Technical Report

TR-2011-005

ISPE: Adaptive differentially private data release and query estimation

by

Yonghui Xiao, Li Xiong, Yuan Chun

Mathematics and Computer Science

EMORY UNIVERSITY

ISPE: Adaptive Differentially Private Data Release and Query Estimation

Yonghui Xiao*
Graduate School at Shenzhen
Tsinghua University
Shenzhen 518055,China
xiaoyh08@mails.tsinghua.edu.cn

Li Xiong
Emory University
Atlanta GA 30322, USA
lxiong@mathcs.emory.edu

Chun Yuan
Graduate School at Shenzhen
Tsinghua University
Shenzhen 518055,China
yuanc@sz.tsinghua.edu.cn

ABSTRACT

Although the mechanism of differential privacy provides a strong guarantee for privacy protection, it remains a key open problem to find efficient algorithms for non-interactive differentially private data release while maintaining good utility. In this paper, we propose an adaptive framework, called ISPE, to release differentially private histogram data through an interactive differentially private interface and estimate arbitrary count queries with high accuracy. The data release component adaptively releases a set of histograms that explicitly exploit the underlying data indirectly observed by the differentially private interface. The estimation component uses a novel notion of neighborhood reference and proposes an algorithm based on the minimum mean squared error (MMSE) estimator that seeks an estimate answer for a given query such that the error between the estimated answer and the true answer is minimized. We present a set of experimental results demonstrating the feasibility and high accuracy of the approach using both synthetic data and real world dataset.

1. INTRODUCTION

Privacy preserving data analysis and data publishing [5, 11, 6] has received considerable attention in recent years as a promising approach for sharing information while preserving data privacy. Differential privacy [5, 6] is widely accepted as one of the strongest known unconditional privacy guarantees. It requires that 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 [11, 6]. 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. Programming interfaces such as PINQ [23] are now available for providing an encapsulated differentially private interface to raw data that ensures differential privacy for all its outputs. Due to the composibility of differential privacy [23], given an overall privacy requirement or budget, expressed as a privacy parameter, it has to be allocated to subroutines or each query in the query sequence to ensure the overall privacy. After the budget is exhausted, either the database has to be shut down, or any further query would be rejected. This limitation has greatly hindered their applicability, especially in the scenario that multiple users need to pose a large number of queries for exploratory analysis. Blum et al. [3] showed the possibility of non-interactive data release, publishing a "sanitized" version of the data, with differential privacy at the cost of preserving usefulness for only restricted classes of queries. [8] further proposed more efficient algorithms with hardness results obtained and it remains a key open problem to find efficient algorithms for non-interactive data release with differential privacy for many domains. [15] pointed out that a natural approach to side-stepping the hardness is relaxing the utility requirement, and not requiring accuracy for *every* input database. Sharing the insights from [15], our primary viewpoint is that it is possible and desirable in practice to design efficient *data-driven* heuristic mechanisms for differentially private data release.

In this paper, we consider the problem of non-interactive differentially private date release for answering random predicate counting queries. A counting query retrieves the number of tuples with the given combination of attribute values specified by the query predicate. A few recent works [20, 16, 28] proposed promising query strategies for releasing data focused on linear counting queries. However, the query strategies are generated either from a known workload or as a static hierarchical or wavelet query sequence, *oblivious* of the data. We argue that more effective solutions could be achieved by exploiting the characteristics of the underlying dataset.

Contributions. We propose ISPE, a *data-driven* and *adaptive* differentially private data release mechanism through an

*Research conducted while the author is a visiting student at Emory university.

VLDB '10, September 13-17, 2010, Singapore

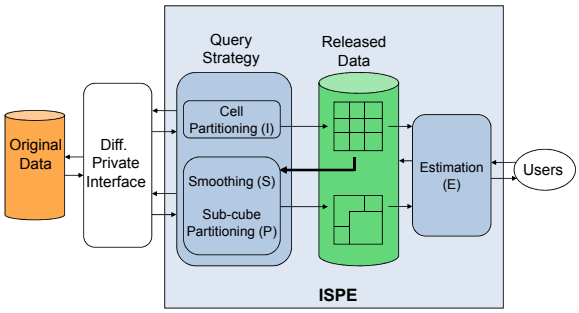


Figure 1: ISPE: Adaptive Differentially Private Data Release

interactive differentially private interface. **Figure 1** presents an overview of our approach. An interactive differential private interface, such as PINQ [23], is used to provide differentially private access to the original data. Our approach accesses the underlying database *indirectly* via this interface using a designed query strategy consisting of a sequence of queries and generates a differentially private view of the original data, modeled as multidimensional data cubes or histograms, on which the users could conduct unlimited number of queries and various data mining tasks. For a query issued by users, given potentially multiple ways of answering the query using the release data, the estimation component generates an answer using statistical inference techniques. We circumvent the hardness of differentially private data release in the non-interactive setting by novel and sophisticated use of the privacy preserving interface exploiting the characteristics of the underlying data. The key innovations of our approach are the following.

- It utilizes an adaptive query strategy for releasing data. The query strategy consists of the following steps: 1) domain-based cell partitioning or Identity matrix query strategy (I) for releasing a cell histogram, 2) smoothing (S) of the cell histogram, and 3) sub-cube partitioning (P) based on the cell histogram for releasing a sub-cube histogram. The key innovations are: 1) it uses a feedback loop between the released data and the query strategy to explicitly exploit the underlying data, i.e. the cell histogram indirectly observed by the differentially private interface in the I step adaptively determines the partitioning query strategy in the P step, 2) the smoothing and the sub-cube partitioning utilize techniques from image processing and group neighboring cells with similar density to enable the neighborhood reference in the estimation step to boost accuracy.

- Given an arbitrary user query, it uses statistical estimation techniques (E) to accurately answer the query using the released data. We formulate the query answering problem as a probabilistic inference problem and propose an algorithm based on the minimum mean squared error (MMSE) estimator. It has several unique and novel features: 1) Unlike previous approach [16], it minimizes the error between the estimated answer and the true answer, 2) it uses neighborhood reference, the neighboring cells in the data cube that have similar density, to help boosting accuracy when estimating a given cell, and 3) this estimation method can conduct not only point estimation but also interval estimation for arbitrary count queries.

2. PRELIMINARIES AND DEFINITIONS

2.1 Notation

We use bold characters to denote vector or matrix, normal character to denote one row of the vector or matrix, subscript i to denote the ith row of the vector or matrix. For a vector, the operator "[i]" is also used to denote the ith number. We will introduce our notations as we introduce the definitions. For reference purposes, Appendix A gives an overview of the key notations used in this paper.

2.2 Differential Privacy

DEFINITION 2.1 (α-DIFFERENTIAL PRIVACY [4]). *A data access mechanism $\mathcal{A}$ satisfies α-differential privacy if for any neighboring databases*[1] *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 $Q(D)$,*

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

DEFINITION 2.2 (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,*

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

A commonly used mechanism to achieve differential privacy is the Laplace mechanism [7] that return $Q(D) + Y$ in place of the original result $Q(D)$ where Y is a random noise of Laplace distribution $Lap(GS_Q/\alpha)$ [7].

2.3 Composition

The composability of differential privacy [23] 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 2.1 (SEQUENTIAL COMPOSITION [23]). *Let M_i each provide α_i-differential privacy. The sequence of M_i provides $(\sum_i \alpha_i)$-differential privacy.*

THEOREM 2.2 (PARALLEL COMPOSITION [23]). *If $\boldsymbol{x_i}$ are disjoint subsets of the original database and M_i provides α-differential privacy for each $\boldsymbol{x_i}$, then the sequence of M_i provides α-differential privacy.*

[1]We use the definition of neighboring databases consistent with [23] which treats the databases as multisets of records and requires their symmetric difference to be 1.

2.4 Differentially Private Interface

Our approach is built on top of a differentially private interface such as PINQ [23]. The salient advantage of such an interface is that it guarantees that any of its outputs is differentially private. Users are freed from potential privacy concerns given the composable privacy guarantee from the interface and consequently can concentrate on the utility aspect. The interface provides operators for database aggregate queries such as count (**NoisyCount**) and sum (**NoisySum**). Each of these operators can be implemented by the Laplace mechanism or other mechanisms such as exponential mechanism [25]. Of particular importance to our approach is the **NoisyCount** operator which we assume is implemented by the Laplace mechanism. Another operator important to our approach is the **Partition** operator which is strongly related to the property of parallel composition. It partitions a data set into multiple disjoint subsets according to user-defined partitioning keys. According to the parallel composition property, the privacy cost of queries posed on disjoint partitions does not accumulate, but is determined by the worst.

An important factor of using the interface is privacy budgeting. Given the total privacy budget specified by the data publisher, we need to properly allocate the budget to a set of queries. The budget should be efficiently used such that the magnitude of noise added could be minimized so that better utility could be achieved.

3. DATA RELEASE

In this section, we describe our data release strategy. As a running example, we use the original data shown in **Figure 2** with attributes including age and income. The domain value of age is 20∼30, 30∼40 and 40∼50; the domain value of income is $0{\sim}10K$, $10K{\sim}20K$ and $> 20K$. We can represent the data in a multi-dimensional OLAP (Online Analytical Processing) data cube. **Figure 2** shows the data represented in a two-dimensional count cube or histogram, in which each cell represents the population count corresponding to the age and income values. We can represent the cells in the original data cube by a vector $\mathbf{x}$. In our example, the vector $\mathbf{x}$ is shown in Equation (1).

$$\mathbf{x} = \begin{bmatrix} 10 & 21 & 37 & 20 & 0 & 0 & 53 & 0 & 0 \end{bmatrix}^T \quad (1)$$

ID	Age	Income
1	20~30	10~20K
2	30~40	30K
...	...	...
n	30~40	60K

Figure 2: Example original data represented in a relational table (left) and a 2-dimensional count cube (right)

Our main goal is to release a set of data cubes such that the utility is maximized for random counting queries given *a predefined privacy budget*. Note that the data cube consists of disjoint cells or partitions, the privacy parameters used for obtaining a noisy count in each cell or partition for a single cube do not add up due to the parallel composition. However, the privacy parameters used for releasing multiple cubes will add up due to the serial composition. Our approach consists of three steps: 1) domain-based cell partitioning or Identity matrix query strategy (I) for releasing a cell histogram, 2) smoothing (S) of the cell histogram, and 3) sub-cube partitioning (P) based on the cell histogram for releasing a sub-cube histogram. The advantage of our approach, compared to a hierarchical approach such as in [16], are two folds: 1) the sub-cube histogram is adaptively determined by the cell histogram which exploits the underlying data distribution indirectly observed through the differentially private interface, 2) the smoothing and the sub-cube partitioning utilize techniques from image processing and group neighboring cells with similar density to enable the neighborhood reference in the estimation step to boost accuracy, 3) the privacy budget only needs to be divided between the two steps for releasing the noisy cell histogram and sub-cube histogram respectively. We next present each of the steps in detail with examples.

3.1 Cell Partitioning

The first step 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. Given an overall privacy budget α, we can allocate α_1 to this step, and use remaining budget $\alpha - \alpha_1$ for the sub-cube histogram release. Algorithm 1 shows the process for this step.

Algorithm 1 Cell partitioning release

Require: α_1: privacy budget

1. **Partition** the data based on all domains.
2. release **NoisyCount** for each partition using privacy parameter α_1.

To facilitate our discussion for estimation algorithms, we also represent our query strategy using the query matrix introduced in [20]. A *query* is a linear combination of $\mathbf{x}$. For example, a query Q with a predicate covering x_5 and x_6 can be represented as $Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$. A *query matrix* $\mathbf{Q}$ is a set of queries with each row representing a query.

The cell partitioning query strategy can be represented as the Identity matrix $\mathbf{I}$. The released data cube can be represented as a perturbed vector $\mathbf{y}_I = \mathbf{I}\mathbf{x} + \tilde{\mathbf{N}}_I$ where $\tilde{\mathbf{N}}_I$ is a vector of noises from the Laplace mechanism determined by the privacy parameter α_1. Using our example, Equation (2) shows the Identity matrix $\mathbf{I}$ and the released cell histogram $\mathbf{y}_I$. The released data is also illustrated as a two-dimensional cube in the left of **Figure 3**.

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{y}_I = \begin{bmatrix} 10 + \tilde{N}_1 \\ 21 + \tilde{N}_2 \\ 37 + \tilde{N}_3 \\ 20 + \tilde{N}_4 \\ 0 + \tilde{N}_5 \\ 0 + \tilde{N}_6 \\ 53 + \tilde{N}_7 \\ 0 + \tilde{N}_8 \\ 0 + \tilde{N}_9 \end{bmatrix} \quad (2)$$

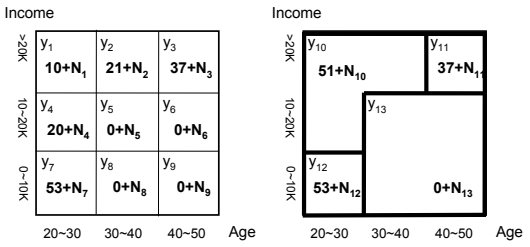


Figure 3: Released cell histogram (left) and sub-cube histogram (right)

3.2 Smoothing

Smoothing, also called "denoising" in image processing, is to remove noise from $\mathbf{y}_I$. The purpose of this step is to diminish the effects of noise $\tilde{\mathbf{N}}_I$ to show a clearer distribution trend. Note that the smoothed distribution is only used for the subsequent partitioning strategy, not for data release.

Many smoothing methods could be used, for example, Gaussian filtering [14], Median filtering [14] and curvature based level set method [26]. In this paper, we adopt the gaussian filtering method. Appendix B shows the details about the gaussian filtering method.

3.3 Sub-cube Partitioning

Our next step is to find a sub-cube partitioning of the data points based on the observed distribution of the data from the cell partitioning step. The goal is to group cells with similar counts together such that the generalized data cube can be used to answer or estimate random count queries. We adopt a density based partitioning algorithm, in particular, the region growing algorithm [12], for this purpose. The main reason we chose the region growing algorithm instead of other similar algorithms such as DBSCAN [9] is that not only the regions with similar density, but also the sparse regions need to be clustered together. We introduce the notion of connected region below.

DEFINITION 3.1 (CONNECTED REGION). *For a given cell C_i, the connected region of C_i include all the cells C_j with $dist(C_i, C_j) \leq d$ where $dist()$ is the distance function of two cells such as Euclidean distance and d is a threshold.*

Using our example in **Figure 2**, if $d = 1$, then the connected region for cell x_5 is x_2,x_4,x_6 and x_8; if $d = \sqrt{2}$, then all cells are connected region for cell x_5. Similarly, it can be extended to high dimensional space. The region growing algorithm will then group the similar cells together. We can see that $\{x_5, x_6, x_8, x_9\}$ is a continuous region. Using the resulting partitioning keys, we can then get the noisy count for the partitions.

Using the query matrix representation, our sub-cube partitioning strategy can be expressed as $\mathbf{P}$ in Equation (3) and the released cube can be expressed as $\mathbf{y}_P = \mathbf{P}\mathbf{x} + \tilde{\mathbf{N}}_P$. The released cube based on the sub-cube partitioning is also illustrated on the right in **Figure 3**.

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \mathbf{y}_P = \begin{bmatrix} 51 + \tilde{N}_{10} \\ 37 + \tilde{N}_{11} \\ 53 + \tilde{N}_{12} \\ 0 + \tilde{N}_{13} \end{bmatrix} \tag{3}$$

Algorithm 2 Sub-cube partitioning release

Require: $Sm(\mathbf{y}_I)$: smoothed data, d: threshold for connected region; ξ: threshold for region growing; $\alpha - \alpha_1$: privacy budget

1. Let $\mathbf{P}$ be the set of partitions;
while true **do**
  if $\exists$ not clustered cell C_i **then**
    Besides $\mathbf{P}$, generate new partition P_{new} containing C_i;
    Add P_{new} to $\mathbf{P}$;
    for each C_j in region P_{new} **do**
      for each C_k connected with C_j **do**
        if $Sm(\mathbf{y}_I)[C_k] - Sm(\mathbf{y}_I)[C_j] < \xi$ **then**
          Add C_k into region P_{new};
          Label C_k as clustered;
        end if
      end for
    end for
  else
    break;
  end if
end while
2. **Partition** the data based on $\mathbf{P}$.
3. release **NoisyCount** for each partition using privacy parameter $\alpha - \alpha_1$.

3.4 Summary and Privacy Guarantee

Putting things together, the overall query strategy of our approach can be represented as matrix $\mathbf{A}$ and the released data as $\mathbf{y}$ shown in Equation (4).

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} \\ \mathbf{P} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_I \\ \mathbf{y}_P \end{bmatrix} \tag{4}$$

THEOREM 3.1. *Released data $\mathbf{y}$ is α-differentially private.*

PROOF. Since $\mathbf{y}_I[i]$ and $\mathbf{y}_P[i]$ provide α_1 and $(\alpha - \alpha_1)$-differential privacy respectively for disjoint subsets of the original database, by Theorem 2.2, $\mathbf{y}_I$ and $\mathbf{y}_P$ are α_1 and $(\alpha - \alpha_1)$-differentially private. Then according to Theorem 2.1, $\mathbf{y}$ consisting of $\mathbf{y}_I$ and $\mathbf{y}_P$ is α-differentially private. □

4. QUERY ESTIMATION

Given the query strategy and the released data $\mathbf{y}$, our next question is how to answer an arbitrary user query Q using $\mathbf{y}$ such that the error is minimized. We formulate the problem as a probabilistic inference problem. Given a target query Q, suppose the original answer to the query is $\theta = Q\mathbf{x}$. Given the released data $\mathbf{y}$, we have potentially multiple ways to derive an answer to Q. We model the set of answers, $\tilde{\theta}_1, \tilde{\theta}_2, \cdots, \tilde{\theta}_n$, as a set of observations, denoted as $\tilde{\Theta}$, about the real value θ. Our task is to estimate an answer $\hat{\theta}$ such that $\mathbf{min}\ \mathbf{E}(\hat{\theta} - \theta)^2$. We propose an algorithm based on the minimum mean squared error (MMSE) estimator for such purpose, which optimizes the error between the estimated answer $\hat{\theta}$ and the original answer θ . According to the MMSE estimator, our goal is to compute

$$\hat{\theta} = E(\theta|\tilde{\Theta}) = E(\theta|\tilde{\theta}_1, \tilde{\theta}_2, \cdots, \tilde{\theta}_n). \tag{5}$$

A key point to note is that we are using an estimator to minimize the error between the estimated answer and

the real answer. In contrast, the existing solution based on consistency check [16] uses a Least Square (LS) approach to solve $\hat{\theta}$ which finds $\hat{\theta}$ to achieve $min||\tilde{\Theta} - \mathbf{y}||_2$ where $\mathbf{y}$ is the *released or perturbed data*.

Our algorithm consists of the following two main steps which we explain in detail in the subsections.

1. compute the possible answers or observed values $\tilde{\Theta}$ for θ given a query Q
2. estimate the answer $\hat{\theta}$ for θ given the observed values $\tilde{\Theta}$

4.1 Computing Observed Answers

Given a query Q, how do we find potential answers, or the observations, using $\mathbf{y}$? In other words, we need to find a transformation matrix $\mathbf{B}$ so that all the rows in $\mathbf{By}$ are representations of the query answer θ:

$$\tilde{\theta}_i = B_i\mathbf{y} = \theta + \tilde{N}_i \tag{6}$$

where $\tilde{N}_i$ is a linear combination of Laplace noises determined by B_i.

We present the solution we take in ISPE for different kinds of queries below. A query Q can be one of the three kinds: cell query, partition query and random range (sub-cube) query within a partition or across many partitions. A cell query is any row in matrix $\mathbf{I}$; a partition query is any row in matrix $\mathbf{P}$; a random query is any random vector of boolean elements.

For a partition query, cell query or a random query within a partition, the two released cubes, $\mathbf{y}_I$ and $\mathbf{y}_P$, correspond to two sources of observed values. For each partition, the relationship of $\mathbf{y}_I$ and $\mathbf{y}_P$ is like Figure 5 where there is a partition count in $\mathbf{y}_P$ as the parent node and its corresponding cell counts in $\mathbf{y}_I$ as children. We use the notation y_p as the partition count and $\mathbf{y}_c$ as the cell counts of one partition. For a partition query, y_p and $\sum \mathbf{y}_c$ would be the two observations. For a cell query or a random query within a partition, 1) the cell(s) composing the partition can be an observed value; 2) if we denote the sum of the rest of the cell counts besides Q by *sibling*, then $y_p - sibling$ is another observed value.

In addition, as initially designed in our data release, we can use the connected neighborhood of a cell to estimate the cell due to their similarity, which we call *neighborhood reference*. This means all the connected cells in the same region can be considered as observed values of the target cell. Algorithm 3 shows a sketch of the algorithm.

Algorithm 3 Computing observed answers for a partition query, cell query or random query within a partition

Require: Q, y_p: partition count, $\mathbf{y}_c$: cells count
1. $\tilde{\theta}_1 \leftarrow Q\mathbf{y}_I$
2. $\tilde{\theta}_2 \leftarrow y_p - sibling$ where *sibling* is the sum of the counts of the cells that are not covered by Q in the partition
3. If it's a cell query, $i \leftarrow 3$;

for each connected cell in the same partition **do**
  $\tilde{\theta}_i \leftarrow$count of the connected cell;
  $i \leftarrow i + 1$;
end for
return $\tilde{\Theta}$ and $\mathbf{B}$

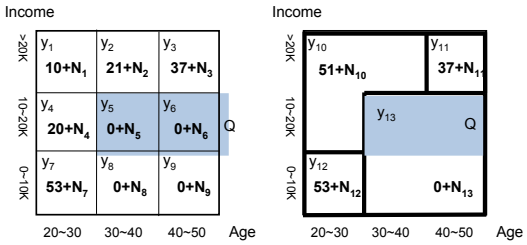


Figure 4: Example query given released data cubes

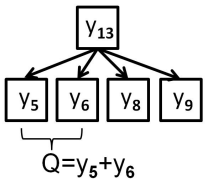


Figure 5: Relationship between the Counts

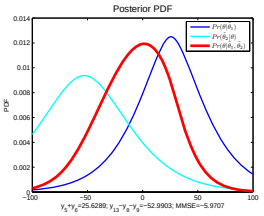


Figure 6: Estimated Answer

We illustrate our approach through an example. Given our query strategy and released data $\mathbf{y}$, we want to estimate the answer for a **random** query $Q\mathbf{x} = x_5 + x_6$, denoted as θ. **Figure 4** shows the query region of Q on the two released data cubes. The partition count y_p of this region is y_{13}; the cells' counts of this partition are $\mathbf{y}_c = [y_5; y_6; y_8; y_9]$. **Figure 5** shows the parent/child relationship between the partition count and the cell count. Using the above algorithm, we have

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\tilde{\Theta} = \mathbf{By} = \begin{bmatrix} \tilde{\theta}_1 \\ \tilde{\theta}_2 \end{bmatrix} = \begin{bmatrix} y_5 + y_6 \\ -y_8 - y_9 + y_{13} \end{bmatrix} = \begin{bmatrix} \theta + (\tilde{N}_5 + \tilde{N}_6) \\ \theta + (\tilde{N}_{13} - \tilde{N}_8 - \tilde{N}_9) \end{bmatrix}$$

For a random query across multiple partitions, we can first divide the query into several parts with every part belonging to one partition. Then we can use the above method to estimate each part. Finally, the sum of all parts are the estimation of Q.

4.2 Estimating Original Answer

The next question is how to compute the estimate given the observed answers? We first introduce two theorems before presenting our algorithm.

THEOREM 4.1. *Let* $Pr_i(\theta) = Pr(\theta|\tilde{\theta}_1, \tilde{\theta}_2, \cdots, \tilde{\theta}_i)$,

$$Pr_i(\theta) = Pr_{i-1}(\theta)\frac{Pr(\tilde{\theta}_i|\theta)}{Pr(\tilde{\theta}_i)} \tag{7}$$

Proof in appendix D

THEOREM 4.2. *Let $Pr_{\tilde{N}_i}()$ be the PDF of $\tilde{N}_i$ in Equation (6), then*

$$Pr(\tilde{\theta}_i|\theta) = Pr_{\tilde{N}_i}(\tilde{\theta}_i - \theta) \quad (8)$$

Proof in Appendix D.

Given the above theorems, we present an iterative algorithm as shown in Algorithm 4. We first use $\tilde{\theta}_1$ to compute $Pr(\theta|\tilde{\theta}_1)$. Next we use $\tilde{\theta}_2$ to compute $Pr(\theta|\tilde{\theta}_1, \tilde{\theta}_2)$. We continue until we compute $Pr(\theta|\tilde{\Theta})$ and $\hat{\theta}$.

Algorithm 4 Query estimation for a target query Q

Require: Q, $\mathbf{A}$, $\mathbf{y}$, $\tilde{\Theta}$, $\mathbf{B}$, α
1. Initialize $Pr_0(\theta)$ with a uniform distribution;
2. Compute $Pr_i(\theta)$ where $Pr_i(\theta) = Pr(\theta|\tilde{\theta}_1, \tilde{\theta}_2, \cdots, \tilde{\theta}_i)$:
for each $\tilde{\theta}_i$ in **By do**
(1) Compute $Pr_{\tilde{N}_i}()$;
(2) Compute $Pr(\tilde{\theta}_i|\theta)$ by Theorem 4.2;
(3) Compute $Pr_i(\theta)$ by Theorem 4.1;
end for
3. $\hat{\theta} = E(\theta|\tilde{\theta}_1, \tilde{\theta}_2, \cdots, \tilde{\theta}_{last}) = \int \theta Pr_{last}(\theta) d\theta$
return $\hat{\theta}$

Computing linear combination of multiple Laplace noises. One key issue for the above algorithm is that we need to compute the PDF of the *linear combination of multiple Laplace noise* $\tilde{N}_i$, for instance, $\tilde{N}_5 + \tilde{N}_6$ in our example. By the equality of Bilateral gamma distribution [18] and sum of n i.i.d. Laplace distribution, we have the following theorem:

THEOREM 4.3. *The PDF of sum of n i.i.d Laplace noises is*

$$f_n(z) = \frac{1}{2^n b^n \Gamma^2(n)} exp(-\frac{|z|}{b}) \int_0^\infty v^{n-1}(|z| + \frac{vb}{2})^{n-1} e^{-v} dv \quad (9)$$

Proof in Appendix E

Unfortunately this is difficult to compute when n becomes large. To circumvent this difficulty, we simulate the **discrete** PDF of linear combination of n i.i.d Laplace noises. The detailed algorithms are included in Appendix F with sample results showing that the simulated PDF is very close to the theoretical PDF for small n values. It is worth mentioning that in order to construct the PDF of the linear combination of Laplace noises, we need the privacy parameter of α_1 and α which determine the magnitude of the Laplace noises. We note that no matter whether we release α, the released data is differentially private.

Example. We illustrate the estimation process using our example. We first assume the distribution of θ is uniform. After having $\tilde{\theta}_1 = y_5 + y_6 = \theta + \tilde{N}_5 + \tilde{N}_6$, we can estimate the posterior PDF of θ, $Pr(\theta|\tilde{\theta}_1)$. Similarly, we add $\tilde{\theta}_2$ to our estimation by Theorem 4.1, we have $Pr_2(\theta)$, which can be used to make an estimation: $\hat{\theta}_2 = E(\theta|\tilde{\theta}_1, \tilde{\theta}_2) = \int \theta Pr_2(\theta) d\theta$. If we have other $\tilde{\theta}$ in $\tilde{\Theta}$, by adding them all, we compute $\hat{\theta} = \int \theta Pr_{last}(\theta) d\theta$.

To interpret this process by true values, we assume $\alpha_1 = 0.05$, $y_{13} = -5.2668$ and $[y_5, y_6, y_8, y_9]^T = [-15.4486, 41.0775, -2.9380, 50.6616]^T$, then we compute $Pr_1(\theta)$ shown in Figure 6. To compute $Pr(\tilde{\theta}_2|\theta)$, we need to compute $Pr(y_8 + y_9|x_8 + x_9)$ and $Pr(y_{13}|x_{13})$ first. By adding $Pr(\tilde{\theta}_2|\theta)$ to Equation (7), we compute $Pr_2(\theta)$. Finally, we have $\hat{\theta} = -5.9707$.

It is important to note that after having the posterior PDF of θ, not only can we compute the point estimation of θ, but also we can compute the interval estimation.

Table 1: Parameters tuning

Name	Value	Other Parameters	Figures
Privacy budget	α_1, α_2	$n_{it} = 5$; $\xi = 6$;	7, 8, 9
Iteration time in smoothing	n_{it}	$\alpha_1 = 0.15$; $\alpha_2 = 0.05$; $\xi = 6$	10,11,12
Threshold in Partitioning	ξ;	$\alpha_1 = 0.15$; $\alpha_2 = 0.05$; $n_{it=5}$	13,14,15

5. EXPERIMENT

5.1 Experiment Setup

5.1.1 Data

We use three kinds of data: continuous data, random data and real world data. The continuous data is generated by ourselves on the assumption of continuous distribution; the random data is generated without assumption of distribution. The real world data is part of the UCI Adult data for our experiments. For simplicity, we use two dimensional data. However, the algorithm can be easily extended to high dimension. To be fair, once we perturbed the original data, we **keep the perturbed data stable for all other parameters**.

5.1.2 Query

Three kinds of queries: cell query, partition query and random range query(in or across partitions). Because of **random queries** and **random noises**, we ask 100 random queries and use the average error. To be fair, once we generate the random queries, we **keep these queries stable for all other parameters**.

5.1.3 Accuracy

We use absolute error: $|\hat{\theta} - \theta|$, to measure the accuracy in our experiment. Y-axis presents error in all the following figures.

5.2 Parameters tuning

To make sure the neighborhood reference works, the parameters of partitioning strategy needs to tuned.

We introduce the parameters used in our experiment in table 1.

5.2.1 α

If we fix $\alpha = \alpha_1 + \alpha_2 = 0.2$, then Figure 7, 8 and 9 show then error changing trend with α_1. Although it's not clear in random data, we can see that there is an optimal point for α_1 in continuous data.

5.2.2 n_{it} on Gaussian filtering

In figure 11, error declines when n_{it} increases; while in figure 12, error increases when n_{it} increases. Because we

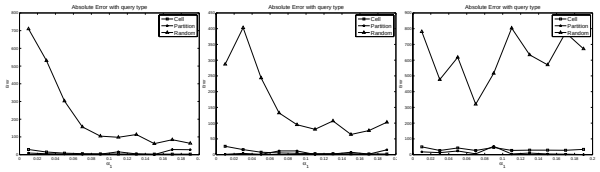


Figure 7: α_1 and α_2 on Adult data

Figure 8: α_1 and α_2 on continuous data

Figure 9: α_1 and α_2 on random data

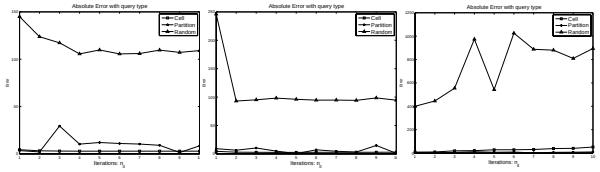


Figure 10: n_{it} on Adult data

Figure 11: n_{it} on continuous data

Figure 12: n_{it} on random data

cannot know the exact distribution of original data, error may first decline then incline gradually.

5.2.3 ξ

It's easy to understand that error declines when ξ increases for continuous data as figure 14. However, the trend is not clear for random data. Then error may first decline then remain stable for real world data as figure 13.

5.3 Comparison with other method

We compare ISPE with consistency check[16]. Note that the perturbed data and queries are the same in the comparison. First, we use the partitioning strategy of ISPE to compare. Because the goal of this partitioning is to use neighborhood reference, we use MMSE(NR) to denote the method in this step.

Second, we use the hierarchical partitioning strategy in [16]. Because we cannot use neighborhood reference in this partitioning, we use MMSE to denote the method in this step.

5.3.1 Estimation method

In real world data and continuous data, ISPE outperforms consistency check because of the neighborhood reference as figure 16 and 17. However, because it's nearly impossible to reference neighbor in random data, the two method performs equally for random queries as figure 18.

5.3.2 Query Range Size

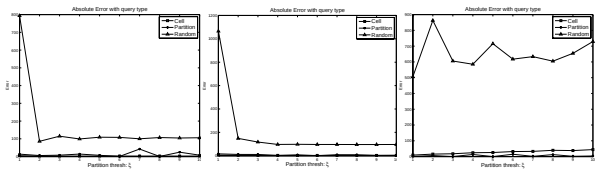


Figure 13: ξ on Adult data

Figure 14: ξ on continuous data

Figure 15: ξ on random data

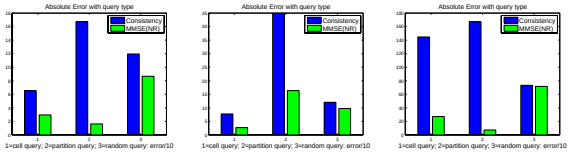


Figure 16: Queries on Adult data

Figure 17: Queries on continuous data

Figure 18: Queries on random data

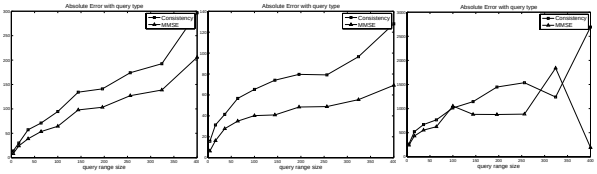


Figure 19: Query range size on Adult data

Figure 20: Query range size on continuous data

Figure 21: Query range size on random data

If we limit the range size of random queries, the error increases gradually. In real world data and continuous data, error of ISPE increases slower than consistency check as figure 19 and 20. However, in random data, the trend is not very clear.

5.3.3 Hierarchical partitioning

If we use the (K,L)-tree in [16] to partition data, one question is how to divide α in each height. If we divide α equally for each height, $\alpha_h = \alpha/L$, then the two method performs almost equally as figure 22; if we divide α as $\alpha_h = \alpha * 2^{h-L-1}$, then MMSE outperforms as figure 23.

6. RELATED WORKS

Privacy preserving data analysis and publishing has received considerable attention in recent years. We refer readers to [5, 11, 6] for several up-to-date surveys. The early approaches in statistical databases is summarized in [1]. A large body of literature on privacy preserving data publishing [11] adopts a relaxed *adversarial* or *Bayes-optimal privacy* notion [22] by considering specific types of attacks (*attack specific*) and assumes the attacker has limited background knowledge (*background knowledge sensitive*). Differential privacy [5, 6] has emerged recently and is widely ac-

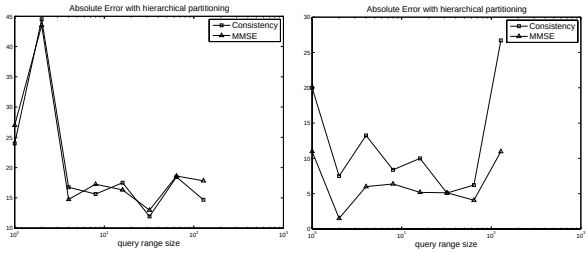


Figure 22: Hierarchical partitioning: $\alpha_h = \alpha/l$

Figure 23: Hierarchical partitioning: $\alpha_h = 2^{h-l-1}\alpha$

cepted as one of the strongest known unconditional privacy guarantees. [23] developed a programmable interfaces called PINQ which provides an encapsulated differentially private interface to raw data and lays a foundation for developing differentially private data release and analysis.

A few works started addressing non-interactive data release that achieves differential privacy. Blum et al. [3] proved the possibility of non-interactive data release satisfying differential privacy for queries with polynomial VC-dimension, such as predicate queries. However, the result remains theoretical and the general algorithm is inefficient for the complexity and required data size. Feldman et al. [10] proposed the notion "private coreset" to release data for certain queries. Machanavajjhala et al. [21] showed a method to release synthetic data of the commuting patterns of the population, which statistically mimic the original data without formal demonstration of the utility of the synthetic data. [8] further proposed more efficient algorithms with hardness results obtained for non-interactive differentially private data release. Several recent work studied differentially private mechanisms for particular kinds of tasks such as term counting in search logs [19, 13], recommender systems [24] or record linkage [17].

Closely related to our approach are several recent works considering a data release strategy for predicate counting queries. X. Xiao et al. [28] developed an algorithm using wavelet transforms. [16] generates differentially private histograms for single dimensional range queries through a hierarchical release and consistency check technique. [20] proposes a query matrix mechanism that generates an optimal query strategy based on a known query workload of linear count queries and further mapped the work in [28] and [16] as special query strategies that can be represented by hierarchical and wavelet matrices respectively. Appendix G shows the Identity ($\mathbf{I}$), Hierarchical ($\mathbf{H}$) and Wavelet ($\mathbf{Y}$) matrices. It is important to note that the above mentioned query strategies are data-oblivious in that they are determined by the query workload, or a static hierarchical or wavelet matrix without taking into consideration the underlying data. Our earlier preliminary work [29] explored a cell-based partitioning approach and a kd-tree like partitioning approach. This paper extends the work in several aspects: 1) it uses smoothing and density-based partitioning to exploit the similarity between cells, 2) it introduced the new estimation component for effective query answering. Appendix 3 shows a summary comparison of these related approaches.

The idea of post-processing the output of a differentially private mechanism to ensure consistency was introduced in [2] and followed up in [16]. Recent work [27] observed the connection between probabilistic inference and differentially private algorithms and the potential of applying probabilistic inference to integrate multiple views generated by differentially private mechanisms to derive posterior distributions over the data sets and model parameters thereof. We take a similar viewpoint and treat the query answering as a probabilistic inference problem and present algorithms based on the MMSE estimator for inferring answers for particular user queries, instead of model parameters.

7. CONCLUSION

We presented the ISPE framework for adaptively releasing differentially private histograms and estimating arbitrary count queries with high accuracy. The experimental results demonstrated the feasibility and high accuracy of the approach using both synthetic data and real world dataset. Our future work are along several directions. First, we would like to study the impacts of different techniques for smoothing and density-based clustering and different estimators. Second, the current approach is more suitable for small range queries due to its cell histogram. The framework can be easily extended to generate hierarchical domain-based histograms along with the data-driven sub-cube partitioning histograms. Our hypothesis is that it will provide better results for sparse data and larger range queries. Finally we are interested in investigating approaches that are both data-aware and workload-aware to incorporate a given workload.

Acknowledgments

The research is partially supported by a Cisco research award and an International Collaboration Fund by Shenzhen China. We thank James Nagy, Wenshuo Zhou, Slawomir Goryczka, Michael Hay, Xiaokui Xiao and Andrzej Rucinski for their insightful discussions and help during the preparation of the paper.

8. REFERENCES

[1] N. R. Adams and J. C. Wortman. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 21(4), 1989.

[2] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*, pages 273–282, 2007.

[3] 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.

[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. A firm foundation for private data analysis. *Commun. ACM.*, 2010.

[7] 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.

[8] 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.

[9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. *In Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996.

[10] 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.

[11] 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.

[12] R. C. Gonzalez and R. Woods. *Digital Image Processing 2nd Edition*. Prentice Hall, New Jersey, 2002.

[13] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Privacy in search logs. *CoRR*, abs/0904.0682, 2009.

[14] R. A. Haddad and A. N. Akansu. A class of fast Gaussian binomial filters for speech and image processing. *IEEE Transactions on Signal Processing*, 39:723–727, Mar. 1991.
[15] M. Hardt and G. Rothblum. A multiplicative weights mechanism for interactive privacy-preserving data analysis. In *FOCS*, 2010.
[16] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially-private queries through consistency. In *VLDB*, 2010.
[17] 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.
[18] U. Kchler and S. Tappe. On the shapes of bilateral gamma densities. *Statistics & Probability Letters*, 78(15):2478–2484, October 2008.
[19] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *WWW*, 2009.
[20] 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.
[21] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. *2008 Ieee 24th International Conference on Data Engineering, Vols 1-3*, pages 277–286, 2008. 24th IEEE International Conference on Data Engineering APR 07-12, 2008 Cancun, MEXICO.
[22] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3, 2007.
[23] 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.
[24] 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.
[25] 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.
[26] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. 2002.
[27] O. Williams and F. McSherry. Probabilistic inference and differential privacy. In *Neural Information Processing Systems (NIPS)*, 2010.
[28] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, pages 225–236, 2010.
[29] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. *Secure Data Management*, 38(1), 2010.

APPENDIX

A. NOTATIONS

Q: Query matrix;
H: Hierarchical workload;
Y: wavelet workload;
I: Identity matrix;
A: query strategy matrix;
P: partitioning strategy matrix, $\mathbf{P} = Par(Sm(\mathbf{Ix}))$;
x: vector of original cell value;
$\tilde{\mathbf{N}}$: Laplace noise;
θ: original answer of a query Q, $\theta = Q\mathbf{x}$;
$\tilde{\theta}$: observed answers of a query Q, $\tilde{\theta} \in span(\mathbf{y})$[2];
$\hat{\theta}$: estimation of θ;
B: transformation matrix for θ so that $B_i y_i = \theta + B_i\tilde{\mathbf{N}}$;
$Pr_i(\theta)$: posterior PDF of θ at ith iteration;
$Pr_{\tilde{N}_i}$: the PDF of $\tilde{N}_i$ where $\tilde{N}_i$ is the linear combination of some Laplace noises;
α: sensitivity of differential privacy;
ϵ: small number, e.g. the smallest number for a computer;
GS: global sensitivity of a query;
b: parameter of Laplace distribution: $b = GS/\alpha$;
y: perturbed answers of query matrix, $\mathbf{Qx} + \tilde{\mathbf{N}}$;
PDF: probability density function;
$f_n(z)$: PDF of sum of n i.i.d. Laplace noise;
$D_{PDF}(P_s, P_t)$: the difference function of theoretical and simulated PDF.
$Par()$: Partition function;
$Sm()$: Smooth function;

B. GAUSSIAN FILTERING

Mathematically, a Gaussian filter modifies data by convolution with a Gaussian function. For example, in two dimensions, the Gaussian function[14] is like

$$g(x, y) = \frac{1}{2\pi\sigma^2} exp(-\frac{x^2 + y^2}{2\sigma^2}) \tag{10}$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution.

Gaussian filtering needs two steps to work: First, the mask size should be chosen to generate a gaussian mask; second, for each point, compute the convolution of gaussian mask, then replace the value with convolution.

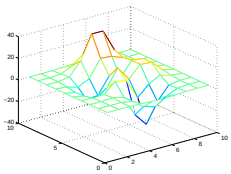
The gaussian mask is the gaussian function for a chosen size. For example, if database has two dimensions and we choose the size as 3, then use Equation (10) to compute the mask as following[14]. Note that in this paper, we only use the discrete function.

$$Gaussian(3) = \begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix}$$

After having the gaussian mask, we compute the convolution of each point and replace the value of the point with the convolution.

Figure 24 is the original data before smoothing. Figure 25 is the perturbed data before smoothing. Figure 26 and 27 are the smoothed data after 5 and 10 iterations. We can see that the distributions of smoothed data become clearer than perturbed data.

[2] $span(\mathbf{y}) = \lambda_1 y_1 + \lambda_2 y_2 + \cdots + \lambda_n y_n$



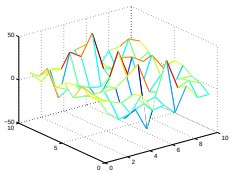
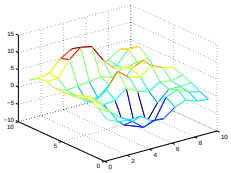


Figure 24: Original data before smoothing **Figure 25: Perturbed Data before smoothing**



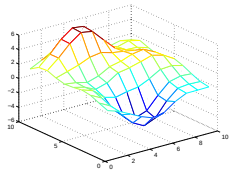


Figure 26: After 5 iterations of gaussian filtering **Figure 27: After 10 iterations of gaussian filtering**

C. INTRODUCTION OF MMSE

First, we need to compute the Bayes posterior probability $Pr(\theta|\tilde{\Theta})$:

$$Pr(\theta|\tilde{\Theta}) = \frac{Pr(\tilde{\Theta}|\theta)Pr(\theta)}{\int Pr(\tilde{\Theta}|\theta)Pr(\theta)d\theta} \tag{11}$$

Note that in above equation, we may not know $Pr(\theta)$ at the very beginning. If so, we assume $Pr(\theta)$ is uniformly distributed.

Our goal is the $\hat{\theta}$ to get the minimal $Error_\theta$.

$$Error_\theta = E(\hat{\theta} - \theta)^2 \tag{12}$$

$$\begin{aligned} Error_\theta &= \int\int (\hat{\theta} - \theta)^2 Pr(\theta, \tilde{\Theta}) d(\tilde{\Theta}) d\theta \\ &= \int Pr(\tilde{\Theta}) \int (\hat{\theta} - \theta)^2 Pr(\theta|\tilde{\Theta}) d\theta d(\tilde{\Theta}) \end{aligned}$$

To get $min(Error_\theta)$, we need $\int(\hat{\theta} - \theta)^2 Pr(\theta|\tilde{\Theta})d\theta$ to be minimal. Then,

$$\begin{aligned} &\frac{\partial}{\partial\hat{\theta}} \int (\hat{\theta} - \theta)^2 Pr(\theta|\tilde{\Theta}) d\theta \\ &= 2\int (\hat{\theta} - \theta) Pr(\theta|\tilde{\Theta}) d\theta \end{aligned}$$

We get

$$\hat{\theta} = \int \theta Pr(\theta|\tilde{\Theta}) d\theta = E(\theta|\tilde{\Theta}) \tag{13}$$

D. PROOF

Theorem 4.1. Let $Pr_i(\theta) = Pr(\theta|\tilde{\theta}_1, \tilde{\theta}_2, \cdots, \tilde{\theta}_i)$,

$$Pr_i(\theta) = Pr_{i-1}(\theta)\frac{Pr(\tilde{\theta}_i|\theta)}{Pr(\tilde{\theta}_i)} \tag{14}$$

PROOF.

$$\begin{aligned} Pr(\theta|\tilde{\Theta}_{l-1}, \tilde{\theta}_l) &= \frac{Pr(\theta)Pr(\tilde{\Theta}_{l-1}|\theta)Pr(\tilde{\Theta}_l|\theta)}{Pr(\tilde{\Theta}_{l-1})Pr(\tilde{\theta}_l)} \\ &= \frac{Pr(\tilde{\Theta}_{l-1})Pr(\theta|\tilde{\Theta}_{l-1})Pr(\tilde{\theta}_l|\theta)}{Pr(\tilde{\Theta}_{l-1})Pr(\tilde{\theta}_l)} \\ &= \frac{Pr(\theta|\tilde{\Theta}_{l-1})Pr(\tilde{\theta}_l|\theta)}{Pr(\tilde{\theta}_l)} \\ &= Pr_{l-1}(\theta)\frac{Pr(\tilde{\theta}_l|\theta)}{Pr(\tilde{\theta}_l)} \end{aligned}$$

□

Theorem 4.2 Let $Pr_{\tilde{N}_i}()$ be the PDF of $\tilde{N}_i$ in Equation (6), then

$$Pr(\tilde{\theta}_i|\theta) = Pr_{\tilde{N}_i}(\tilde{\theta}_i - \theta) \tag{15}$$

PROOF.

$$Pr(\tilde{\theta}_i|\theta) = Pr(\tilde{\theta}_i = \theta + \tilde{N}_i|\theta) = Pr(\tilde{N}_i = \tilde{\theta}_i - \theta|\theta) \tag{16}$$

Note that the shape of $Pr_{\tilde{N}_i}()$ doesn't change with θ, therefore,

$$Pr(\tilde{N}_i = \tilde{\theta}_i - \theta|\theta) = Pr_{\tilde{N}_i}(\tilde{\theta}_i - \theta) \tag{17}$$

□

E. LAPLACE RELATED PDF

We denote PDF of exponential distribution for $x \geq 0$ *and* $b \geq 0$ as

$$Pr(x, b) = \frac{1}{b}exp(-\frac{x}{b}); \tag{18}$$

PDF of Laplace distribution as

$$Pr(x, b) = \frac{1}{2b}exp(-\frac{|x|}{b}); \tag{19}$$

PDF of Gamma distribution as

$$Pr(x, n, b) = x^{n-1}\frac{exp(-x/b)}{b^n\Gamma(n)}. \tag{20}$$

where $\Gamma(n) = (n-1)!$.

LEMMA E.1. *To draw a Laplace noise $\tilde{N}$ from Laplace distribution, it's equivalent to draw two i.i.d. variables X_1 and X_2 with exponential distribution and let $\tilde{N} = X_1 - X_2$.*

LEMMA E.2. *Let $X_1, X_2, \cdots X_n$ are i.i.d. exponential variables, then $Y = \sum_i^n X_i$ has Gamma distribution.*

LEMMA E.3. *Let $X_1, X_2, \cdots X_n, X_{n+1}, \cdots, X_{2n}$ are i.i.d. exponential variables, then*

$$Z = \sum_{i=1}^{n}(X_i - X_{n+i}) = \sum_{i=1}^{n} X_i - \sum_{i=n+1}^{2n} X_i$$

which means sum of Laplace variables has the same PDF with subtraction of Gamma variables. Z is called bilateral gamma variables[18].

Lemma E.3 demonstrates the equality of Bilateral gamma distribution and sum of n i.i.d. Laplace distribution. According to [18], we have Equation (9). By assigning particular numbers to n, we have the following corollaries.

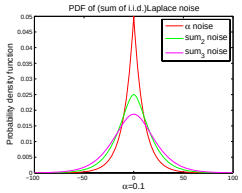


Figure 28: PDF. sum_n noise: sum of n i.i.d. Laplace noise

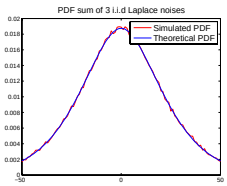


Figure 29: difference between theoretical and simulated PDF

COROLLARY E.1. *The PDF of sum of 2 i.i.d Laplace noises is*

$$f_2(z) = (\frac{1}{4b} + \frac{|z|}{4b^2})exp(-\frac{|z|}{b}) \tag{21}$$

The PDF of sum of 3 i.i.d Laplace noises is

$$f_3(z) = (\frac{3}{8} + \frac{3|z|}{8b} + \frac{z^2}{8b^2})exp(-\frac{|z|}{b}) \tag{22}$$

Figure 28 shows sum of two and three i.i.d. Laplace noises.

F. SIMULATING THE PDF FOR LINEAR COMBINATION OF LAPLACE NOISES

Imagine we know the continuous PDF Pr(z) and compute $Pr(k) = \int_{k-1/2}^{k+1/2} Pr(z)dz$ where $k \in \mathbb{R}$. Then **Pr** becomes an infinite vector. If we delete any $Pr(k) < \epsilon$ where ϵ is a small number, then **Pr** becomes a finite vector. The meaning of **Pr** is this: for a given integer k, $Pr(k) = Pr[k]$. For simplicity, in this paper we ignore the difference between continuous PDF and discrete PDF, which are equivalent for our problem.

Take sum of n i.i.d. noises for example. Suppose $[s_1, s_2, \cdots, s_n]$ are a $m \times n$ matrix in which every element is an i.i.d. Laplace noise, then the PDF of n i.i.d Laplace noises is the PDF of $X = \sum_{i=1}^{n} s_i$. Theoretically, if the m is big enough, the simulated PDF will be the same as the theoretical PDF. Algorithm 5 shows how to simulate discrete PDF of sum of n i.i.d Laplace noises. Note that we can use Lemma E.1 in Appendix E to draw Laplace noise.

Algorithm 5 Simulate discrete PDF of sum of n i.i.d Laplace noises $\sum_{j=1}^{n} \tilde{N}_j$

1. Draw a matrix $[\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_n] \in \mathbb{R}^{m \times n}$ of Laplace variables with PDF=$f_1(z) = \frac{1}{2b}exp(-\frac{|z|}{b})$.
2. compute $\mathbf{z} = \sum_{j=1}^{n} \mathbf{s}_j$.
3. $z = round(z)$.
4. $f_n(z) = \frac{hist(\mathbf{z})}{mn}$.

return simulated PDF $f_n(z)$

To measure the precision of simulated PDF, we define the difference function D_{PDF} as:

DEFINITION F.1. *The difference function of simulated PDF and theoretical PDF is defined as:*

$$D_{PDF}(P_s, P_t) = \sum_{i \in (-\infty, \infty)} |P_s(i) - P_t(i)|$$

where P_s and P_t are the simulated and theoretical PDF, i is any integer.

If we draw 10^6 data points, which means $m \times n = 10^6$, the result is shown in Figure 29. $D_{PDF}(P_s, P_t) = 0.015$.

F.1 Linear combination of PDF

The more samples used, the higher the accuracy of the simulated PDF. Therefore, the computation cost for highly precise PDF would be expensive. Actually we can either simulate Laplace related PDF or compute discrete PDF of linear combination of n i.i.d. Laplace noises. Another reason for us to compute discrete PDF is that the posterior PDF will become different from Laplace distribution after computation with Equation (7).

We define this problem as follow:
$\tilde{\mathbf{N}} = [\tilde{N}_1, \tilde{N}_2, \cdots, \tilde{N}_n]$ are n noises and we know their PDF(may not necessarily be Laplace distribution). B_i is a linear representation[3] of $\tilde{\mathbf{N}}$, we want to know the PDF of $B_i\tilde{\mathbf{N}}$.

For this problem, we solve operator "$\pm$". Then $B_i\tilde{\mathbf{N}}$ can be solved by iteration. Algorithm 6 describes the process of computing PDF of linear combination of PDF.

Algorithm 6 compute linear combination of PDF

Require: $\tilde{\mathbf{N}}$ with discrete PDF $Pr_{\tilde{\mathbf{N}}}()$, B_i
1. $PL_1 \leftarrow Pr_{\tilde{N}_1}()$ where PL is the PDF of linear combination;
2.
for each $Pr_{\tilde{N}_j}()$ in $Pr_{\tilde{\mathbf{N}}}()$, $j \geq 2$ **do**
  add $Pr_{\tilde{N}_j}()$ to PL_{j-1} to generate PL_j:
  for each $k \in \mathbb{R}$ **do**
    for each $u, v \in \mathbb{R}$ where $u \pm v = k$ **do**
      $PL_j(k) \pm = PL_{j-1}(u)Pr_{\tilde{N}_j}(v)$;
    end for
  end for
end for
return discrete PDF: $Pr_{B_i\tilde{\mathbf{N}}}() = PL$

Note that we will not compute all $k \in \mathbb{R}$ in practice, need to break the loop when $PL_j(k) < \epsilon$.

G. QUERY MATRICES

Table 2: Identity (I), Hierarchical H and Wavelet Y Matrices

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

H. COMPARISON OF RELATED WORKS

[3]In this paper, the elements in **B** can only be 1,0 or -1.

Table 3: Comparison of related works

Related Work	Query Strategy	Estimation	Target Workload
Wavelet [28]	$\mathbf{Y}$		Random
Hierarchical + consistency [16]	$\mathbf{H}$	Consistency	Random
Query matrix mechanism [20]	$\mathbf{A}(\mathbf{W})$		$\mathbf{W}$
kd-tree [29]	$\begin{bmatrix} \mathbf{I} \\ kd(\mathbf{y}_I) \end{bmatrix}$		Random
ISPE	$\begin{bmatrix} \mathbf{I} \\ \mathbf{P}(\mathbf{S}(\mathbf{y}_I)) \end{bmatrix}$	MMSE	Random