2, \dots, c_k)$. Then, W is an ordered collection of all k -way marginal queries on $columns(W)$, where the order is taken as the lexicographic order of the values corresponding to queries. Furthermore, we denote the number of marginal queries in W with $|W|$. The value of a workload over a dataset $f : \mathcal{X} \rightarrow \mathbb{N}_0$ is defined as the tuple $W(f) = (q(f))_{q \in W}$.

Table 1: Query error metrics for the baseline and proposed methods for the *Eviction-weekly*, *Eviction-bi-weekly*, and *Adult-randomized-bs-50* datasets.

Dataset	ϵ	Metric	Baseline	Proposed
Eviction-weekly	0.5	AvgRelWE	1.4523	0.6024
		AvgWE	0.0478	0.0191
		MaxRelWE	4.0261	1.7817
		MaxWE	0.218	0.0467
	1.0	AvgRelWE	0.6906	0.4409
		AvgWE	0.0219	0.014
		MaxRelWE	1.7988	1.3978
		MaxWE	0.0378	0.0436
	2.0	AvgRelWE	0.3309	0.2746
		AvgWE	0.0096	0.0079
		MaxRelWE	1.1009	0.7573
		MaxWE	0.032	0.0154
	4.0	AvgRelWE	0.2066	0.2632
		AvgWE	0.0054	0.0075
		MaxRelWE	0.923	0.71
		MaxWE	0.0303	0.0162
Eviction-bi-weekly	0.5	AvgRelWE	0.7719	0.4009
		AvgWE	0.0248	0.0121
		MaxRelWE	2.0291	1.6396
		MaxWE	0.043	0.0499
	1.0	AvgRelWE	0.3487	0.2864
		AvgWE	0.01	0.0082
		MaxRelWE	1.1644	1.1651
		MaxWE	0.0331	0.0411
	2.0	AvgRelWE	0.2046	0.2166
		AvgWE	0.0053	0.006
		MaxRelWE	1.0008	0.71
		MaxWE	0.0333	0.0141
	4.0	AvgRelWE	0.1571	0.2207
		AvgWE	0.0039	0.0064
		MaxRelWE	0.6945	0.71
		MaxWE	0.0238	0.0167
Adult-randomized-bs-50	0.5	AvgRelWE	0.5624	0.3035
		AvgWE	0.0151	0.0075
		MaxRelWE	3.6469	1.7575
		MaxWE	0.1182	0.0504
	1.0	AvgRelWE	0.4005	0.3191
		AvgWE	0.0103	0.0079
		MaxRelWE	2.3292	1.8162
		MaxWE	0.0712	0.0514
	2.0	AvgRelWE	0.351	0.1863
		AvgWE	0.0085	0.0039
		MaxRelWE	1.5646	0.977
		MaxWE	0.0488	0.0208
	4.0	AvgRelWE	0.2606	0.1673
		AvgWE	0.0058	0.0033
		MaxRelWE	0.9187	0.9778
		MaxWE	0.0243	0.0208

Table 2: Query error metrics for the baseline and proposed methods for the *Adult-randomized-bs-200*, *Adult-ordered-bs-50*, and *Adult-ordered-bs-200* datasets.

Dataset	ϵ	Metric	Baseline	Proposed
Adult-randomized-bs-200	0.5	AvgRelWE	0.4113	0.2658
		AvgWE	0.0104	0.0064
		MaxRelWE	2.1987	1.6246
		MaxWE	0.0698	0.0419
	1.0	AvgRelWE	0.2786	0.1975
		AvgWE	0.0065	0.0044
		MaxRelWE	1.0789	1.1166
		MaxWE	0.0323	0.0249
	2.0	AvgRelWE	0.1802	0.1802
		AvgWE	0.0036	0.0036
		MaxRelWE	0.8907	0.8907
		MaxWE	0.0191	0.0191
	4.0	AvgRelWE	0.1802	0.1802
		AvgWE	0.0036	0.0036
		MaxRelWE	0.8907	0.8907
		MaxWE	0.0191	0.0191
Adult-ordered-bs-50	0.5	AvgRelWE	0.5559	0.3165
		AvgWE	0.0149	0.008
		MaxRelWE	3.5589	1.9028
		MaxWE	0.1157	0.0497
	1.0	AvgRelWE	0.3953	0.3214
		AvgWE	0.0102	0.008
		MaxRelWE	2.2663	1.9528
		MaxWE	0.0695	0.0547
	2.0	AvgRelWE	0.3264	0.1904
		AvgWE	0.0079	0.004
		MaxRelWE	1.4234	0.9798
		MaxWE	0.0444	0.0208
	4.0	AvgRelWE	0.2402	0.1667
		AvgWE	0.0053	0.0033
		MaxRelWE	0.9192	0.978
		MaxWE	0.0211	0.0208
Adult-ordered-bs-200	0.5	AvgRelWE	0.381	0.2593
		AvgWE	0.0096	0.0063
		MaxRelWE	1.9203	1.6522
		MaxWE	0.0601	0.0413
	1.0	AvgRelWE	0.2608	0.1972
		AvgWE	0.006	0.0043
		MaxRelWE	1.006	1.067
		MaxWE	0.0302	0.0232
	2.0	AvgRelWE	0.1969	0.1711
		AvgWE	0.0041	0.0035
		MaxRelWE	0.7189	0.9752
		MaxWE	0.0152	0.0208
	4.0	AvgRelWE	0.1705	0.1563
		AvgWE	0.0034	0.0031
		MaxRelWE	0.8939	0.9776
		MaxWE	0.019	0.0208

Using workloads instead of marginal queries is a common practice in the literature and is sometimes referred to as the *marginal trick* [18]. To see the advantage, note that the collections of queries in a workload create a disjoint space in the sense that a user can contribute data in at most one of them. Thus we can use the Parallel Composition of differential privacy and do not have to divide the privacy budget among the queries in a workload when estimating them. In other words, estimating any workload is a histogram query with sensitivity 1 for any pair of neighboring datasets. This is true even though there are (likely) multiple queries in the workload.

Note that since we are now using workloads instead of marginal queries, the following changes are needed in Algorithm 3 to ensure compatibility,

- (1) Q is an ordered set of workloads such that for any $i \in |Q|$, $|q_i|$ denotes the number of marginal queries in q_i ;
- (2) each counter instance C_i , corresponding to the workload q_i , is a multi-dimensional counter of dimension $|q_i|$, as defined in [17];
- (3) at an iteration l of time t , we define $e_{t,l}$ as

$$e_{t,l} = \begin{pmatrix} \frac{1}{|q_i|} \|q_i(\nabla f_t + g_{t-1}) - q_i(h_{t,l-1})\|_{\ell_1} \\ -|X_{columns(q_i)}| \end{pmatrix}_{i \in J_{t,l}},$$

where $|X_{columns(q_i)}|$ is a bias correction term accounting for the number of queries in a workload;

- (4) for 2-way workloads, the resulting sensitivity of the exponential mechanism is $\frac{1}{4}$.

7.3 Evaluation metrics

We measure the performance of our algorithm using the error introduced by the generated synthetic data in answering queries. Since we use workloads in our algorithm, we measure the error in queries grouped by the workloads. This results in two levels of aggregation, one for queries within a workload and the second over different workloads.

Let us assume that we are looking for error at time t for the synthetic stream g given the input stream f and the set of workloads Q . Then, we define the workload errors as follows:

- (1) *Workload error (WE)*: For any workload W , we define workload error at time t as the average error in queries within W , that is

$$WE(W, f, g, t) := \frac{1}{|W|} \sum_{q \in W} |q(f_t) - q(g_t)|.$$

- (2) *Relative workload error (RelWE)*: Similar to workload error, for any workload W , we define relative workload error at time t as the average relative error in queries within W , that is

$$RelWE(W, f, g, t) := \frac{1}{|W|} \sum_{q \in W} \left| \frac{q(f_t) - q(g_t)}{q(f_t)} \right|.$$

Our final metric is the aggregated (average and maximum) error over all workloads. Given a set Q containing workloads, an input stream f , and a synthetic stream g , at any time t we define:

- (1) *Average over workload errors (AvgWE)*: as the average workload error over workloads in Q , that is

$$AvgWE(Q, f, g, t) := \frac{1}{|Q|} \sum_{W \in Q} WE(W, f, g, t).$$

- (2) *Maximum over workload errors (MaxWE)*: as the maximum workload error over workloads in Q , that is

$$MaxWE(Q, f, g, t) := \max_{W \in Q} (WE(W, f, g, t)).$$

- (3) *Average over relative workload errors (AvgRelWE)*: as the average relative workload error over workloads in Q , that is

$$AvgRelWE(Q, f, g, t) := \frac{1}{|Q|} \sum_{W \in Q} RelWE(W, f, g, t).$$

- (4) *Maximum over relative workload errors (MaxRelWE)*: as the maximum relative workload error over workloads in Q , that is

$$MaxRelWE(Q, f, g, t) := \max_{W \in Q} (RelWE(W, f, g, t)).$$

7.4 Results

Figures 1, 2, 3, 4, 5, and 6 provide our results for the Eviction-weekly, Eviction-bi-weekly, Adult-randomized-bs-50, Adult-randomized-bs-200, Adult-ordered-bs-50, and Adult-ordered-bs-200 datasets respectively. The horizontal axis in all of these figures represents time and the vertical axis is the metric mentioned in the y-axis label of the corresponding subfigure. At the beginning of time, we see a large variance in the metrics, and the proposed method has a larger error in some experiments. However, as the time index increases, in most cases, our method outperforms the baseline across various datasets, metrics, and privacy budgets. Moreover, we observe that among the various metrics, the most variation occurs in metrics that measure the worst-case errors: MaxWE and MaxRelWE. We also provide these metrics in a tabular view in Tables 1 and 2 to facilitate the comparison. These tables contain the value of each metric averaged over the last 10 time steps for various datasets and privacy budgets ϵ .

8 A NEW (UNBOUNDED) BLOCK COUNTER

We extend the Two-Level counter mechanism (also referred to as Block counter) due to [4] to unbounded streams. We present it formally in Algorithm 4. The idea is similar to how the bounded Binary Mechanism is extended to the unbounded Hybrid Mechanism in [4]. As shown in [4], an optimal block size of the Bounded Block Counter for a stream of size T is $\sqrt{T}$. The key idea is to partition the time dimension of the stream $f : \mathbb{N} \rightarrow \mathbb{R}$ into intervals of size 4, 9, 16, ... (that is perfect squares), and within each of the corresponding intervals, we use a bounded block counter of block size 2, 3, 4, ... respectively.

THEOREM 8.1 (PRIVACY OF UNBOUNDED BLOCK COUNTER). *The unbounded block counter, as presented in Algorithm 4, satisfies ϵ -differential privacy.*

PROOF. Note, Algorithm 4 is exactly the block counter algorithm, except the size of the block changes over time. However, the change in the block size is independent of the input data stream. Hence,

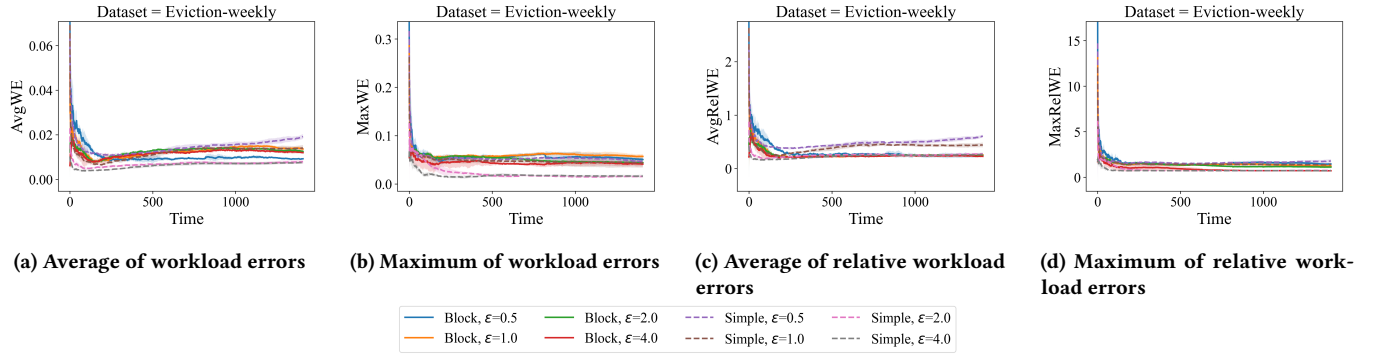


Figure 7: Metrics over time to compare the performance of simple and block counters for the Eviction-weekly dataset.

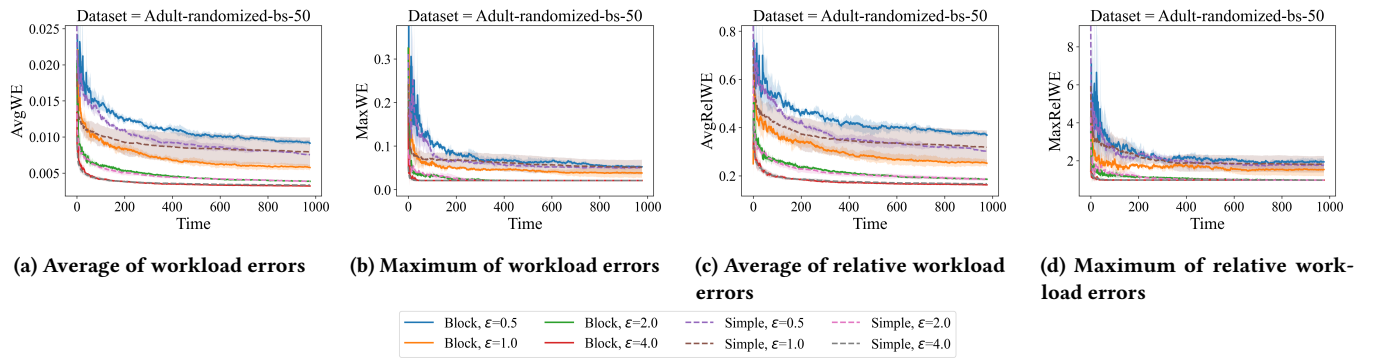


Figure 8: Metrics over time to compare the performance of simple and block counters for the Adult-randomized-bs-50 dataset.

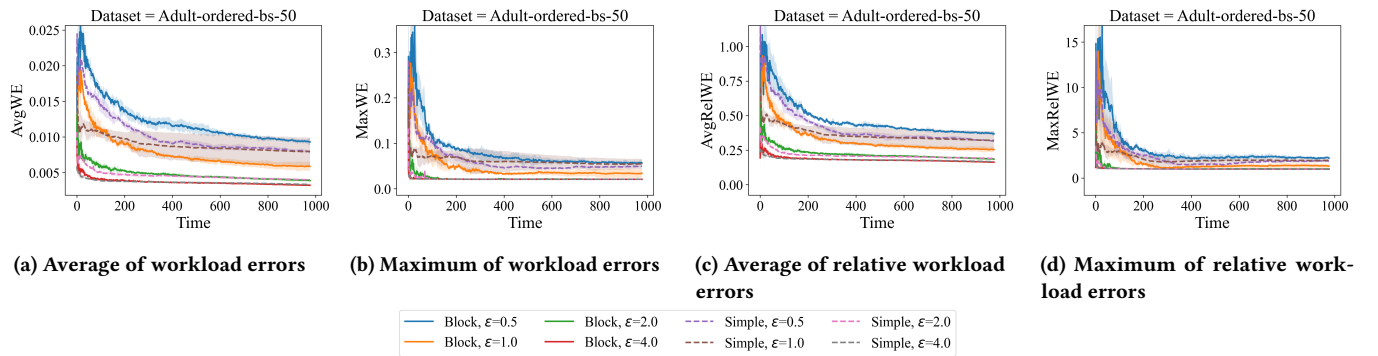


Figure 9: Metrics over time to compare the performance of simple and block counters for the Adult-ordered-bs-50 dataset.

similar to the Block counter, Algorithm 4 is ϵ -differentially private. $\square$

8.1 Results

In this section, we present our results for empirical analysis of the proposed unbounded block counter (Algorithm 4) as compared to the simple counter when used as the subroutine $\mathcal{A}_{Dataset}$ in Algorithm 3. Based on the evidence in [17], we know that the block counter performs better than the simple counter only after

sufficiently large time t , hence we only use the datasets Eviction-weekly, Adult-ordered-bs-50, and Adult-randomized-bs-50 in our experiments. The length of time horizons for these datasets are 1409, 977, and 977 respectively.

We present the findings of our experiments in Figures 7, 8, and 9 which show various error metrics over time, analogous to Section 7.4. We also provide a tabular view of these metrics in Table 3. Let us first focus on the Adult dataset and privacy budget $\epsilon \geq 1$. Using the block counter in Algorithm 3 is typically better than using the simple counter. However, for $\epsilon = 0.5$, we see that the simple

Table 3: Query error metrics comparing the simple and block counters in the proposed method for the *Eviction-weekly*, *Adult-randomized-bs-50*, and *Adult-ordered-bs-50* datasets.

Dataset	ϵ	Metric	Simple	Block
Eviction-weekly	0.5	AvgRelWE	0.6024	0.2545
		AvgWE	0.0191	0.0092
		MaxRelWE	1.7817	1.4485
		MaxWE	0.0467	0.0507
	1.0	AvgRelWE	0.4409	0.2553
		AvgWE	0.014	0.0136
		MaxRelWE	1.3978	1.2361
		MaxWE	0.0436	0.057
	2.0	AvgRelWE	0.2746	0.2376
		AvgWE	0.0079	0.0126
		MaxRelWE	0.7573	1.121
		MaxWE	0.0154	0.044
	4.0	AvgRelWE	0.2632	0.2284
		AvgWE	0.0075	0.012
		MaxRelWE	0.71	0.71
		MaxWE	0.0162	0.0411
Adult-randomized-bs-50	0.5	AvgRelWE	0.3035	0.3718
		AvgWE	0.0075	0.0092
		MaxRelWE	1.7575	1.9395
		MaxWE	0.0504	0.0525
	1.0	AvgRelWE	0.3191	0.2534
		AvgWE	0.0079	0.0058
		MaxRelWE	1.8162	1.5294
		MaxWE	0.0514	0.0383
	2.0	AvgRelWE	0.1863	0.1862
		AvgWE	0.0039	0.0039
		MaxRelWE	0.977	0.9804
		MaxWE	0.0208	0.0208
	4.0	AvgRelWE	0.1673	0.1629
		AvgWE	0.0033	0.0032
		MaxRelWE	0.9778	0.9779
		MaxWE	0.0208	0.0208
Adult-ordered-bs-50	0.5	AvgRelWE	0.3165	0.3698
		AvgWE	0.008	0.0093
		MaxRelWE	1.9028	2.2472
		MaxWE	0.0497	0.0572
	1.0	AvgRelWE	0.3214	0.2552
		AvgWE	0.008	0.0059
		MaxRelWE	1.9528	1.3285
		MaxWE	0.0547	0.0338
	2.0	AvgRelWE	0.1904	0.1864
		AvgWE	0.004	0.0039
		MaxRelWE	0.9798	1.0114
		MaxWE	0.0208	0.0208
	4.0	AvgRelWE	0.1667	0.1632
		AvgWE	0.0033	0.0032
		MaxRelWE	0.978	0.978
		MaxWE	0.0208	0.0208

Algorithm 4 Unbounded Block Counter

```

1: Input: An input data stream  $f : \mathbb{N} \rightarrow \mathbb{R}$ , the privacy budget  $\epsilon$ .
2: Output: A synthetic stream  $g : \mathbb{N} \rightarrow \mathbb{R}$ .
3: Initialize partition size  $T \leftarrow 4$ .
4: Initialize block size  $B \leftarrow 2$ .
5: Last block value  $\alpha_{lastBlock} \leftarrow 0$ .
6: True value within block  $\alpha_{trueInBlock} \leftarrow 0$ .
7: Synthetic value within block  $\alpha_{synthInBlock} \leftarrow 0$ .
8: Time when the last partition changed  $t_{atPartition} \leftarrow 0$ .
9: Set  $g(0) = 0$ .
10: for  $t = 1, 2, \dots$  do
11:   Set  $\delta \leftarrow t - t_{atPartition}$ .
12:   Update  $\alpha_{trueInBlock} \leftarrow \alpha_{trueInBlock} + f(t)$ .
13:   if  $\delta = kB$  for some  $k \in \mathbb{Z}$  then
14:     Update  $\alpha_{lastBlock} \leftarrow \alpha_{lastBlock} + \alpha_{trueInBlock} + \text{Lap}\left(\frac{2}{\epsilon}\right)$ .
15:     Update  $\alpha_{trueInBlock} \leftarrow 0$  and  $\alpha_{synthInBlock} \leftarrow 0$ .
16:     Set  $g(t) \leftarrow \alpha_{lastBlock}$ .
17:     if  $\delta = T$  then
18:       Update  $t_{atPartition} \leftarrow t$ .
19:       Update  $B \leftarrow B + 1$  and  $T \leftarrow B^2$ .
20:   else
21:     Update  $\alpha_{synthInBlock} \leftarrow \alpha_{synthInBlock} + f(t) + \text{Lap}\left(\frac{2}{\epsilon}\right)$ .
22:     Set  $g(t) \leftarrow \alpha_{lastBlock} + \alpha_{synthInBlock}$ .
23:   Release  $g(t)$ .

```

counter performs better. The results of the experiments over the Eviction dataset do not yield a clear conclusion whether the block counter is better than the simple counter for any particular ϵ . We believe that the high variance of the block counter at the beginning of time, together with the selection error due to the Exponential mechanism, leads to such behavior.

9 CONCLUSION

In this work, we discuss the task of streaming differentially private high-dimensional synthetic data that accurately represents the true data over a set of marginal queries. Our focus is on developing an algorithm that can be used in practical applications and is better than the naive algorithm of running independent instances of offline algorithms on differential data stream at any time. We build upon existing research in offline synthetic data generation and counters for streaming algorithms to create a framework for the task. We also show with experiments over real-world datasets that our method outperforms the baseline.

ACKNOWLEDGMENTS

G.K. and T.S. acknowledge support from NSF DMS-2027248, NSF DMS-2208356, and NIH R01HL16351. R.V. acknowledges support from NSF DMS-1954233, NSF DMS-2027299, and NSF+Simons Research Collaborations on the Mathematical and Scientific Foundations of Deep Learning.

REFERENCES

- [1] Sergul Ayidore, William Brown, Michael Kearns, Krishnam Kenthapadi, Luca Melis, Aaron Roth, and Ankit A Siva. 2021. Differentially Private Query Release Through Adaptive Projection. In *Proceedings of the 38th International Conference*

- on Machine Learning (*Proceedings of Machine Learning Research*, Vol. 139). PMLR, 457–467. <https://proceedings.mlr.press/v139/aydore21a.html>
- [2] Barry Becker and Ronny Kohavi. 1996. Adult. Published: UCI Machine Learning Repository.
 - [3] Mark Bun, Marco Gaboardi, Marcel Neunhoffer, and Wanrong Zhang. 2024. Continual Release of Differentially Private Synthetic Data from Longitudinal Data Collections. *Proceedings of the ACM on Management of Data* 2, 2 (May 2024), 94:1–94:26. <https://doi.org/10.1145/3651595>
 - [4] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. 2011. Private and Continual Release of Statistics. *ACM Transactions on Information and System Security* 14, 3 (Nov. 2011), 26:1–26:24. <https://doi.org/10.1145/2043621.2043626>
 - [5] Yan Chen, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2017. PeGaSus: Data-Adaptive Differentially Private Stream Processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1375–1388. <https://doi.org/10.1145/3133956.3134102>
 - [6] City and County of San Francisco. 2024. Eviction Notices | DataSF | City and County of San Francisco — data.sfgov.org. https://data.sfgov.org/Housing-and-Buildings/Eviction-Notices/5cei-gny5/about_data. https://data.sfgov.org/Housing-and-Buildings/Eviction-Notices/5cei-gny5/about_data [Accessed 25-01-2024].
 - [7] Cynthia Dwork. 2006. Differential privacy. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II* 33. Springer, 1–12.
 - [8] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. 2010. Differential privacy under continual observation. In *Symposium on the Theory of Computing*.
 - [9] Úlfar Erlingsson, Vasily Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. Association for Computing Machinery, New York, NY, USA, 1054–1067. <https://doi.org/10.1145/2660267.2660348> event-place: Scottsdale, Arizona, USA.
 - [10] Marco Gaboardi, Emilio Jesus Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. 2014. Dual Query: Practical Private Query Release for High Dimensional Data. In *Proceedings of the 31st International Conference on Machine Learning*. PMLR, 1170–1178. <https://proceedings.mlr.press/v32/gaboardi14.html> ISSN: 1938-7228.
 - [11] Moritz Hardt, Katrina Ligett, and Frank Mcsherry. 2012. A Simple and Practical Algorithm for Differentially Private Data Release. In *Advances in Neural Information Processing Systems*, Vol. 25. Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2012/hash/208e43f0e45c4c78cafadb83d2888cb6-Abstract.html>
 - [12] Yiyun He, Roman Vershynin, and Yizhe Zhu. 2024. Online Differentially Private Synthetic Data Generation. <https://doi.org/10.48550/arXiv.2402.08012> [cs, math, stat] version: 1.
 - [13] Monika Henzinger, A. R. Sricharan, and Teresa Anna Steiner. 2023. Differentially Private Histogram, Predecessor, and Set Cardinality under Continual Observation. <http://arxiv.org/abs/2306.10428> arXiv:2306.10428 [cs].
 - [14] Palak Jain, Iden Kalemaj, Sofya Raskhodnikova, Satchit Sivakumar, and Adam Smith. 2023. Counting Distinct Elements in the Turnstile Model with Differential Privacy under Continual Observation. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 4610–4623. https://proceedings.neurips.cc/paper_files/paper/2023/file/0ef1afa0daa888d695dcd5e9513bafa3-Paper-Conference.pdf
 - [15] Palak Jain, Sofya Raskhodnikova, Satchit Sivakumar, and Adam Smith. 2023. The Price of Differential Privacy under Continual Observation. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 14654–14678. <https://proceedings.mlr.press/v202/jain23b.html> ISSN: 2640-3498.
 - [16] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. 2018. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. <https://openreview.net/forum?id=S1zk9iRqF7>
 - [17] Girish Kumar, Thomas Strohmer, and Roman Vershynin. 2024. An Algorithm for Streaming Differentially Private Data. <https://doi.org/10.48550/arXiv.2401.14577> arXiv:2401.14577 [cs, math, stat].
 - [18] Terrance Liu, Giuseppe Vietri, and Steven Z. Wu. 2021. Iterative Methods for Private Synthetic Data: Unifying Framework and New Methods. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 690–702. https://proceedings.neurips.cc/paper_files/paper/2021/hash/0678c572b0d5597d2d4a6b5bd135754c-Abstract.html
 - [19] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2023. Optimizing Error of High-Dimensional Statistical Queries Under Differential Privacy. *Journal of Privacy and Confidentiality* 13, 1 (Aug. 2023). <https://doi.org/10.29012/jpc.791> Number: 1.
 - [20] Ryan McKenna, Gerome Miklau, and Daniel Sheldon. 2021. Winning the NIST Contest: A scalable and general approach to differentially private synthetic data. *Journal of Privacy and Confidentiality* 11, 3 (Dec. 2021). <https://doi.org/10.29012/jpc.778> Number: 3.
 - [21] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 4435–4444. <https://proceedings.mlr.press/v97/mckenna19a.html>
 - [22] Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. 2022. Benchmarking Differentially Private Synthetic Data Generation Algorithms. <https://doi.org/10.48550/arXiv.2112.09238> arXiv:2112.09238 [cs].
 - [23] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. 2019. Dp-cgan: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 0–0.
 - [24] Tianhao Wang, Joann Qiongna Chen, Zhikun Zhang, Dong Su, Yueqiang Cheng, Zhou Li, Ninghui Li, and Somesh Jha. 2021. Continuous Release of Data Streams under both Centralized and Local Differential Privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*. Association for Computing Machinery, New York, NY, USA, 1237–1253. <https://doi.org/10.1145/3460120.3484750>
 - [25] Chugui Xu, Ju Ren, Deyu Zhang, Yaoxue Zhang, Zhan Qin, and Kui Ren. 2019. GANobfuscator: Mitigating information leakage under GAN via differential privacy. *IEEE Transactions on Information Forensics and Security* 14, 9 (2019), 2358–2371. Publisher: IEEE.
 - [26] Xiang Yue, Huseyin A Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Huan Sun, David Levitan, and Robert Sim. 2022. Synthetic text generation with differential privacy: A simple and practical recipe. *arXiv preprint arXiv:2210.14348* (2022).
 - [27] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private Data Release via Bayesian Networks. *ACM Transactions on Database Systems* 42, 4 (Oct. 2017), 25:1–25:41. <https://doi.org/10.1145/3134428>

A ACCURACY OF BASELINE ALGORITHM

PROOF OF THEOREM 5.1. The below analysis is similar to the analysis of (offline) MWEM algorithm due to [11]. Let us focus the analysis on iteration l of time t .

Selection error: First, we will analyze the error in query selection at Step 9. Let $\text{maxerr}_{t,l}$ denote the maximum possible absolute difference between values of any query in Q as measured on $h_{t,l-1}$ and ∇f_t , that is,

$$\text{maxerr}_{t,l} = \max_{q \in Q} |q(h_{t,l-1}) - q(f_t)|. \quad (11)$$

At the l^{th} iteration at time t , we select the query with index j , where j is a shorthand for $e_{t,l}$. By the utility of exponential mechanism (Theorem 3.10) invoked with a privacy budget $\varepsilon/2k$ and sensitivity 1, for any $\beta > 0$, we have,

$$\mathbb{P} \left\{ |q_j(h_{t,l-1}) - q_j(\nabla f_t)| \leq \text{maxerr}_{t,l} - \frac{4k}{\varepsilon} \ln \frac{|Q|}{\beta} \right\} \leq \beta. \quad (12)$$

Additive error: Let us now analyze the error due to the Laplace Mechanism at Step 13. Let $\text{adderr}_{t,l}$ denote the additive error when measuring the query $q_{e_{t,l}}$, that is,

$$\text{adderr}_{t,l} = |m_{t,l} - q_{e_{t,l}}(\nabla f_t)|. \quad (13)$$

Again using j as shorthand for $e_{t,l}$. By concentration of the Laplace random variable we have, that for a noise of scale $\frac{2k}{\varepsilon}$,

$$\mathbb{P} \left\{ |m_{t,l} - q_{e_{t,l}}(\nabla f_t)| > \frac{2k}{\varepsilon} \log \frac{1}{\beta} \right\} = \beta. \quad (14)$$

Relative entropy: Similar to [11] we rely on relative entropy to show improvement in each iteration by using the multiplicative weights algorithm. Let the relative entropy at the end of iteration l at time t be given as

$$\Psi_{t,l} = \frac{1}{|\nabla f_t|} \sum_{x \in \mathcal{X}} \nabla f_t(x) \ln \left(\frac{\nabla f_t(x)}{h_{t,l}(x)} \right). \quad (15)$$

Then we have the following relations,

$$\Psi_{t,l} \geq 0, \quad (16)$$

$$\Psi_{0,0} \leq \ln |\mathcal{X}|, \quad (17)$$

$$\Psi_{t,l-1} - \Psi_{t,l} \geq \left(\frac{q_{e_{t,l}}(h_{t,l-1}) - q_{e_{t,l}}(\nabla f_t)}{2|\nabla f_t|} \right)^2 - \left(\frac{m_{t,l} - q_{e_{t,l}}(\nabla f_t)}{2|\nabla f_t|} \right)^2. \quad (18)$$

Equation (18) can be derived as follows,

$$\Psi_{t,l-1} - \Psi_{t,l} = \frac{1}{|\nabla f_t|} \sum_{x \in \mathcal{X}} \nabla f_t(x) \ln \left(\frac{h_{t,l-1}(x)}{h_{t,l}(x)} \right) = \frac{1}{|\nabla f_t|} \sum_{x \in \mathcal{X}} \nabla f_t(x) \ln \left(\frac{h_{t,l-1}(x) \cdot \exp \left(q_{e_{t,l}}(x) \cdot \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|} \right) \right)}{h_{t,l-1}(x) Z_{t,l}} \right),$$

where $Z_{t,l} = \frac{1}{|\nabla f_t|} \sum_{x \in \mathcal{X}} h_{t,l-1}(x) \exp \left(q_{e_{t,l}}(x) \cdot \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|} \right) \right)$ is the normalization constant. So,

$$\Psi_{t,l-1} - \Psi_{t,l} = \frac{1}{|\nabla f_t|} \sum_{x \in \mathcal{X}} \nabla f_t(x) \left(q_{e_{t,l}}(x) \cdot \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|} \right) - \ln Z_{t,l} \right) = \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|^2} \right) q_{e_{t,l}}(\nabla f_t) - \ln Z_{t,l}.$$

Using $e^x \leq 1 + x + x^2$ for all $|x| \leq 1$ and $\left| q_{e_{t,l}}(x) \cdot \frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|} \right| \leq 1$, we have,

$$\begin{aligned}
Z_{t,l} &= \frac{1}{|\nabla f_t|} \sum_{x \in \mathcal{X}} h_{t,l-1}(x) \exp \left(q_{e_{t,l}}(x) \cdot \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|} \right) \right) \\
&\leq \frac{1}{|\nabla f_t|} \sum_{x \in \mathcal{X}} h_{t,l-1}(x) \left(1 + q_{e_{t,l}}(x) \cdot \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|} \right) + \left(q_{e_{t,l}}(x) \cdot \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|} \right) \right)^2 \right) \\
&\leq 1 + \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|} \right)^2 + q_{e_{t,l}}(h_{t,l-1}) \cdot \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|^2} \right). \\
\Rightarrow \ln Z_{t,l} &\leq \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|} \right)^2 + q_{e_{t,l}}(h_{t,l-1}) \cdot \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|^2} \right).
\end{aligned}$$

Using this in the entropy difference bound we have,

$$\begin{aligned}
\Psi_{t,l-1} - \Psi_{t,l} &\geq \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|^2} \right) \left(q_{e_{t,l}}(\nabla f_t) - q_{e_{t,l}}(h_{t,l-1}) \right) - \left(\frac{m_{t,l} - q_{e_{t,l}}(h_{t,l-1})}{2|\nabla f_t|} \right)^2 \\
&= \left(\frac{q_{e_{t,l}}(h_{t,l-1}) - q_{e_{t,l}}(\nabla f_t)}{2|\nabla f_t|} \right)^2 - \left(\frac{m_{t,l} - q_{e_{t,l}}(\nabla f_t)}{2|\nabla f_t|} \right)^2.
\end{aligned} \tag{19}$$

Finally: Suppose we are interested in error at time T . Let $\beta > 0$ be the failure probability at some time $t \in [T]$. Then, using Equations (12), (14) and (18) and a union bound over $l \in [k]$, with probability at least $1 - \beta$, for all $l \in [k]$ simultaneously,

$$\text{maxerr}_{t,l} \leq \left| q_{e_{t,l}}(h_{t,l-1}) - q_{e_{t,l}}(\nabla f_t) \right| + \frac{4k}{\varepsilon} \log \left(\frac{2k|Q|}{\beta} \right), \tag{20}$$

and,

$$\text{adderr}_{t,l} = \left| m_{t,l} - q_{e_{t,l}}(\nabla f_t) \right| \leq \frac{2k}{\varepsilon} \log \left(\frac{2k}{\beta} \right). \tag{21}$$

Combining the above two equations with Equation 18 we have, that with probability at least $1 - \beta$,

$$\text{maxerr}_{t,l} \leq \left(4|\nabla f_t|^2 \left(\Psi_{t,l-1} - \Psi_{t,l} \right) + \text{adderr}_{t,l}^2 \right)^{1/2} + \frac{4k}{\varepsilon} \log \left(\frac{2k|Q|}{\beta} \right).$$

Finally, we can bound the maximum error in approximating the differential dataset as,

$$\begin{aligned}
\max_{q \in Q} |q(\nabla g_t) - q(\nabla f_t)| &= \max_{q \in Q} \left| q \left(\text{avg}_{l \in [k]} h_{t,l} \right) - q(\nabla f_t) \right| \\
&\leq \text{avg}_{l \in [k]} \max_{q \in Q} |q(h_{t,l}) - q(\nabla f_t)| = \text{avg}_{l \in [k]} \text{maxerr}_{t,l} \\
&\leq \text{avg}_{l \in [k]} \left(4|\nabla f_t|^2 \left(\Psi_{t,l-1} - \Psi_{t,l} \right) + \text{adderr}_{t,l}^2 \right)^{1/2} + \frac{4k}{\varepsilon} \log \left(\frac{2k|Q|}{\beta} \right) \\
&= \left(\frac{4|\nabla f_t|^2}{k} \left(\Psi_{t,0} - \Psi_{t,k} \right) + \text{adderr}_{t,l}^2 \right)^{1/2} + \frac{4k}{\varepsilon} \log \left(\frac{2k|Q|}{\beta} \right) \\
&\leq \left(\frac{4|\nabla f_t|^2}{k} \ln |\mathcal{X}| + \text{adderr}_{t,l}^2 \right)^{1/2} + \frac{4k}{\varepsilon} \log \left(\frac{2k|Q|}{\beta} \right) \\
&\leq 2|\nabla f_t| \sqrt{\frac{\ln |\mathcal{X}|}{k}} + \frac{2k}{\varepsilon} \log \left(\frac{2k}{\beta} \right) + \frac{4k}{\varepsilon} \log \left(\frac{2k|Q|}{\beta} \right).
\end{aligned}$$

Let us suppose we are interested in error at time $T \in \mathbb{N}$. Taking a union bound over time we have, that with probability at least $1 - \beta$, for all $t \leq T$ simultaneously,

$$\begin{aligned}
\max_{q \in Q} |q(g_t) - q(f_t)| &\leq \sum_{t=1}^T \max_{q \in Q} |q(\nabla g_t) - q(\nabla f_t)| \\
&\leq \sum_{t=1}^T \left(2|\nabla f_t| \sqrt{\frac{\ln |X|}{k}} + \frac{2k}{\varepsilon} \log \left(\frac{2kt}{\beta} \right) + \frac{4k}{\varepsilon} \log \left(\frac{2kt|Q|}{\beta} \right) \right) \\
&\leq 2|f_T| \sqrt{\frac{\ln |X|}{k}} + \frac{2k}{\varepsilon} \sum_{t=1}^T \left(\log \left(\frac{2t|Q|}{\beta} \right) + 2 \log \left(\frac{2t|Q|^2}{\beta} \right) \right) \\
&\leq 2|f_T| \sqrt{\frac{\ln |X|}{k}} + \frac{6k}{\varepsilon} \sum_{t=1}^T \left(\log \left(\frac{2t|Q|^{5/3}}{\beta} \right) \right) \\
&\leq 2|f_T| \sqrt{\frac{\ln |X|}{k}} + \frac{6kT}{\varepsilon} \log \left(\frac{2T|Q|^{5/3}}{\beta} \right).
\end{aligned}$$

Let us compare the upper bound to a function of the form $u(k) = \frac{a}{\sqrt{k}} + bk$, then we can optimize for the value of k with $k_* = \left(\frac{a}{2b}\right)^{2/3}$. This results in $u(k_*) = \left(2^{1/3} + 2^{-1/3}\right) a^{2/3} b^{1/3}$. Using this optimal value in our upper bound so far, we have,

$$\max_{q \in Q} |q(g_t) - q(f_t)| \leq O \left(\left(|f_T| \sqrt{\ln |X|} \right)^{2/3} \left(\frac{T}{\varepsilon} \log \left(\frac{T|Q|^{5/3}}{\beta} \right) \right)^{1/3} \right) \leq O \left(|f_T|^{2/3} \left(\frac{\ln |X| \ln |Q| (T \log T)}{\varepsilon \beta} \right)^{1/3} \right).$$

□

B ACCURACY ANALYSIS FOR THE PROPOSED METHOD

In this section, we try to find a bound on the accuracy of Algorithm 3. The analysis mostly follows what we did in Section A and we use the notations $\text{maxerr}_{t,l}$, $\text{adderr}_{t,l}$ and $\Psi_{t,l}$ from that proof. Additionally, we use the notation maxerr_t to denote the maximum error comparing the input and synthetic stream snapshot at time t , that is

$$\text{maxerr}_t := \max_{q \in Q} |q(g_t - f_t)|, \tag{22}$$

note that there is only one index in the subscript here, unlike $\text{maxerr}_{t,l}$.

Furthermore, we use the accuracy guarantees of the binary tree mechanism from [4] as stated in Lemma 4.

LEMMA B.1. *[Accuracy of unbounded binary tree counter] For any $t \in \mathbb{N}$ and $\beta > 0$, an ε -differentially private unbounded binary tree counter is $\left(O \left(\frac{1}{\varepsilon} (\log t)^{1.5} \log \frac{1}{\beta} \right), \beta \right)$ -accurate.*

Selection error: In Algorithm 3, we use an approximation of the true data in the exponential mechanism, such that at any time $t \in \mathbb{N}$, we use $\nabla f_t + g_{t-1}$ instead of f_t for true data. This introduces bias which can be analyzed as,

$$\begin{aligned}
\max_{q \in Q} |q(h_{t,l-1}) - q(\nabla f_t - g_{t-1})| &= \max_{q \in Q} |q(h_{t,l-1} - f_t) - q(g_{t-1} - f_{t-1})| \\
&\geq \max_{q \in Q} |q(h_{t,l-1} - f_t)| - |q(g_{t-1} - f_{t-1})| \\
&\geq \max_{q \in Q} |q(h_{t,l-1} - f_t)| - \max_{q \in Q} |q(g_{t-1} - f_{t-1})| \\
&= \text{maxerr}_{t,l} - \text{maxerr}_{t-1}.
\end{aligned}$$

At the l^{th} iteration at time t , we select the query with index i , where i is a shorthand for $e_{t,l}$. By the utility of the Exponential mechanism (Theorem 3.10) invoked with a privacy budget $\varepsilon/2k$ and sensitivity 1, we have,

$$\begin{aligned}
& \mathbb{P} \left\{ \left| q_i(h_{t,l-1}) - q_i(\nabla f_t + g_{t-1}) \right| \leq \max_{q \in Q} |q(h_{t,l-1}) - q(\nabla f_t + g_{t-1})| - \frac{4k}{\varepsilon} \ln \frac{|Q|}{\beta} \right\} \leq \beta \\
& \mathbb{P} \left\{ \left| q_i(h_{t,l-1}) - q_i(\nabla f_t + g_{t-1}) \right| \leq \text{maxerr}_{t,l} - \text{maxerr}_{t-1} - \frac{4k}{\varepsilon} \ln \frac{|Q|}{\beta} \right\} \leq \beta \\
& \mathbb{P} \left\{ \left| q_i(h_{t,l-1} - f_t) + q_i(f_{t-1} - g_{t-1}) \right| \leq \text{maxerr}_{t,l} - \text{maxerr}_{t-1} - \frac{4k}{\varepsilon} \ln \frac{|Q|}{\beta} \right\} \leq \beta \\
& \mathbb{P} \left\{ \left| q_i(h_{t,l-1} - f_t) \right| + \max_{q \in Q} |q(f_{t-1} - g_{t-1})| \leq \text{maxerr}_{t,l} - \text{maxerr}_{t-1} - \frac{4k}{\varepsilon} \ln \frac{|Q|}{\beta} \right\} \leq \beta \\
& \mathbb{P} \left\{ \left| q_i(h_{t,l-1} - f_t) \right| + \text{maxerr}_{t-1} \leq \text{maxerr}_{t,l} - \text{maxerr}_{t-1} - \frac{4k}{\varepsilon} \ln \frac{|Q|}{\beta} \right\} \leq \beta,
\end{aligned}$$

which results in,

$$\mathbb{P} \left\{ \left| q_i(h_{t,l-1} - f_t) \right| \leq \text{maxerr}_{t,l} - 2 \text{maxerr}_{t-1} - \frac{4k}{\varepsilon} \ln \frac{|Q|}{\beta} \right\} \leq \beta. \quad (23)$$

Additive error: Using $m_{t,l} = C_i(t) + r_i(t)$, additive error becomes,

$$\text{adderr}_{t,l} = |C_i(t) + r_i(t) - q_i(f_t)|. \quad (24)$$

Let $N_i(t) \subseteq N$ be the times when query index i is selected by the exponential mechanism at or before time t . Let $\tilde{N}_i(t) = [t] \setminus N_i(t)$. Then,

$$\begin{aligned}
\text{adderr}_{t,l} & \leq \left| C_i(t) - q_i(f_{N_i(t)}) \right| + \left| r_i(t) - q_i(f_{\tilde{N}_i(t)}) \right| \\
& \leq \left| C_i(t) - q_i(f_{N_i(t)}) \right| + \left| q_i(g_{t-1}) - C_i(t-1) - q_i(f_{\tilde{N}_i(t)}) \right| \\
& \leq \left| C_i(t) - q_i(f_{N_i(t)}) \right| + \left| q_i(g_{t-1}) - C_i(t-1) - q_i(f_{\tilde{N}_i(t)}) \right| \\
& \leq \left| C_i(t) - q_i(f_{N_i(t)}) \right| + |q_i(g_{t-1}) - q_i(f_{t-1})| + \left| C_i(t-1) - q_i(f_{N_i(t-1)}) \right|
\end{aligned}$$

By Lemma B.1, we have,

$$\mathbb{P} \left\{ \left| C_i(t) - q_i(f_{N_i(t)}) \right| \geq \frac{c}{\varepsilon} \ln \left(\frac{1}{\beta} \right) \left(\ln |N_i(t)| \right)^{3/2} \right\} \leq \beta, \quad (25)$$

for some constant c .

Overall: Then, using a union bound and Equations (23) and (25), with probability at least $1 - \beta$, for all $l \in [k]$ and $t \in [T]$ simultaneously,

$$\text{maxerr}_{t,l} \leq |q_i(h_{t,l-1} - f_t)| + 2 \text{maxerr}_{t-1} + \frac{4k}{\varepsilon} \ln \left(\frac{2kT|Q|}{\beta} \right), \quad (26)$$

and

$$\text{adderr}_{t,l} \leq \frac{c}{\varepsilon} \ln \left(\frac{2kT}{\beta} \right) \left(\left(\ln |N_i(t)| \right)^{3/2} + \left(\ln |N_i(t-1)| \right)^{3/2} \right) + \text{maxerr}_{t-1}. \quad (27)$$

Relative entropy Recall that Equation (18) from the proof of Theorem 5.1 states,

$$\Psi_{t,l-1} - \Psi_{t,l} \geq \left(\frac{q_{e_{t,l}}(h_{t,l}) - q_{e_{t,l}}(f_t)}{2|f_t|} \right)^2 - \left(\frac{m_{t,l} - q_{e_{t,l}}(f_t)}{2|f_t|} \right)^2.$$

Combining the equations we have,

$$\text{maxerr}_{t,l} \leq \left(4|f_t|^2 \left(\Psi_{t,l-1} - \Psi_{t,l} \right) + \text{adderr}_{t,l}^2 \right)^{1/2} + 2 \text{maxerr}_{t-1} + \frac{4k}{\varepsilon} \ln \left(\frac{2kT|Q|}{\beta} \right).$$

Then, similar to the proof of Theorem 5.1, for any $t \in [T]$, we have,

$$\max_{q \in Q} |q(g_t) - q(f_t)| \leq 2|f_t| \sqrt{\frac{\ln |X|}{k}} + \text{adderr}_{t,l} + 2 \text{maxerr}_{t-1} + \frac{4k}{\varepsilon} \ln \left(\frac{2kT|Q|}{\beta} \right).$$

This results in the following recursive relation

$$\max_{q \in Q} |q(g_t) - q(f_t)| \leq 2|f_t| \sqrt{\frac{\ln |X|}{k}} + \frac{2c}{\varepsilon} \ln \left(\frac{2kT}{\beta} \right) (\ln t)^{3/2} + \frac{4k}{\varepsilon} \ln \left(\frac{2kT|Q|}{\beta} \right) + 3 \text{maxerr}_{t-1}. \quad (28)$$

We can simplify the above recursive relation such that at time T we have,

$$\max_{q \in Q} |q(g_T) - q(f_T)| \leq \sum_{t=1}^T 3^{T-t} \left(2|f_t| \sqrt{\frac{\ln |\mathcal{X}|}{k}} + \frac{2c}{\varepsilon} \ln \left(\frac{2kT}{\beta} \right) (\ln t)^{3/2} + \frac{4k}{\varepsilon} \ln \left(\frac{2kT|Q|}{\beta} \right) \right).$$

Note that the above bound has a term exponential in time and thus is not better than what we had for the accuracy of StreamingMWEM in Theorem 5.1. However, the results of our empirical experiments (Section 7.4) suggest that the proposed method outperforms the StreamingMWEM. The reason that we do not have a better bound in the theoretical proof is that our algorithm depends on the output of the previous time step, which results in the worst-case recursive relation as mentioned in Equation (28). Moreover, the theorem and proof do not exploit potential cancellations in the added noise and we conjecture that being able to utilize cancellations should give an improved bound.