Open access • Book Chapter • DOI:10.1007/978-3-642-15546-8_11

Differentially private data release through multidimensional partitioning

Yonghui Xiao, Li Xiong, Chun Yuan

Institutions: Emory University, Tsinghua University

Published on: 17 Sep 2010 - Very Large Data Bases

Topics: Differential privacy

Differentially Private Data Release through Multidimensional Partitioning

Yonghui Xiao[1,2], Li Xiong[1], and Chun Yuan[2]

[1] Emory University, Atlanta GA 30322, USA
[2] Tsinghua University Graduate School at Shenzhen, Shenzhen 518055, China

Abstract. Differential privacy is a strong notion for protecting individual privacy in privacy preserving data analysis or publishing. In this paper, we study the problem of differentially private histogram release based on an interactive differential privacy interface. We propose two multidimensional partitioning strategies including a baseline cell-based partitioning and an innovative kd-tree based partitioning. In addition to providing formal proofs for differential privacy and usefulness guarantees for linear distributive queries , we also present a set of experimental results and demonstrate the feasibility and performance of our method.

1 Introduction

As information technology enables the collection, storage, and usage of massive amounts and types of information about individuals and organizations, privacy becomes an increasingly important issue. Governments and organizations recognize the critical value in sharing such information while preserving the privacy of individuals. Privacy preserving data analysis and data publishing [5, 10, 3] has received considerable attention in recent years as a promising approach for sharing information while preserving data privacy. There are two models for privacy protection [5]: the interactive model and the non-interactive model. In the interactive model, a trusted *curator* (e.g. hospital) collects data from *record owners* (e.g. patients) and provides an access mechanism for *data users* (e.g. public health researchers) for querying or analysis purposes. The result returned from the access mechanism is perturbed by the mechanism to protect privacy. In the non-interactive model, the curator publishes a "sanitized" version of the data, simultaneously providing utility for data users and privacy protection for the individuals represented in the data.

Differential privacy [6, 4, 5, 3] is widely accepted as one of the strongest known unconditional privacy guarantees with the advantage that it makes no assumption on the attacker's background knowledge. It requires the outcome of computations to be formally indistinguishable when run with and without any particular record in the dataset, as if it makes little difference whether an individual is being opted in or out of the database. Many meaningful results have been obtained for the interactive model with differential privacy [6, 4, 5, 3]. Non-interactive data release with differential privacy has been recently studied with

hardness results obtained and it remains an open problem to find efficient algorithms for many domains [7].

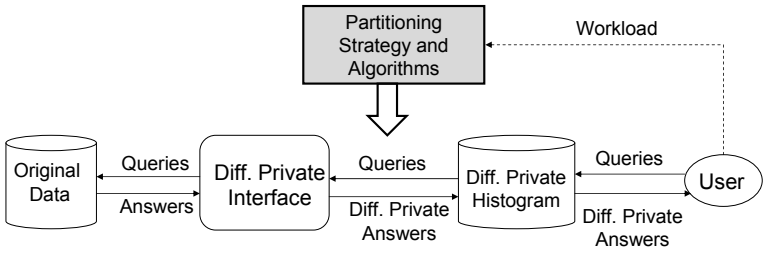

Fig. 1. Differentially Private Histogram Release

In this paper, we study the problem of differentially private histogram release based on an interactive differential privacy interface, as shown in Figure 1. A *histogram* is a disjoint partitioning of the database points with the number of points which fall into each partition. An interactive differential privacy interface, such as the Privacy INtegrated Queries platform (PINQ) [19], provides a differentially private access to the raw database. An algorithm implementing the partitioning strategy submits a sequence of queries to the interface and generates a differentially private histogram of the raw database. The histogram can then serve as a sanitized synopsis of the raw database and, together with an optional synthesized dataset based on the histogram, can be used to support count queries and other types of OLAP queries and learning tasks.

An immediate question one might wonder is what is the advantage of the non-interactive release compared to using the interactive mechanism to answer the queries directly. A common mechanism providing differential private answers is to add carefully calibrated noise to each query determined by the privacy parameter and the sensitivity of the query. The composability of differential privacy [19] ensures privacy guarantees for a sequence of differentially-private computations with additive privacy depletions in the worst case. Given an overall privacy requirement or budget, expressed as a privacy parameter, it can be allocated to subroutines or each query in the query sequence to ensure the overall privacy. When the number of queries grow, each query gets a lower privacy budget which requires a larger noise to be added. When there are multiple users, they have to share a common privacy budget which degrades the utility rapidly. The non-interactive approach essentially exploits the data distribution and the query workload and uses a carefully designed algorithm or query strategy such that the overall noise is minimized for a particular class of queries. As a result, the partitioning strategy and the algorithm implementing the strategy for generat-

ing the query sequence to the interface are crucial to the utility of the resulting histogram or synthetic dataset.

Contributions. We study two partitioning strategies for the differentially private histogram release for random query workload and evaluate their utility. We summarize our contributions below.

- We study two partitioning strategies for the differentially private histogram release problem: 1) a baseline strategy using the most fine-grained cell partitioning, and 2) a kd-tree based partitioning strategy. There are several innovative features in our kd-tree based strategy. First, we incorporate a uniformity measure in the partitioning process which seeks to produce partitions that are close to uniform so that approximation errors within partitions are minimized. Second, we implement the strategy using a two-step algorithm that generates the kd-tree partitions based on the histogram generated from a cell partitioning so that the access to the differentially private interface is minimized.
- We present formal results and discuss the applications of the histogram for general online analytical processing (OLAP) and learning, and present a set of experimental evaluations and show the actual performance of our algorithm. We show that the cell partitioning strategy, while simple, provides formal bounded utility for linear distributive queries including count and sum. The kd-tree based partitioning, on the other hand, achieves better results empirically.

2 Related works

Privacy preserving data analysis and publishing has received considerable attention in recent years. We refer readers to [5, 10, 3] for several up-to-date surveys. We briefly review here the most relevant work to our paper and discuss how our work differs from existing work.

There has been a series of studies on interactive privacy preserving data analysis based on the notion of differential privacy [6, 4, 5, 3]. A primary approach proposed for achieving differential privacy is to add Laplace noise [6, 5, 4] to the original results. McSherry and Talwar[21] give an alternative method to implement differential privacy based on the probability of a returned result, called the exponential mechanism. Roth and Roughgarden [23] proposes a median mechanism which improves upon the Laplace mechanism. McSherry implemented the interactive data access mechanism into PINQ[19], a platform providing a programming interface through a SQL-like language.

There are recently a few works that studied general non-interactive data release with differential privacy [6, 2, 8, 24]. Blum et al. [2] proved the possibility of non-interactive data release satisfying differential privacy for queries with polynomial VC-dimension, such as predicate queries. It also proposed an inefficient algorithm based on the exponential mechanism. The result largely remains theoretical and the general algorithm is inefficient for the complexity and required

data size. Feldman et al. [8] proposed the notion "private coreset" to release data for certain queries: k-median, k-mean, k-center. X. Xiao et al. [24] developed a differentially private data release algorithm for predicate queries using wavelet transforms. In addition, several recent work studied differentially private mechanisms for particular kinds of data such as search logs [16, 11] or for specific applications such as recommender systems [20] or record linkage [14]. It is important to note that [14] uses several tree strategies including k-d tree in its partitioning step and our results show that our our uniformity-driven k-d tree strategy achieves better utility for random count queries. Another closely related work is [13] which generates differentially private histograms for single dimensional range queries through a consistency check technique. Several works [12, 17] studies mechanisms for a given query workload. [12] proposes an enhanced Laplace mechanism by examining the geometry shape of a given set of linear queries. [17] proposes a query matrix mechanism that generates an optimal query strategy based on the query workload of linear count queries. It is worth noting that the cell-based partitioning in our approach is essentially the identity query matrix referred in [17]. On the other hand, our kd-tree based partitioning will generate a query matrix that is dependent on the approximate data distribution.

In summary, our work complements and advances the above works in that we focus on differentially private histogram release for random query workload using a multidimensional partitioning approach that is "data-aware". The method provides a formal utility guarantee for a set of queries and also supports applications for general OLAP and learning.

3 Preliminaries and definitions

Given an original database D, we use $\mathcal{A}(D)$ to denote an interactive mechanism to access the database D. For a query $Q(D)$ the interactive query mechanism returns a perturbed result of $\mathcal{A}_Q(D)$. In the non-interactive mechanism, our goal is to release a database $\hat{D}$ to answer user queries, which satisfies differential privacy. For simplicity, we assume the output range of queries is arbitrary. In this section, we formally introduce the definitions of differential privacy, (ϵ, δ)-usefulness, and the notion of a data cube to facilitate our discussions.

3.1 Differential Privacy

Definition 31 (α-Differential privacy[4]) *In the interactive model, an access mechanism $\mathcal{A}$ satisfies α-differential privacy if for any neighboring databases*[3] *D_1 and D_2, for any query function Q, $r \subseteq Range(Q)$, $\mathcal{A}_Q(D)$ is the mechanism to return an answer to query $Q(D)$,*

$$Pr[\mathcal{A}_Q(D_1) = r] \leq e^{\alpha} Pr[\mathcal{A}_Q(D_2) = r]$$

[3] We use the definition of neighboring databases consistent with [19] which treats the databases as multisets of records and requires their symmetric difference to be 1.

In the non-interactive model, a data release mechanism $\mathcal{A}$ satisfies α-differential privacy if for all neighboring database D_1 and D_2, and released output $\hat{D}$,

$$Pr[\mathcal{A}(D_1) = \hat{D}] \leq e^{\alpha} Pr[\mathcal{A}(D_2) = \hat{D}]$$

To achieve differential privacy, we use the Laplace mechanism [6] that adds random noise of Laplace distribution to the true answer of a query Q, $\mathcal{A}(x) = Q(x) + Y$, where Y is the Laplace noise. The magnitude of the noise depends on the privacy level and the query's sensitivity.

Definition 32 (Sensitivity) *For arbitrary neighboring databases D_1 and D_2, the sensitivity of a query Q is the maximum difference between the query results of D_1 and D_2,*

$$GS_Q = max|Q(D_1) - Q(D_2)|$$

To achieve α-differential privacy for a given query Q on dataset D, it is sufficient to return $Q(D) + Y$ in place of the original result $Q(D)$ where we draw Y from $Lap(GS_Q/\alpha)$ [6].

3.2 Composition

The composability of differential privacy [19] ensures privacy guarantees for a sequence of differentially-private computations. For a general series of analysis, the privacy parameter values add up, i.e. the privacy guarantees degrade as we expose more information. In a special case that the analyses operate on disjoint subsets of the data, the ultimate privacy guarantee depends only on the worst of the guarantees of each analysis, not the sum.

Theorem 31 (Sequential Composition [19]) *Let M_i each provide α_i-differential privacy. The sequence of M_i provides $(\sum_i \alpha_i)$-differential privacy.*

Theorem 32 (Parallel Composition [19]) *If D_i are disjoint subsets of the original database and M_i provides α-differential privacy for each D_i, then the sequence of M_i provides α-differential privacy.*

3.3 Sufficient Bound of α

The level of differential privacy is determined by the parameter α. However, there is no specific guidelines on how to choose a proper α value. We attempt to analyze the risk of large α, and give a sufficient condition of α to guarantee privacy through the analysis of the prior and posterior probability of a value disclosure.

Theorem 33 *Assume an attacker has a priori belief for a target victim's value being d as P_0. After l queries that include the target victim using the Laplace noise based differential privacy mechanism, we have $(l\alpha)$-differentially privacy [19]. The posteriori belief after l queries, P_l, satisfies $P_l \leq P_0 * exp(l\alpha)$. (Proof in Appendix A)*

Corollary 31 (Sufficient bound of α) *$\alpha < ln(x)/l$ is a sufficient bound for guaranteeing $P_l/P_0 < x$. $\alpha < -ln(P_0)/l$ is a sufficient bound for guaranteeing $P_l < 1$.*

3.4 Data cube

We use a "data cube" to represent the data space $\mathcal{D}^n$. For example, if the database has N dimensions, it is an N-dimensional cube. We denote the data cube by Ω, a query $Q : \Omega \to \mathcal{R}$ maps the data in Ω to output range $\mathcal{R}$. All the records in the database are points in the data cube. We use the term "partition" to refer to any sub-cube in the data cube. We denote any sub-cube that that is not divided by any more dimensions by "cell", meaning it's the "smallest" sub-cube. We denote the number of cells by β.

3.5 (ϵ, δ)-usefulness

We represent the utility of the released data, by considering whether it is (ϵ, δ)-useful.

Definition 33 ((ϵ, δ)-usefulness[2]) *A database mechanism $\mathcal{A}$ is (ϵ, δ)-useful for queries in class C if with probability $1 - \delta$, for every $Q \in C$, and every database D, $\mathcal{A}(D) = \hat{D}$, $|Q(\hat{D}) - Q(D)| \leq \epsilon$.*

3.6 Categorization of Aggregate Queries

Definition 34 (Distributive query [18]) *A distributive aggregate query is a function that can be computed for a given data set by partitioning the data into small subsets, computing the function of each subset, and then merging the results in order to arrive at the function's value for the original (entire) data set.*

Definition 35 (Linear distributive query) *A linear distributive query is a function with result that can be computed as a linear function of the result from each subset.*

For example, sum() can be computed by first partitioning the data, then summing up the sums of each partition. avg() can not be distributively computed but it can be computed by a function of sum() and count().

In this paper, we mainly focus on linear count queries which will be used to generate the histogram and to form a query workload to evaluate the released histogram. We will discuss later how the released histogram can be used to support other types of OLAP queries such as sum and average.

Definition 36 (count query) *Absolute count query AC and relative count RC on a multi-dimensional database D is defined to be:*

$$AC_{P(x)}(D) = \sum_{x \in D} P(x) \qquad RC_{P(x)}(D) = \frac{\sum_{x \in D} P(x)}{n}$$

where $P(x)$ returns 1 or 0 depending on the predicate.

Note that $GS_{AC} = 1$, $GS_{RC} = \frac{1}{n}$ [4].

3.7 PINQ

PINQ [19] is a programming interface that provides a differentially private interface to a database. It provides operators for database aggregate queries such as count (**NoisyCount**) and sum (**NoisySum**) which uses Laplace noise and the exponential mechanism to enforce differential privacy. It also provides a **Partition** operator that can partition the dataset based on the provided set of candidate keys. The **Partition** operator takes advantage of parallel composition and thus the privacy costs do not add up.

4 Multidimensional Partitioning Approach

4.1 Overview

For differentially private histogram release, a multi-dimensional histogram on a set of attributes is constructed by partitioning the data points into mutually disjoint subsets called *buckets* or partitions. The counts or frequencies in each bucket is then released. Any access to the original database is conducted through the differential privacy interface to guarantee differential privacy. The histogram can be then used to answer random count queries and other types of queries.

The partitioning strategy will largely determine the utility of the released histogram to arbitrary count queries. Each partition introduces a bounded Laplace noise or *perturbation error* by the differential privacy interface. If a query predicate covers multiple partitions, the perturbation error is aggregated. If a query predicate falls within a partition, the result has to be estimated assuming certain distribution of the data points in the partition. The dominant approach in histogram literature is making the *uniform distribution assumption*, where the frequencies of records in the bucket are assumed to be the same and equal to the average of the actual frequencies[15]. This introduces an *approximation error*.

We illustrate the errors and the impact of different partitioning strategies through an example shown in Figure 2. Consider an original database that has 4 cells, each of which has 100 data points. In the first partitioning, the 4 cells are grouped into one partition and we release a noisy count for the partition. Alternatively, the 4 cells are separated into 4 partitions, each of which contain one cell, and we release a noisy count for each of the partitions or cells. Note that the noise are independently generated for each partition. Because the sensitivity of the count query is 1 and the partitioning only requires parallel composition of differential privacy, the magnitude of the noise in the two approaches are the same. Then if we have a query, count(A), to ask how many data points are in the region A, the best estimate for the first strategy based on the uniform distribution assumption will be an approximate answer, which is $100 + Y/4$. So the query error is $Y/4$. In this case, the approximation error is 0 because

[4] n should be the upper bound of data size, so it's constant.

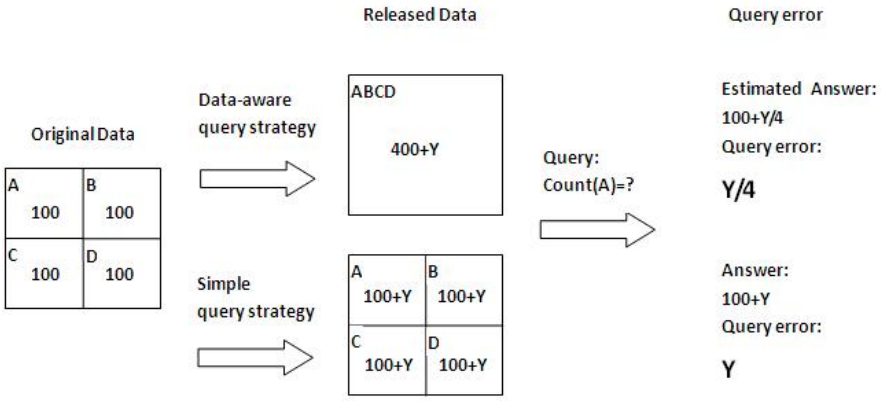

Fig. 2. Data-aware query strategy and simple query strategy

the cells in the partition are indeed uniform. If not, approximation error will be introduced. In addition, the perturbation error is also amortized among the cells. For the cell-based partitioning, the query error is Y which only consists of the perturbation error.

In general, a finer-grained partitioning will introduce smaller approximation errors but larger aggregated perturbation errors. Finding the right balance in this tradeoff to optimize the overall approximation of the data distribution and minimize the overall error for a random query workload is a key question. Not surprisingly, finding the optimal multi-dimensional histogram even without the privacy constraints is a challenging problem and optimal partitioning even in two dimensions is NP-hard[22]. Motivated by the above example and guided by the composition theorems, we summarize our two design goals: 1) generate uniform or close to uniform partitions so that the approximation error with the partitions is minimized, and 2) carefully and efficiently use the privacy budget to minimize the perturbation error. In this paper, we study two heuristic strategies: 1) the most fine-grained cell-based partitioning as a baseline strategy, which does not introduce approximation error but only perturbation error, and 2) a kd-tree based partitioning strategy that seeks to produce close to uniform partitions and an efficient implementation that seeks to minimize the perturbation error.

4.2 Cell-based algorithm

A simple strategy is to partition the data based on the domain and then release the count for each cell. The implementation is quite simple, taking advantage the **Partition** operator followed by **NoisyCount** on each partition.

Theorem 41 *Algorithm 1 achieves α-differential privacy.*

Proof. Because every cell is a disjoint subset of the original database, according to theorem 32, it's α-differentially private.

Algorithm 1 Cell-based algorithm

Require: α: differential privacy budget

1. **Partition** the data based on all domains.
2. release **NoisyCount** of each partition using privacy parameter α

Utility. We present a general theorem following a lemma that states a formal utility guarantee with cell-based partitioning for linear distributive queries.

Lemma 41 *If Y_i is the random variables i.i.d from Lap(b) with mean 0, then*

$$Pr[\sum_{i=1}^{\beta} |Y_i| \leq \epsilon] \geq 1 - \beta \cdot exp(-\frac{\epsilon}{\beta b})$$

Theorem 42 *The released $\hat{D}$ of algorithm 1 maintain (ϵ, δ)-useful for linear distributive query $Q(D) = \sum_{i=1}^{x} Q(D_i)$, if $GS \leq \frac{\alpha\epsilon}{\beta ln(\beta/\delta)}$, where x is the number of cells contained in the predicate, $x \leq \beta$, β is the number of partitioned cells of the data cube.*

Proof. By interactive mechanism of differential privacy, the returned answer $\mathcal{A}_Q(D) = Q(D) + Y$, where $Q(D)$ is the true answer of the query and Y is the Laplace noise Lap(b) where b=GS/α. we use D_i to present the data in the cells, then the returned answer of $\hat{D}$ is

$$Q(\hat{D}) \;=\; \sum_{i=1}^{x}(Q(D_i) \,+\, Y_i) \;=\; \sum_{i=1}^{x} Q(D_i) \,+\, \sum_{i=1}^{x} Y_i \;=\; Q(D) \,+\, \sum_{i=1}^{x} Y_i$$

With Lemma 41(proved in Appendix B), we have

$$Pr[|Q(x, D) - Q(x, \hat{D})| \leq \epsilon] \geq 1 - \beta \cdot exp(-\frac{\epsilon}{\beta b})$$

If $\beta \cdot exp(-\frac{\epsilon}{\beta b}) \leq \delta$, then we can get

$$Pr[|Q(x, D) - Q(x, \hat{D})| \leq \epsilon] \geq 1 - \delta$$

So, $\beta \cdot exp(-\frac{\epsilon}{\beta b}) \leq \delta$, b=GS/$\alpha$, we have

$$GS \leq \frac{\alpha\epsilon}{\beta ln(\beta/\delta)}$$

Corollary 41 *Algorithm 1 is (ϵ, δ)-useful for absolute count queries if $\epsilon \geq \beta ln(\beta/\delta)/\alpha$, for relative count queries if $n \geq \frac{\beta ln(\beta/\delta)}{\alpha\epsilon}$.*

4.3 K-d Tree based Algorithm

We now present our kd-tree based partitioning strategy. A kd-tree (k-dimensional tree) is a space-partitioning data structure for organizing data points in a k-dimensional space. A typical kd-tree construction starts from the root node which covers the entire space. At each step, a splitting dimension and a split value, typically median, from the range of the current partition on that dimension are chosen heuristically to divide the space into subspaces. The algorithm repeats until a pre-defined requirement (such as tree height or number of data points in each partition) are met. This method leads to a balanced kd-tree, in which each leaf node is about the same distance from the root, which is desired in indexing. In our setting, our main design goal is to generate uniform or close to uniform partitions so that the approximation error within the partitions is minimized. Thus we propose a uniformity based heuristic to make the decision whether to split the current partition. Concretely, we do not split a partition if it is close to uniform and split it otherwise. There are several metrics that can be used to measure the uniformity of a partition such as information entropy and variance. In our current implementation, we use a variance-like metric H defined below. If $H > \xi_1$ where ξ_1 is a threshold, then we split.

Definition 41 *Assuming we have an sub-cube D_0 with β cells, the average count would be $a_0 = \sum_{c_i \in D_0} count(c_i)/\beta$ where c_i is each cell in D_0. The heuristic metric is defined as:*

$$H(D_0) = \sum_{c_i \in D_0} |count(c_i) - a_0|$$

A straightforward implementation of the above kd-tree strategy is similar to that used in [14]. At each step, a **NoisyCount** is requested for each value of the splitting attribute in order to determine the split value median. The process repeats until the stop criteria is reached. As each step queries the original database, the composition theorem applies which results in cumulated privacy cost. Hence the privacy budget needs to be divided among all the steps and the final step to obtain the **NoisyCount** of each partition which is an inefficient use of the privacy budget. To this end, we propose a two-step algorithm that generates the kd-tree partitions based on the histogram generated from the cell partitioning. First, we generate a synthetic database D_c based on the cell-based algorithm. Then we perform kd-tree partitioning on D_c and use the resulting partitioning keys to **Partition** the original database. We finally release **NoisyCount** for each of the partitions. Essentially, the kd-tree is based on an approximate distribution of the original data. The original database is not queried during the kd-tree construction which saves the privacy budget. Our experiments verify that the benefit of having smaller perturbation error outweighs the aggregated approximation error. Algorithm 2 is the framework; Algorithm 3 is the step 2 of Algorithm 2.

[5] In this paper, we take the dimension which has the largest range in the current partition.

Algorithm 2 K-d tree based algorithm

Require: β: number of cells; α: the overall privacy budget

1. **Partition** the original database based on all domains.
2. get **NoisyCount** of each partition using privacy parameter $\alpha/2$ and generate a synthetic dataset D_c.
3. Partition D_c by algorithm 3.
4. **Partition** the original database based on the partition keys returned from step 3. 5. release **NoisyCount** of each partition using privacy parameter $\alpha/2$

Algorithm 3 K-d tree partitioning

Require: D_t: input database; ξ_0: threshold of generating k-d tree; ξ_1: threshold for function H;

1. Find a dimension of D_t[5];
2. Find the median m of the dimension;
3.

if $H(D_t) > \xi_1$ or $Count(D_t) > \xi_0$ **then**
  Divide D_t into D_{t1} and D_{t2} by m.
  partition D_{t_1} and D_{t_2} by algorithm 3.
end if
return partitions

Theorem 43 *Algorithm 2 is α-differentially private.*

Proof. Step 2 and Step 5 are $\alpha/2$-differentially private. So the sequence is α-differentially private because of theorem 31.

Utility. The utility of the algorithm is directly decided by the parameter ξ_1 and ξ_0 as well as the data distribution. We resort to experiments to evaluate the utility of the algorithm.

4.4 Applications

Having presented the multidimensional partitioning approach for differentially private histogram release, we now briefly discuss the applications that the released histogram can support.

OLAP On-line analytical processing (OLAP) is a key technology for business-intelligence applications. The computation of multidimensional aggregates, such as count(), sum(), max(), avg(), is the essence of on-line analytical processing. We discuss the applications of the above released data to common OLAP aggregate functions. We assume all queries have a predicate φ. An example form of the predicate can be φ_1 *and* $\varphi_2 \ldots$ *and* φ_m where φ_j can be a range predicate of the form $a_l \leq A_j \leq a_h$. The predicate determines a set of cells S_φ that satisfies the predicate. We denote the value of attribute A for cell i as a_i, and the perturbed count for cell i as c_i.

– Count queries are supported directly by the released data. The cell-based partitioning provides a guaranteed utility bound.

- Sum queries sum(A) for an attribute or dimension A can be computed as $\sum_{i \in S_\varphi}(a_i * c_i)$. Based on Theorem 42, the cell-based partitioning also provides a formal guarantee for sum query which is a linear distributive query.
- Average queries avg(A) for an attribute or dimension A can be computed as $\frac{\sum_{i \in S_\varphi}(A_i * c_i)}{\sum_{i \in S_\varphi}(c_i)}$. The algorithm, however, does not provide a formal utility guarantee for the average queries. We experimentally evaluate the utility for average queries in Section 5.

Learning. The released data can be also used for learning tasks such as construction of decision tree. Below we use a simple learning task of learning parity function to illustrate the idea. Given a non-zero vector $v \in \{0,1\}^{d-1}$, a data set D $\in \{0,1\}^d$, let k be the kth row of the data set and $x_k^{(j)}$ is the jth attribute of x_k. Without loss of generality, we define parity function as the inner product of v and the first $d-1$ attributes of x_i modulo 2 and the result is saved to the dth attribute of x_i, that is

$$g(k, v) = \oplus_{j \leq d-1} x_k^{(j)} v^{(j)} = x_k^{(d)},$$

Without loss of generality, we assume the parity query has the form:

$$PQ_v = Pr[g(v) = 1] = \frac{|\{i \in [n] : g(k, v) = 1\}|}{|D|}$$

So if without noise, the more $Pr[g(v)]$ is near 1, the more likely hypothesis v is correct.

Our problem is to learn the parity function based on the perturbed counts with noise. [1, 9] proposed a learning algorithm for parity function in the presence of random classification noise and adversarial noise. We introduce the algorithm in Appendix C.

5 Experiment

We use the CENSUS data(http://www.ipums.org) for our experiments. It has 1 million tuples and 4 attributes: Age, Education, Occupation and Income, whose domain sizes are 79, 14, 23 and 100 respectively. All experiments were run on a computer with Intel P8600($2 * 2.4$ GHz) CPU and 2GB memory.

5.1 Cell-based algorithm

Absolute count queries We use the Income attribute to simulate 1D count queries. The following parameters are used for the desired level of differential privacy and utility guarantee: $\beta = 100; \alpha = 0.05; \delta = 0.05$. The run time for this experiment is 0.155 seconds.

Figure 3, 4 show the original and released distribution. The x-axis is the Income, the y-axis is the corresponding count of each income value over the

complete domain. Figure 5 is the difference(Laplace noise) between the original and released distribution. We performed 100 independent releases and tested 1000 randomly generated queries for each data release and averaged the results. The average error is 62.2. We counted the times each error happens in Figure 6. The x-axis is the query error ϵ, and y-axis is the probability of each error. Figure 7 shows the actual error that is far less than the theoretical error.

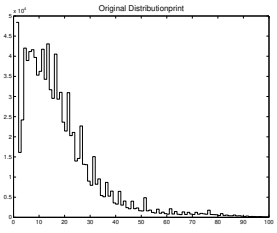

Fig. 3. Original Distribution of Income

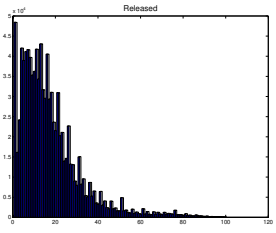

Fig. 4. Released Distribution of Income

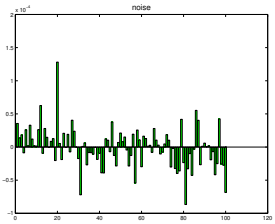

Fig. 5. Perturbation Noise

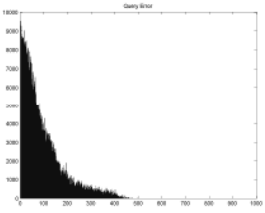

Fig. 6. Query Error of 1D Count Queries

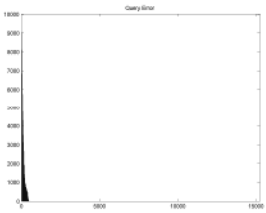

Fig. 7. Actual Error and Bound

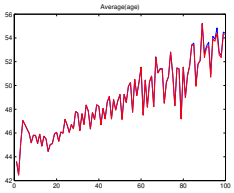

Fig. 8. Average(age) by Income

5.2 Average query

We use average query to demonstrate our method could be used for other queries which are not distributive queries. We extend the 1D count to 2D count using Age and Income as dimensions. Then we compare the difference of average(age) between original data and released data by each Income. Figure 8 shows the comparison. The blue line is the result of original data, red line is the result of perturbed data. We can see that the difference of the two sets of value differs lightly.

5.3 K-d tree based algorithm

We use the Age and Income attribute to simulate 2D rectangle queries with data generated by algorithm 2. We also implemented an alternative kd-tree strategy similar to that used in [14], referred to as hierarchical kd-tree, for comparison.

Hierarchical kd-tree The height of hierarchical kd-tree is the key for the query error of random workload. Figure 9 and 10 show the query error of absolute count and relative count. We can see that the least noise appears when the height is around 13.

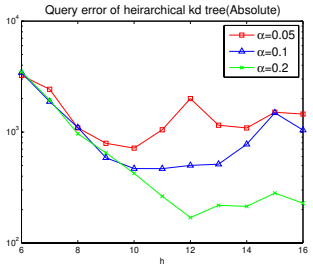

Fig. 9. Hierarchical kd-tree: Absolute count

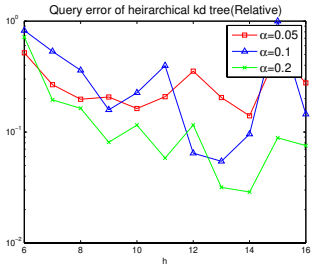

Fig. 10. Hierarchical kd-tree: Relative count

Query error vs different thresholds We analyze the different thresholds of ξ_0 and ξ_1 in our kd-tree algorithm. Figure 11 and 12 show the results. We can see that the threshold ξ_0 significantly affects the query error.

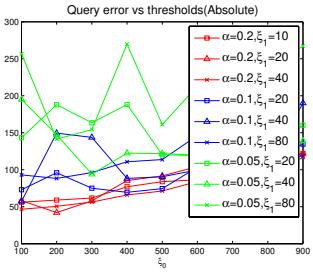

Fig. 11. Query error and thresholds: Absolute count

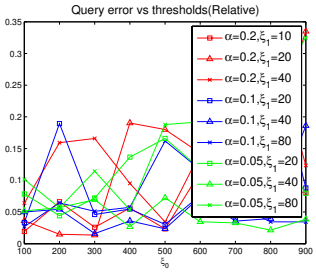

Fig. 12. Query error and thresholds: Relative count

Query error vs alpha We fix the height in hierarchical kd-tree algorithm and the threshold in our kd-tree algorithm to compare the query error. Figure 13 and 14 show the results. We can see that our kd tree algorithm provides best utility for random workload. Note that for relative count query, the utility outperforms other algorithm much more because for sparse data, the error of relative count query vary very much. Therefore, our kd-tree algorithm provides better results for sparse data.

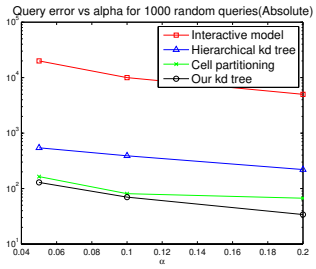

Fig. 13. Query error and α: Absolute count

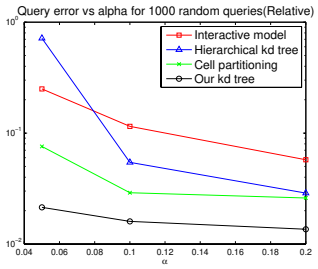

Fig. 14. Query error and α: Relative count

Query error vs Number of queries in interactive model When number of queries is small, the interactive model provides better result than our non-interactive approach. However, when the number of queries increases, the query error of random workload in interactive model may becomes larger. We can see form Figure 15 and 16 that when the number of queries are around 3 and 7, our kd-tree algorithm outperforms interactive model for absolute query and relative query.

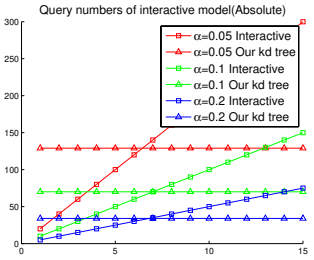

Fig. 15. Query error and number of queries: Absolute count

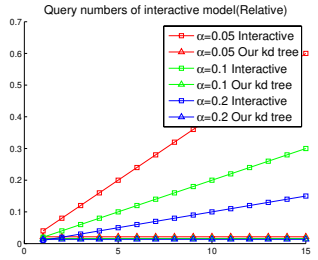

Fig. 16. Query error and number of queries: Relative count

6 Conclusions and Future works

We have described our preliminary studies on two multidimensional partitioning algorithms for differentially private histogram release based on an interactive differential privacy mechanism. By carefully designing the partitioning strategy and allocating the usage of the privacy budget, we show that our kd-tree based algorithm outperforms the baseline cell-based algorithm and an alternative kd-tree based algorithm for a random workload of counting queries. We also show that the result is significantly better than answering the queries directly through the interactive mechanism which is oblivious of the correlations of the queries and the data. We are interested in mapping the algorithms to the matrix mechanism framework [17] and conduct a formal error analysis. In addition, we are interested in exploring other spatial indexing techniques and integrating them with the consistency check techniques [13]. Finally, we plan to investigate in algorithms that are both data- and workload-aware to boost the accuracy for a specific workload.

Acknowledgements

The research is supported in part by an International Science Cooperation Fund of Shenzhen, China, GJ200807210023A and a Career Enhancement Fellowship by the Woodrow Wilson Foundation. The authors would also like to thank the chairs for their patience and support and the anonymous reviewers for their insightful comments that helped improve the paper.

References

1. A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the Acm*, 50(4):506–519, 2003.
2. A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. *Stoc'08: Proceedings of the 2008 Acm International Symposium on Theory of Computing*, pages 609–617, 2008. 14th Annual ACM International Symposium on Theory of Computing MAY 17-20, 2008 Victoria, CANADA.
3. C. Dwork. A firm foundation for private data analysis. (to appear).
4. C. Dwork. Differential privacy. *Automata, Languages and Programming, Pt 2*, 4052:1–12, 2006. Bugliesi, M Prennel, B Sassone, V Wegener, I 33rd International Colloquium on Automata, Languages and Programming JUL 10-14, 2006 Venice, ITALY.
5. C. Dwork. Differential privacy: A survey of results. In M. Agrawal, D.-Z. Du, Z. Duan, and A. Li, editors, *TAMC*, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2008.
6. C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. pages 265–284. Proceedings of the 3rd Theory of Cryptography Conference, 2006.
7. C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 381–390, New York, NY, USA, 2009. ACM.

8. D. Feldman, A. Fiat, H. Kaplan, and K. Nissim. Private coresets. *Stoc'09: Proceedings of the 2009 Acm Symposium on Theory of Computing*, pages 361–370, 2009. 41st Annual ACM Symposium on Theory of Computing MAY 31-JUN 02, 2009 Bethesda, MD.
9. V. Feldman, P. Gopalan, S. Khot, and A. K. Ponnuswami. New results for learning noisy parities and halfspaces. *47th Annual IEEE Symposium on Foundations of Computer Science, Proceedings*, pages 563–572, 2006. 47th Annual IEEE Symposium on Foundations of Computer Science OCT 21-24, 2006 Berkeley, CA.
10. B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey on recent developments. *ACM Computing Surveys*, 42(4), 2010.
11. M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Privacy in search logs. *CoRR*, abs/0904.0682, 2009.
12. M. Hardt and K. Talwar. On the geometry of differential privacy. In *STOC '10: Proceedings of the 42nd ACM symposium on Theory of computing*, pages 705–714, New York, NY, USA, 2010. ACM.
13. M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially-private queries through consistency. In *VLDB*, 2010.
14. A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino. Private record matching using differential privacy. In *13th International Conference on Extending Database Technology (EDBT)*, 2010.
15. Y. Ioannidis. The history of histograms (abridged). In *Proc. of VLDB Conference*, 2003.
16. A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *WWW*, 2009.
17. C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS '10: Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems of data*, pages 123–134, New York, NY, USA, 2010. ACM.
18. K. M and H. J, W. *Data mining: concepts and techniques, Second Edition*. MorganKaufman, 2006.
19. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, pages 19–30, New York, NY, USA, 2009. ACM.
20. F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 627–636, New York, NY, USA, 2009. ACM.
21. F. McSherry and K. Talwar. Mechanism design via differential privacy. *48th Annual Ieee Symposium on Foundations of Computer Science, Proceedings*, pages 94–103, 2007. 48th Annual IEEE Symposium on Foundations of Computer Science OCT 20-23, 2007 Providence, RI.
22. S. Muthukrishnan, V. Poosala, and T. Suel. On rectangular partitionings in two dimensions: Algorithms, complexity, and applications. In *ICDT*, pages 236–256, 1999.
23. A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *STOC*, pages 765–774, 2010.
24. X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *CoRR*, abs/0909.5530, 2009.

A Proof of Theorem 33

Proof.

$$Pr[D|A_1, A_2] = \frac{Pr[A_1, A_2|D]}{Pr[A_1, A_2]} = \frac{Pr[A_2|D] \cdot Pr[A_1|D] \cdot Pr[D]}{Pr[A_1, A_2]}$$
$$= \frac{Pr[A_2|D] \cdot Pr[D|A_1] \cdot Pr[A_1]}{Pr[A_2] \cdot Pr[A_1]} = \frac{Pr[A_2|D] \cdot Pr[D|A_1]}{Pr[A_2]}$$

Let's recap the model of differential privacy. Assume the attacker knows all the record except the ith record d_i. Let q_l be the lth query, $Q(q_l)$ be the true answer of q_l, a_l be the perturbed answer of q_l. Let x be the range of d, D be the event that $d_i = x_1$, P_l be the posterior probability of $Pr[D|a_1, a_2, \cdots a_l]$. Let $A_1 = \{a_1, a_2, \cdots, a_{l-1}\}$, $A_2 = \{a_l\}$, then

$$P_l = Pr[D|A_1, A_2] = P_{l-1} \cdot \frac{Pr[A_2|D]}{Pr[A_2]}$$
$$= P_{l-1} \cdot \frac{Pr[a_l|d_i = x_1]}{\sum_{x_j \in x} Pr[a_l|d_i = x_j] \cdot Pr[d_i = x_j]}$$

If we adopt Laplace noise to achieve differential privacy, then

$$\frac{p_l}{p_{l-1}} = \frac{\frac{1}{2b} exp(-|a_l - Q(q_l, x_1)|/b)}{\sum_{x_j \in x} \frac{1}{2b} exp(-|a_l - Q(q_l, x_j)|/b) \cdot Pr[d_i = x_j]}$$
$$\leq \frac{1}{\sum_{x_j \in x} exp(-|Q(q_l, x_j) - Q(q_l, x_1)|/b) \cdot Pr[d_i = x_j]}$$

Recall the definition of GS, then

$$|Q(q_l, x_j) - Q(q_l, x_1)| \leq GS$$

$$\frac{P_l}{P_{l-1}} \leq \frac{1}{\sum_{x_j \in x} exp(-GS/b) \cdot Pr[d_i = x_j]}$$

Because $b = GS/\alpha$, then $P_l/P_{l-1} \leq e^{\alpha}$. So, $Pr[V = d] \leq P_0 * exp(l\alpha)$.

B Proof of Lemma 41

Lemma B1 *If Y_i is the random variables i.i.d from Lap(b) with mean 0, then*

$$Pr[\sum_{i=1}^{\beta} |Y_i| \leq \epsilon] \geq 1 - \beta \cdot exp(-\frac{\epsilon}{\beta b})$$

Proof (of Lemma 41). We assume each $|Y_i| \leq \epsilon_1$ where $\epsilon_1 = \epsilon/\beta$. Otherwise we call $|Y_i| > \epsilon_1$ a FAILURE. If no FAILURE happens,we have

$$\sum_{i=1}^{\beta} |Y_i| \leq \beta \cdot \epsilon_1 = \epsilon$$

If a FAILURE happens, then we have $|Lap(b)| > \epsilon_1$, which means

$$\Pr[\text{a FAILURE}]=2\int_{\epsilon_1}^{\infty} \frac{1}{2b} exp(-\frac{x}{b})=e^{-\epsilon_1/b}$$

For each Y_i, Pr[no FAILURE happens]=1-Pr[FAILURE happens] and each Y_i is i.i.d distributed, we have

$$Pr[\sum_{i=1}^{\beta} |Y_i| \leq \epsilon] \geq (1 - e^{-\epsilon_1/b})^{\beta}$$

Let $F(x) = (1-x)^{\beta} + \beta x - 1$, then $F(0) = 0$. $F'(x) = -\beta(1-x)^{\beta-1} + \beta = \beta(1 - (1-x)^{\beta-1}) > 0$ when $0 < x < 1$. Note that $0 < e^{-\epsilon_1/b} < 1$, so $F(e^{-\epsilon_1/b}) > 0$. We get $(1 - e^{-\epsilon_1/b})^{\beta} > 1 - \beta \cdot e^{-\epsilon_1/b}$. Because $\epsilon_1 = \epsilon/\beta$,

$$Pr[\sum_{i=1}^{\beta} |Y_i| \leq \epsilon] \geq 1 - \beta \cdot exp(-\frac{\epsilon}{\beta b})$$

C Learning algorithm of parity

We assume the data is uniformly distributed and the noise added is classification noise. For the cell that satisfies the parity vector r, even if we add Laplace noise to the cell by the mechanism of differential privacy, it is not noise for the parity query. We use η_1 to present the original noise rate. Our algorithm add η_2 noise rate to the data. The total noise rate η in the released data is $\eta_1 + \eta_2$. Therefore, $\eta_2 \leq \epsilon$, $\eta \leq \eta_1 + \epsilon$.

Theorem C1 ([1]) *the length-d parity problem, for noise rate η equal to any constant less than 1/2, can be solved with number of samples and total computation-time $2^{O(d/logd)}$.*

Therefore, if $\eta < 1/2$, the $\hat{D}$ is learnable. $\eta \leq \eta_1 + \epsilon$ and $n \geq \frac{ln\frac{\beta}{\delta}}{\alpha\epsilon}\beta$, so if $n \geq \frac{ln\frac{\beta}{\delta}}{\alpha(1/2-\eta_1)}\beta$, the $\hat{D}$ is learnable.

Lemma C1 ([1]) *Take s samples from a database with parity vector v and noise rate η, then*

$$Pr[\oplus_{i=1}^{s} x_i^{(d)} = \oplus_{i=1}^{s} g(i,v)] = \frac{1}{2} + \frac{1}{2}(1-2\eta)^s.$$

note that each $x_i^{(d)}$ may not be the correct answer of g(i,v) because of noise.

We give the algorithm as follows:

Learning steps for parity vector:
For every target bit j of vector v, $j = 1 : d-1$
1. Draw s samples from the released data where $s > d-1$ to avoid linear dependence.
2. for $k = 1 : d-1$,

choose a record x_i where $x_i^{(k)} = 1$, remove x_i from the samples and XOR the rest samples with x_i, $x = x \oplus x_i$.

3. if in left samples doesn't exists $x^{(j)} = 1$, then goto 1; else after the elimination of step 2, we discard those samples which become 0, then randomly draw a sample t from the left samples.

$$Pr[v^{(j)} = 1] = Pr[t^{(j)} = 1] = \frac{1}{2} + \frac{1}{2}(1-2\eta)^{d-2}$$

with the lemma described above, we know that every $x^{(j)}$ has probability $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{d-2}$ to be correct. Therefore, with probability $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{d-2}$, we output the correct bit of vector v.

D Expected error of Laplace noise

Theorem D1 (Expected error of laplace noise) *The expected error of laplace noise with parameter α is GS/α.*

Proof.

$$err = |Q(\hat{D}) - Q(D)| = |Lap(GS/\alpha)| \tag{1}$$

$$b = GS/\alpha \tag{2}$$

$$E(err) = \int_{-\infty}^{+\infty} err * (\frac{1}{2b} exp(-\frac{|x|}{b})) \tag{3}$$

solve the equations, we get $E(err) = GS/\alpha$.