Differentially Private Histogram Publication

Jia Xu[1], Zhenjie Zhang[2], Xiaokui Xiao[3], Yin Yang[2], Ge Yu[1]

[1]*College of Information Science & Engineering, Northeastern University, China*
{xujia,yuge}@ise.neu.edu.cn

[2]*Advanced Digital Sciences Center, Illinois at Singapore Pte., Singapore*
{zhenjie,yin.yang}@adsc.com.sg

[3]*School of Computer Engineering, Nanyang Technological University, Singapore*
xkxiao@ntu.edu.sg

***Abstract*— Differential privacy (DP) is a promising scheme for releasing the results of statistical queries on sensitive data, with strong privacy guarantees against adversaries with arbitrary background knowledge. Existing studies on DP mostly focus on simple aggregations such as counts. This paper investigates the publication of DP-compliant histograms, which is an important analytical tool for showing the distribution of a random variable, e.g., hospital bill size for certain patients. Compared to simple aggregations whose results are purely numerical, a histogram query is inherently more complex, since it must also determine its *structure*, i.e., the ranges of the bins. As we demonstrate in the paper, a DP-compliant histogram with finer bins may actually lead to significantly lower accuracy than a coarser one, since the former requires stronger perturbations in order to satisfy DP. Moreover, the histogram structure itself may reveal sensitive information, which further complicates the problem.**

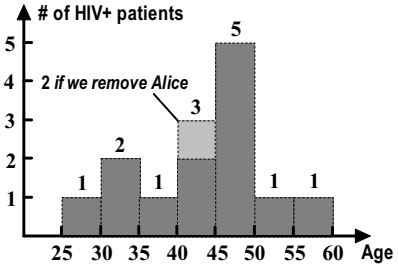
Motivated by this, we propose two novel algorithms, namely *NoiseFirst* and *StructureFirst*, for computing DP-compliant histograms. Their main difference lies in the relative order of the noise injection and the histogram structure computation steps. NoiseFirst has the additional benefit that it can improve the accuracy of an *already published* DP-complaint histogram computed using a naïve method. Going one step further, we extend both solutions to answer arbitrary range queries. Extensive experiments, using several real data sets, confirm that the proposed methods output highly accurate query answers, and consistently outperform existing competitors.

I. Introduction

Digital techniques has enabled various organizations to easily gather vast amounts of personal information, such as medical records, web search history, etc. Analysis on such data can potentially lead to valuable insights, including new understandings of a disease and typical consumer behaviors in a community. However, currently privacy concerns is a major hurdle for such analysis, in two aspects. First, it increases the difficulty for third-party data analyzers to access their input data. For instance, medical researchers are routinely required to obtain the approval of their respective institutional review boards, which is tedious and time-consuming, before they can even look at the data they need. Second, privacy concerns complicate the publication of analysis results. A notable example is the dbGaP[1] database, which contains results of genetic studies. Such results used to be publicly available, until a recent paper [1] describes an attack that infers whether a person has participated in a certain study (e.g., on patients with diabetes) from its results; thereafter, access to such results is strictly controlled. Furthermore, a strengthened version of this attack [2] threatens the publication of any research paper on genome-wide association studies, which currently is an active field in biomedical research.

Name	Age	HIV+
Alice	42	Yes
Bob	31	Yes
Carol	32	Yes
Dave	36	No
Ellen	43	Yes
Frank	41	Yes
Grace	26	Yes
...	...	...

(a) Example sensitive data



(b) Unperturbed histogram

Fig. 1. Example sensitive dataset and its corresponding histogram

The recently proposed concept of differential privacy (DP) [3]–[8] addresses the above issues by injecting a small amount of random noise into statistical results. DP is rapidly gaining popularity, because it provides rigorous privacy guarantees against adversaries with arbitrary background information. This work focuses on the computation of DP-compliant histograms, which is a common tool for presenting the distribution of a random variable. Fig. 1(a) shows sample records in an imaginary sensitive dataset about HIV-positive patients, and Fig. 1(b) illustrates its histogram showing the age distribution of such patients. Such histograms are commonly found, e.g., in the published statistics by Singapore's Ministry of Health[2]. The application of DP to such histograms guarantees that changing or removing any record from the database has negligible impact on the output histogram. This means that the adversary cannot infer whether a specific patient (say, Alice) is infected by HIV, even if s/he knows the HIV status of all the remaining patients in the database.

A histogram with a given structure reduces to a set of disjoint range-count queries, one for each bin. The state-of-the-art method [3] (called the Laplace Mechanism, or LM) for perturbing the output of such counts to satisfy DP works as follows. First, LM determines the *sensitivity* Δ of these counts, which is the maximum possible changes in the query

[1]http://www.ncbi.nlm.nih.gov/gap

[2]http://www.moh.gov.sg/mohcorp/statistics.aspx

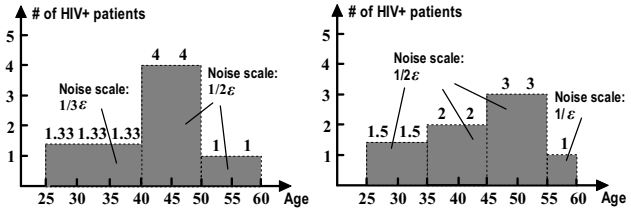
results if we remove one record from (or add one into) the database. In our example, we have $\Delta = 1$, since each patient affects the value of exactly one bin in the histogram by at most 1[3]. For instance, removing Alice decreases the number of HIV+ patients aged 40-45 by 1. Then, LM adds to every bin a random value following the Laplace distribution with mean 0 and scale Δ/ϵ, where ϵ is a parameter indicating the level of privacy. For instance, when $\epsilon = 1$, the noise added to each bin has a variance of 2 [9], which intuitively covers the impact (i.e., 1) of any individual in the database.

A key observation made in this paper is that the accuracy of a DP-compliant histogram depends heavily on its structure. In particular, a coarser histogram can sometimes lead to higher accuracy than a finer one, because the former's reduced noise scale required to satisfy DP. Fig. 2(a) exhibits another histogram of the dataset in Fig. 1(a) with 3 bins 25-40, 40-50 and 50-60, respectively. In the following, we use the term "unit-length range" to mean the range corresponding to a bin in the histogram in Fig. 1(b), e.g., 25-30. In the histogram in Fig. 2(a), each bin covers multiple unit-length ranges, and the numbers on top of each bin correspond to the *average* count of each unit-length range, e.g., 1.33 above range 25-30 is calculated by dividing the *total* number of patients (i.e., 4) in the bin 25-40 by the number of unit-length ranges it covers (i.e. 3). As we prove later in the paper, such averaging decreases the amount of noise therein. Specifically, the Laplace noise added to each unit-length range inside a bin covering b such ranges has a scale of $1/b \cdot \epsilon$, compared to the $1/\epsilon$ scale in the histogram of Fig. 1(b). On the other hand, the use of larger bins also introduces *information loss*, as estimates (e.g., 1.33 for range 25-30) replace their respective exact values (1).

Accordingly, the quality of a histogram structure depends on the balancing between information loss and noise scales. Fig. 2(b), for example, shows yet another histogram for the same dataset, which intuitively has poor accuracy because (i) it merges unit-length ranges with very different values, e.g., ranges 35-40 and 40-45, leading to high information loss; and (ii) it contains a very small bin, i.e., 55-60, that requires a high noise scale $1/\epsilon$, especially for small values of ϵ. This example implies that the best structure depends on the data distribution as well as ϵ. A further complication is that the optimal structure itself may reveal sensitive information, since removing a record from the original database may cause the optimal structure to change, which can be exploited by the adversary to infer sensitive information. Thus, simply selecting the best structure with an existing histogram construction technique violates DP, regardless of the amount of noise injected to the counts.

Facing these challenges, we propose two effective solutions for DP-compliant histogram computation, namely *NoiseFirst* and *StructureFirst*. The former determines the histogram structure *after* injecting random noise, whereas the latter reverses the order of these steps. In particular, NoiseFirst can be used to improve the accuracy of an already published histogram using an existing method [3]. Moreover, we adapt DP-histograms to answer arbitrary range-count queries, which has drawn considerable research attention (e.g., [3], [6]). For such queries, NoiseFirst achieves better accuracy for short ranges, whereas StructureFirst is more suitable for longer ones. Extensive experiments using several real data sets demonstrate that the proposed methods output highly accurate histograms, and significantly outperform existing methods for range count queries.

[3]An alternative definition of sensitivity [3] concerns the maximum changes in the query results after *modifying* a record in the database. In our example this leads to $\Delta = 2$, since in the worst case, changing a person's age can affect the values in two different bins by 1 each.



(a) Optimal histogram with 3 bins (b) Poor histogram structure

Fig. 2. Impact of histogram structures on noise scales

In the remainder of the paper, Section 2 provides necessary background on DP. Section 3 and 4 present NoiseFirst and StructureFirst, respectively. Section 5 discusses range-count query processing. Section 6 contains a thorough experimental study. Section 7 overviews existing DP techniques. Finally, Section 8 offers some concluding remarks.

II. Preliminaries

A. Histogram Construction

A histogram is commonly used in database systems as an effective statistical summarization tool on numerical domains. Given a series of counts in the database, i.e., $D = \{x_1, x_2, \ldots, x_n\}$ with $x_i \in \mathbb{R}^+$ for each i, histogram aims to merge the neighbor counts into k groups and identify a representative count for each group. To accomplish this, the histogram generates k bins, i.e., $H = \{B_1, B_2, \ldots, B_k\}$, over the integer domain $[1, n]$. Each bin $B_j = (l_j, r_j, c_j)$ covers an interval $[l_j, r_j] \subseteq [1, n]$ and maintains a count c_j that approximates all counts in D in the interval, i.e., $\{x_i \mid l_j \leq i \leq r_j\}$. A histogram H is valid, if and only if all the bins completely cover the domain $[1, n]$ without any overlap, i.e., $l_1 = 1$, $r_k = n$, and $r_j = l_{j+1} - 1$ for all $1 \leq j \leq n-1$. Given such a count sequence D, the error of the histogram is measured by the *Sum of Squared Error* (SSE) between the approximate counts and the original counts, i.e.,

$$Error(H, D) = \sum_j \sum_{l_j \leq i \leq r_j} (c_j - x_i)^2 \qquad (1)$$

Note that $Error(H, D)$ can be interpreted as the sum of the squared error for all range count queries. If the structure of the histogram is fixed, i.e., every l_j and r_j are known, the optimal value of c_j for each B_j equals the average count in B_j. That is, setting $c_j = \frac{\sum_{i=l_j}^{r_j} x_i}{r_j - l_j + 1}$ for all j leads to a minimum $Error(H, D)$. Accordingly, the problem of

i=1	2	3	4	5	6	7	
0	0.5	0.67	2.75	11.2	12.8	14	k=1
	0	0.5	0.67	2.67	8.67	11.2	2
		0	0.5	0.67	2.67	2.67	3

Fig. 3. Build of an optimal histogram on Figure 1(b). Each entry $T(i,k)$ keeps the minimal error for first i counts with exactly k bins

conventional *Histogram Construction* [10], [11] is to identify the optimal histogram structure to minimize $Error(H, D)$, as formalized in the following.

Problem 1. *Given the count sequence D and the size constraint k, find the optimal histogram H^* that minimizes $Error(H^*, D)$ among all histograms with exactly k bins.*

In [10], Jagadish et al. showed that the optimal histogram H^* for Problem 1 can be constructed using a *Dynamic Programming* technique, with time complexity $O(n^2k)$ and space complexity $O(nk)$. In Figure 3, we list the table storing the intermediate results of dynamic programming, based on the data sequence in Figure 1(b) and $k = 3$. In the table, each entry $T(i,k)$ is the minimal error of any histogram with exactly k bins covering the partial data set $D_i = \{x_1, \ldots, x_i\}$. In the following, we use $SSE(D, l, r)$ to denote the sum of square error if we merge a partial sequence $\{x_l, \ldots, x_r\}$ into a single bin. Specifically, since the average count on the interval is $\bar{x}(l,r) = \sum_{i=l}^{r} x_i/(r-l+1)$, we have $SSE(D,l,r) = \sum_{i=l}^{r}(x_i - \bar{x}(l,r))^2$

Therefore, the construction of the table utilizes the following recursive rule in dynamic programming:

$$T(i,k) = \min_{k-1 \le j \le i-1} \left(T(j, k-1) + SSE(D, j+1, i)\right)$$

Based on the intermediate results in the table, we build the optimal histogram by tracing back the optimal selections of the bin boundaries. By identifying the grey cells in the table, we are able to construct the optimal result histogram in Figure 2(a), i.e., $H^* = \{(1,3,1.33), (4,5,4.5), (6,7,1)\}$

B. Differential Privacy

Given a count sequence $D = \{x_1, x_2, \ldots, x_n\}$, we say that a sequence D' is a neighbor sequence to D if and only if D' differs from D in only one count, with difference on the count at exactly 1. That is, for some integer $1 \le m \le n$, we have $D' = \{x_1, x_2, \ldots, x_{m-1}, x_m \pm 1, x_{m+1}, \ldots, x_n\}$.

A database query processing mechanism ensures ϵ-differential privacy, if it provides randomized answer H for any query Q, such that

$$\forall D, D', Q, H: \quad \Pr(Q(D) = H) \le \exp(\epsilon) \Pr(Q(D') = H),$$

where D and D' denotes two neighbor sequences, and $Q(D) = H$ denotes the randomized answer H with respect to query Q when the input data set is D. In our problem setting, the randomized answer for Q is the output of the randomized statistical report with noise on all the counts, e.g. the age statistics of AIDS patients. The standard solution mechanism to fulfill differential privacy [3] utilizes the concept of *query sensitivity*. In particular, the sensitivity of the query is defined as the maximal L_1-norm distance between the exact answers of the query set Q on any neighbor databases D and D', i.e.,

$$\Delta = \max_{D,D'} \|Q(D) - Q(D')\|_1$$

Dwork et al. [3] prove that differential privacy can be achieved by adding random noise to the answers of Q, such that the noise follows a zero-mean Laplace distribution with noise scale $b = \frac{\Delta}{\epsilon}$. The Laplace distribution with magnitude b, i.e., $Lap(b)$, follows the probability density function as $\Pr(x = y) = \frac{1}{2b} \exp\left(-\frac{|y|}{b}\right)$. For example, if the original statistical report is randomized with $Lap(b)$, the output report follows the distribution below:

$$\Pr(Q(D) = (x'_1, x'_2, \ldots, x'_n)) = \prod_{i=1}^{n} \frac{1}{2b} \exp\left(-\frac{|x'_i - x_i|}{b}\right)$$

It is tempting to apply the following simple strategy to implement the differential privacy protection on histograms. Given the count sequence D, for example, the system constructs the optimal histogram $H^* = \{B_1, \ldots, B_k\}$ using the solution in [10]. Random noise is then added on every representative count c_i under distribution $Lap(b)$ with $b = \frac{1}{\epsilon \min_j(r_j - l_j + 1)}$. This solution fails to satisfy the requirement of ϵ-differential privacy, since the structure of the result histogram is already leaking the privacies of the records.

In the following, we present an example on how the adversary infers unknown AIDS patients, when the system employs the scheme above. Assume that the adversary knows all the AIDS patients included in the statistical report, except *Alice*, in Fig. 1(b). If *Alice* is not a AIDS patient, there are only two patients between 40 and 45 years old. In that case, the optimal histogram with 3 bins is as shown in Figure 4, which is slightly different from the histogram in Figure 2(a). In the new optimal histogram, the first bin consists of 4 counts instead of 3 counts. Even when the randomized counts of the bins are not released, the adversary knows *Alice must be an AIDS patient*, purely based on the difference of the histogram structure. The example above shows that such a simple scheme fails to secure the privacy of personal records.

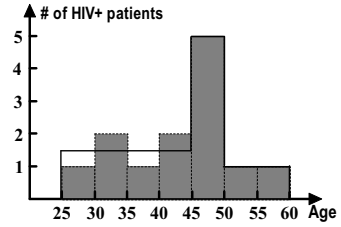


Fig. 4. Optimal histogram when *Alice* is excluded

III. NoiseFirst

Our first solution to the problem of histogram construction under differential privacy is done in two steps. In the first step, the algorithm follows the standard solution to differential privacy [3] by adding Laplace noise to each x_i in the original count sequence, outputting a randomized sequence

$\hat{D} = \{\hat{x}_1, \ldots, \hat{x}_n\}$. In the second step, the system runs the dynamic programming histogram construction algorithm on the dirty data $\hat{D}$. We call this method *NoiseFirst* , because the noise is added before the beginning of dynamic programming. Similar to the analysis in the standard solution proposed in [3], the sensitivity of NoiseFirst is exactly 1. Therefore, Laplace noise with magnitude $b = \frac{1}{\epsilon}$ is sufficient to guarantee ϵ-differential privacy on the resulting histogram. The complete pseudocode of the method is listed in Algorithm 1. The algorithm can be interpreted as a post-optimization technique to the standard solution, by merging adjacent noisy counts. Intuitively, averaging over the neighboring noisy counts is able to eliminate the impact of zero-mean Laplace noise, based on the large number theorem [12].

In the rest of the section, we provide some theoretical analysis on the expected error incurred by NoiseFirst. The major challenge in the analysis is deriving the connection between (i) the error of the histogram on the noisy count sequence $\hat{D}$ and (ii) the error of the same histogram on the original database D. As a first step, we will analyze the impact of merging consecutive noisy counts into a single bin. The following two lemmas quantify the expected errors of a bin on the noisy sequence and original sequence, respectively.

Lemma 1. *Given subsequence of counts $\{x_l, x_{l+1}, \ldots, x_r\}$. Let $\{\hat{x}_l, \ldots, \hat{x}_r\}$ be the noisy counts after the first step in Algorithm 1. If (l, r, c) is the result bin by merging all the counts, the expected squared error of the bin with respect to $\{\hat{x}_l, \ldots, \hat{x}_r\}$ is*

$$\mathbb{E}\left(Error((l,r,c), \{\hat{x}_l, \ldots, \hat{x}_r\}\right) = SSE(D,l,r) + \frac{2(r-l)}{\epsilon^2}$$

Proof. Assume that X_i is the variable of the count x_i after adding noise following $Lap(\frac{1}{\epsilon})$. Let C be the variable of the average count c, i.e. $C = \frac{\sum_{i=l} X_i}{r-l+1}$. We use δ_i to denote the residual variable on X_i, i.e. $\delta_i = X_i - x_i$, and let the size of the bin $s = r - l + 1$. The expected error of the result bin (l, r, c) on the noisy count sequence $\{\hat{x}_l, \ldots, \hat{x}_r\}$ is thus derived as follows.

$$\begin{aligned}
& \mathbb{E}\left\{\sum_{i=l}^{r}(X_i - C)^2\right\} \\
= \; & \mathbb{E}\left\{\sum_{i=l}^{r}(X_i)^2 - \frac{1}{s}\left(\sum_{i=1}^{r} X_i\right)^2\right\} \\
= \; & \mathbb{E}\left\{\sum_{i=l}^{r}(x_i)^2 + \sum_{i=l}^{r}(\delta_i)^2 - \frac{1}{s}\left(\sum_{i=l}^{r} x_i\right)^2 - \frac{1}{s}\left(\sum_{i=l}^{r}\delta_i\right)^2\right\} \\
= \; & \sum_{i=l}^{r}(x_i)^2 - \frac{1}{s}\left(\sum_{i=l}^{r} x_i\right)^2 + \mathbb{E}\left\{\sum_{i=l}^{r}(\delta_i)^2 - \frac{1}{s}\left(\sum_{i=l}^{r}\delta_i\right)^2\right\} \\
= \; & SSE(D,l,r) + \mathbb{E}\left\{\sum_{i=l}^{r}(\delta_i)^2 - \frac{1}{s}\sum_{i=l}^{r}(\delta_i)^2\right\} \\
= \; & SSE(D,l,r) + \frac{2(s-1)}{\epsilon^2}
\end{aligned}$$

Based on our definition $s = r - l + 1$, it directly leads to the conclusion in the lemma. □

Note that the error above is the sum of squared error of the average count on the noisy data in $\hat{D}$. The following lemma shows how to estimate the error of a bin with respect to the original counts in D.

Algorithm 1 NoiseFirst (count sequence D, bin number k, privacy guarantee ϵ)

1: Generate a new database $\hat{D}$ by adding independent $Lap(\frac{1}{\epsilon})$ on every count $x_i \in D$.
2: Build the optimal histogram on $\hat{D}$ with dynamic programming method.
3: Return histogram $\hat{H} = \{(l_1, r_1, c_1), \ldots, (l_k, r_k, c_k)\}$, in which every c_j is the mean of the noisy data $\{\hat{x}_{l_j}, \ldots, \hat{x}_{r_j}\}$.

Lemma 2. *Given subsequence of counts $\{x_l, x_{l+1}, \ldots, x_r\}$. Let $\{\hat{x}_l, \ldots, \hat{x}_r\}$ be the noisy counts after the first step in Algorithm 1. If (l, r, c) is the result bin by merging all the counts, the expected squared error of the bin with respect to $\{x_l, \ldots, x_r\}$ is*

$$\mathbb{E}\left(Error(\{(l,r,c\}, \{x_l, \ldots, x_r\})\right) = SSE(D,l,r) + \frac{2}{\epsilon^2}$$

Proof. Similar to the proof of Lemma 1, we derive the error as follows. Again, we use the notations X_i, C and s, as they are defined in the proof for Lemma 1.

$$\begin{aligned}
& \mathbb{E}\left\{\sum_{i=l}^{r}(x_i - C)^2\right\} \\
= \; & \sum_{i=l}^{r}(x_i)^2 - \frac{2}{s}\left(\sum_{i=l}^{r} x_i\right)^2 + \frac{1}{s}\mathbb{E}\left\{\left(\sum_{i=l}^{r} X_i\right)^2\right\} \\
= \; & SSE(D,l,r) - \frac{1}{s}\left(\sum_{i=l}^{r} x_i\right)^2 + \frac{1}{s}\mathbb{E}\left\{\left(\sum_{i=l}^{r}(x_i + \delta_i)\right)^2\right\} \\
= \; & SSE(D,l,r) + \frac{1}{s}\mathbb{E}\left\{\sum_{i=l}^{r}(\delta_i)^2\right\} \\
= \; & SSE(D,l,r) + \frac{2}{\epsilon^2}
\end{aligned}$$

This completes the proof. □

Without using the histogram technique, the expected error of the noisy counts $\{\hat{x}_l, \ldots, \hat{x}_r\}$ with respect to the original data $\{x_l, \ldots, x_r\}$ is exactly $\frac{2(r-l+1)}{\epsilon^2}$, i.e.,

$$\begin{aligned}
& \mathbb{E}\left(Error(\{(l,l,\hat{x}_l), \ldots, (r,r,\hat{x}_r)\}, \{x_l, \ldots, x_r\})\right) \\
= \; & \frac{2(r-l+1)}{\epsilon^2}
\end{aligned}$$

Lemma 2 shows that the expected error of histogram with a single bin is $SSE(D,l,r) + \frac{2}{\epsilon^2}$. This implies that merging all counts into a single bin is an attractive option, when ϵ is small, i.e., $\epsilon < \sqrt{\frac{2(r-l)}{SSE(D,l,r)}}$. In other words, the performance of a one-bin histogram is superior than the standard solution in [3], given a small enough ϵ. The following theorem is the straightforward extension of Lemma 1 and Lemma 2, extending the analysis from one bin to multiple bins.

In the following analysis, we use H to denote the histogram with the same structure as $\hat{H}$ but with different average counts in the bins. Instead of using noisy counts $\{\hat{x}_{l_j}, \ldots, \hat{x}_{r_j}\}$, H calculates the average count for bin B_j with the original counts $(x_{l_j} + \ldots + x_{r_j})/(r_j - l_j + 1)$. It is easy to see that $Error(H, D) = \sum_j SSE(D, l_j, r_j)$, which helps proving the following theorem by simple extension from Lemma 1 and Lemma 2.

Theorem 1. *Given H and $\hat{H}$ as defined above, the expected error of the histogram on $\hat{D}$ and D are respectively,*

$$\begin{aligned}\mathbb{E}(Error(\hat{H}, \hat{D})) &= Error(H, D) + \frac{2(n-k)}{\epsilon^2} \\ \mathbb{E}(Error(\hat{H}, D)) &= Error(H, D) + \frac{2k}{\epsilon^2}\end{aligned}$$

Since k is fixed before running the algorithm, the theorem above implies that optimizing the histogram by minimizing on $Error(\hat{H}, \hat{D})$ leads to the solution that minimizes $Error(\hat{H}, D)$, i.e.,

$$\arg\min_{\hat{H}} \mathbb{E}(\hat{H}, \hat{D}) = \arg\min_{\hat{H}} \mathbb{E}(\hat{H}, D) \tag{2}$$

The equation provides the intuition behind the correctness of NoiseFirst method in Algorithm 1, which simply runs the dynamic programming on the noisy data sequence $\hat{D}$.

While the analysis above assumes that k is specified by the user, it is actually free for the algorithm to automatically select the optimal k. Since the dirty data already satisfies ϵ-different privacy, the algorithm is allowed to run n times, testing $k = 1, \ldots, n$ and returning the one with the minimal expected error on the original sequence D. Although it is impossible to directly evaluate the expected error on D, we can utilize Theorem 1 again. If $\hat{H}_k$ is the optimal histogram returned by Algorithm 1 with $k = 1, \ldots n$ bins, the final result histogram is selected based on the following optimization objective:

$$\hat{H}^* = \arg\min_{\hat{H}_k} \mathbb{E}\left(Error(\hat{H}_k, \hat{D}) + \frac{2n - 4k}{\epsilon^2}\right) \tag{3}$$

On the other hand, NoiseFirst method adds $Lap(\frac{1}{\epsilon})$ on the counts, assuming that the sensitivity Δ equals 1. As shown in our example in Section I, however, the sensitivity of the statistical report can be much reduced, if we are allowed to add noise after the histogram construction. In next section, we implement this idea by proposing a new algorithm that (i) does not inject noise on the data before starting the dynamic programming, but (ii) randomizes the dynamic programming results.

IV. StructureFirst

The NoiseFirst method preserves ϵ-differential privacy and reduces the expected error by merging bins on the noisy data instead of original data. However, it fails to exploit the reduced sensitivity after bin merging. In this section, we discuss another approach that adopts a completely different strategy, which calculates the histogram structure on the original data before adding the noise on the bin structure. We call this method *StructureFirst*. In the rest of the section, we use D_i to denote the partial database $D_i = \{x_1, x_2, \ldots, x_i\}$, and $H(D_i, j)$ to denote the optimal histogram with j bins on D_i.

Algorithm 2 StructureFirst (count sequence D, bin number k, privacy parameter ϵ, count upper bound F)

1: Partition ϵ into two parts ϵ_1 and ϵ_2 that $\epsilon_1 + \epsilon_2 = \epsilon$.
2: Build the optimal histogram H^* and keep all intermediate results, i.e. $Error(H(D_i, j), D_i)$ for all $1 \le i \le n$ and $1 \le j \le k$
3: Set right boundary of kth bin at $r_k = n$
4: **for** each j from $k-1$ down to 1 **do**
5: **for** each q from j up to $r_{j+1} - 1$ **do**
6: Calculate $E(q, j, r_{j+1}) = Error(H(D_q, j), D_q) + SSE(D, q+1, r_{j+1})$
7: **end for**
8: select $r_j = q$ from $k - 1 \le q < r_{j+1}$ with probability proportional to $\exp\left\{-\frac{\epsilon_1 E(q,j,r_{j+1})}{2k(2F+1)}\right\}$
9: Set $l_{j+1} = r_j + 1$
10: **end for**
11: Calculate average count c_j for every bin on interval $[l_j, r_j]$.
12: Add noise to counts as $\hat{c}_j = c_j + Lap\left(\frac{1}{\epsilon_2(r_j - l_j + 1)}\right)$
13: Return histogram $\{(l_1, r_1, \hat{c}_1), \ldots, (l_k, r_k, \hat{c}_k)\}$

In Algorithm 2, we present the details of our StructureFirst method. The algorithm first constructs the optimal histogram H^* on the basis of the original data D. All the intermediate results are stored in a table, as is done in Figure 3. To add noise to the structure of the histogram, the algorithm randomly moves the boundaries between the bins. When choosing the boundary between B_j and B_{j+1}, it picks up r_j for bin B_j at the count $x_q \in D$ with probability proportional to

$$\exp\left\{-\frac{\epsilon_1 E(q, j, r_{j+1})}{2(k-1)(2F+1)}\right\}$$

Here, $E(q, j, r_{j+1})$ is the error of the histogram, which consists of the optimal histogram with j bins covering $[1, q]$ and the $j+1$th bin covering $[q+1, r_{j+1}]$. If $(q+1, r_{j+1}, c)$ is a new bin constructed by merging counts $\{x_{q+1}, \ldots, x_{r_{j+1}}\}$ and c is the average count, it is straightforward to verify that

$$\begin{aligned}& E(q, j, r_{j+1}) \\ =\ & Error(H(D_q, j) \cup \{(q+1, r_{j+1}, c)\}, D_{r_{j+1}}) \\ =\ & Error(H(D_q, j), D_q) + SSE(D, q+1, r_{j+1})\end{aligned} \tag{4}$$

In the probability function, F is some prior knowledge on the upper bound on the maximal count in the sequence, which is assumed to be independent of the database. After setting down all the boundaries, Laplace noise is added on the average counts. For bin (l_j, r_j, c_j), the magnitude of the Laplace noise on c_j is $\frac{1}{\epsilon_2(r_j - l_j + 1)}$. The complete pseudocode of the algorithm is listed in Algorithm 2.

In the rest of the section, we will analyze the correctness and utility of the algorithm, and discuss how the values of

ϵ_1 and ϵ_2 should be decided. Different from the NoiseFirst method, StructureFirst does not support the optimization on the bin number k. The user must specify the desired k before the beginning of the algorithm. In Section VI, we provide some empirical studies on how to pick up generally good k value on real data sets.

A. Correctness

To pave the way for the correctness proof, we first derive some worst case sensitivity analysis on the intermediate results in dynamic programming, when the count sequence changes by at most 1 on some count $x_i \in D$.

Lemma 3. *Let F be an upper bound of the maximum count in any of the bins. Given two neighbor database D and D', the error of the optimal histograms on D and D' with respect to the first i counts changes by at most*

$$|Error(H(D'_i, j), D'_i) - Error(H(D_i, j), D_i)| \leq 2F + 1$$

Proof: Given two neighbor databases as D and D', there is one and only one count x_m changes by 1, i.e., $x_m - 1 \leq x'_m \leq x_m + 1$. Assume that x_m is in bin (l_z, r_z, c_z) in $H(D_i, j)$. Since $H(D'_i, j)$ is the optimal histogram for D'_i with j bins, it is straightforward to know that given any histogram H' covering interval $[1, i]$, we have

$$Error(H(D'_i, j), D'_i) \leq Error(H', D'_i) \tag{5}$$

In particular, we construct a special histogram H' by reusing all bins in $H(D_i, j)$, except that $c'_z = \frac{\sum_{q=l_z}^{r_z} x_q + (x'_m - x_m)}{r_z - l_z + 1}$ replaces $c_z = \frac{\sum_{q=l_z}^{r_z} x_q}{r_z - l_z + 1}$. Let $s = r_z - l_z + 1$. In the following, we derive some upper bound on the error of such H' on D'_i.

$$\begin{aligned} & Error(H', D'_i) - Error(H(D_i, j), D_i) \\ = \; & (x'_m)^2 - (x_m)^2 - s(c'_z)^2 + s(c_z)^2 \end{aligned} \tag{6}$$

When $x'_m = x_m + 1$, the difference on the error above is

$$\begin{aligned} & Error(H', D'_i) - Error(H(D_i, j), D_i) \\ = \; & (x_m + 1)^2 - (x_m)^2 - s\left(c_z + \frac{1}{s}\right)^2 + s(c_z)^2 \\ = \; & 2x_m + 1 - 2c_z - \frac{1}{s} \\ \leq \; & 2x_m + 1 \end{aligned} \tag{7}$$

When $x'_m = x_m - 1$, the difference can be estimated similarly as

$$\begin{aligned} & Error(H', D'_i) - Error(H(D_i, j), D_i) \\ = \; & (x_m - 1)^2 - (x_m)^2 - s\left(c_z - \frac{1}{s}\right)^2 + s(c_z)^2 \\ = \; & -2x_m + 1 + 2c_z - \frac{1}{s} \\ \leq \; & 2c_z + 1 \\ \leq \; & 2 \max_{l_s \leq q \leq r_s} x_q + 1 \end{aligned} \tag{8}$$

Combining Equation 7 and Equation 8, and given that F is the upper bound of maximal count in the database, we have

$$Error(H(D'_i, j), D'_i) \leq Error(H', D'_i) \leq Error(H, D_i) + 2F + 1$$

This reaches the conclusion of the lemma. □

Provided with the worst case analysis on the change of error, we can prove the validity of differential privacy on the re-selection of bin boundary in our algorithm.

Lemma 4. *The selection of r_j on line 8 of Algorithm 2 satisfies $\frac{\epsilon_1}{k-1}$-differential privacy.*

Proof. Let $E(q, j, r_{j+1})$ be the cost of setting $r_j = q$, we use $E'(q, j, r_{j+1})$ to denote the cost when the same histogram is constructed on D' instead. Based on Lemma 3, we have

$$|E'(q, j, r_{j+1}) - E(q, j, r_{j+1})| \leq 2F + 1$$

Based on pseudocodes in the algorithm, when r_{j+1} is fixed, the probability of selecting $r_j = q$ on D' is

$$\begin{aligned} \Pr(r_j = q) &= \frac{\exp\left\{-\frac{\epsilon_1 E'(q,j,r_{j+1})}{2(k-1)(2F+1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1 E'(z,j,r_{j+1})}{2(k-1)(2F+1)}\right\}} \\ &\leq \frac{\exp\left\{-\frac{\epsilon_1 (E(q,j,r_{j+1}) - 2F - 1)}{2(k-1)(2F+1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1 (E(z,j,r_{j+1}) + 2F + 1)}{2(k-1)(2F+1)}\right\}} \\ &= \exp\left(\frac{\epsilon_1}{k-1}\right) \frac{\exp\left\{-\frac{\epsilon_1 E(q,j,r_{j+1})}{2(k-1)(2F+1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1 E(z,j,r_{j+1})}{2(k-1)(2F+1)}\right\}} \end{aligned}$$

Hence, the selection of r_j follows $(\frac{\epsilon_1}{k-1})$-differential privacy. □

Given the results in previous lemmas, we are able to prove the correctness of Algorithm 2 by the following theorem.

Theorem 2. *Algorithm 2 satisfies ϵ-differential privacy.*

Proof. In Algorithm 2, the output histogram relies on the results on line 8 and line 12, which are all independent of each other. Line 8 is run k times and line 12 is run exactly once. According to Lemma 4, each execution on line 8 costs the privacy budget $\frac{\epsilon_1}{k}$. Based on the concept of generalized sensitivity used in [8], the privacy cost of line 12 is ϵ_2. It is because we can define $W_j = (r_j - l_j + 1)$ to meet the requirement of generalized sensitivity. Consider two databases D and D' with the same boundary structure calculated in the StructureFirst algorithm, we have $\hat{c}_j$ and $\hat{c}'_j$ being the noisy average counts. They satisfy the following function in form of weighted sensitivity.

$$\sum_j W_j \cdot \hat{c}_j - \sum_j W_j \cdot \hat{c}'_j \leq 1$$

The total privacy budget spent in the algorithm is thus $k\frac{\epsilon_1}{k} + \epsilon_2 = \epsilon$, since $\epsilon_1 + \epsilon_2 = \epsilon$. □

B. Cost Analysis

The next question is: how much error is incurred by applying Algorithm 2 on any count sequence D? In this part

of the section, we answer this question with analytic analysis on the expectation case.

Lemma 5. *Given any non-negative real number x and positive constant c, we have $x \cdot e^{-cx} \leq \frac{1}{2c}$.*

Lemma 6. *The error of the histogram increments by no more than $\frac{8(k-1)(2F+1)^2}{\epsilon_1(8(k-1)(2F+1)-\epsilon_1 n^2F^2)}$, every time we move the boundary by line 8 in Algorithm 2.*

Proof. Assume that the optimal histogram with $j+1$ bins is supposed to selection q^* as the boundary between jth bin and $j+1$th bin, to minimize the error $Error(H(D_q, j), D_q) + SSE(q+1, r_{j+1})$. We are interested in the expected increase on the error when the randomized algorithm fails to select q^*. Note that the probabilities remain the same if we reduce each $E(q, j, r_{j+1})$ by $E(q^*, j, r_{j+1})$. Since each $E(q) - E(q^*) \geq 0$ and there is at least one $z = q^*$ that $E(z, j, r_{j+1}) - E(q^*, j, r_{j+1}) = 0$, the following inequalities must be valid, using the well known fact $e^{-x} \geq 1 - x$ for any positive x.

$$\begin{aligned} & \sum_z \exp\left\{-\frac{\epsilon_1(E(z,j,r_{j+1}) - E(q^*,j,r_{j+1}))}{2(k-1)(2F+1)}\right\} \\ \geq & \sum_z \left(1 - \frac{\epsilon_1(E(z,j,r_{j+1}) - E(q^*,j,r_{j+1}))}{2(k-1)(2F+1)}\right) \\ \geq & \; n\left(1 - \frac{\epsilon_1 nF^2}{8(k-1)(2F+1)}\right) \end{aligned} \tag{9}$$

To simplify the notation, we use $E(q)$ to replace $E(q, j, r_{j+1})$, when j and r_{j+1} are clear in the context. Thus, the expectation on the additional error over the optimal one with q^* is

$$\begin{aligned} & \sum_q \{(E(q)) \Pr(r_j = q)\} - E(q^*, j, r_{j+1}) \\ = & \sum_q \{(E(q) - E(q^*)) \Pr(r_j = q)\} \\ = & \sum_q \left\{(E(q) - E(q^*)) \frac{\exp\left\{-\frac{\epsilon_1(E(q)-E(q^*)}{2(k-1)(2F+1)}\right\}}{\sum_z \exp\left\{-\frac{\epsilon_1(E(z)-E(q^*)}{2(k-1)(2F+1)}\right\}}\right\} \\ \leq & \sum_q \left\{(E(q) - E(q^*)) \frac{\exp\left\{-\frac{\epsilon_1(E(q)-E(q^*)}{2(k-1)(2F+1)}\right\}}{n - \frac{\epsilon_1 n^2F^2}{8(k-1)(2F+1)}}\right\} \\ \leq & \sum_q \frac{(k-1)(2F+1)}{\epsilon_1}\left(n - \frac{\epsilon_1 n^2F^2}{8(k-1)(2F+1)}\right)^{-1} \\ \leq & \frac{8(k-1)(2F+1)^2}{\epsilon_1(8(k-1)(2F+1) - \epsilon_1 nF^2)} \end{aligned} \tag{10}$$

Here, the first inequality is due to Equation 9. The second inequality applies Lemma 5. The final inequality is because the number of candidate q is no larger than the size of the database. □

Note that the cost analysis in Lemma 6 does not rely on j or r_{j+1}. Therefore, the total incremental error over the optimal histogram without noise is simply $k-1$ times of the error in Lemma 6. Combined with the error introduced on line 12 in Algorithm 2, the total expected error is upper bounded by the following theorem.

Theorem 3. *The output histogram of Algorithm 2, the expected error is at most*

$$Error(H(D,k),D) + \frac{8(k-1)^2(2F+1)^2}{\epsilon_1(8(k-1)(2F+1) - \epsilon_1 nF^2)} + \frac{2k}{(\epsilon_2)^2}$$

The error analysis in the theorem above shows the superiority of StructureFirst method. We can interpret the expected error of StructureFirst as $Error(H(D,k),D) + O(\frac{k}{\epsilon^2})$, by ignoring all constant items. This is a significant theoretical improvement over NoiseFirst method.

C. Budget Assignment

When k is fixed, the expected error of the result histogram depends on the assignment of ϵ_1 and ϵ_2, since $\epsilon_1 + \epsilon_2 = \epsilon$. In the error analysis of Theorem 3, the total error consists of three parts. The first part $Error(H(D,k),D)$ relies on k and the original data D. The other two parts are independent of the data, but are decided by n, k, F, ϵ_1 and ϵ_2. Therefore, it is possible to minimize the expected error by finding the optimal combination of ϵ_1 and ϵ_2:

$$\begin{aligned} \text{Minimize} \quad & \left(\frac{8(k-1)^2(2F+1)^2}{\epsilon_1(8(k-1)(2F+1) - \epsilon_1 nF^2)} + \frac{2k}{(\epsilon_2)^2}\right) \\ \text{s.t.} \quad & \epsilon_1 + \epsilon_2 = \epsilon \end{aligned}$$

Although the optimization above does not always has a closed-form solution, we can employ numerical methods to identify a near-optimal assignment on ϵ_1 and ϵ_2. Specifically, we can apply a simple sampling technique that tries different $0 < \epsilon_1 < \epsilon$ and returns the best ϵ_1 and $\epsilon_2 = \epsilon - \epsilon_1$ encountered. Since it takes constant time to verify the cost when a specific pair of ϵ_1 and ϵ_2 are given, the computational overhead of the budget assignment optimization is insignificant.

V. Answering Arbitrary Range Counts

A histogram is constructed on the basis of the *Sum of Squared Error* (SSE), which aims to minimize the average error of queries with unit length in the numeric domain. When the user is querying on the sum of consecutive counts, the histogram synopsis may not be optimal in terms of query accuracy. To understand the underlying reason behind the ineffectiveness of histogram synopsis on large range queries, let us revisit the example in Figure 2(a). If the user queries for patients with age between 25 and 35, the average count of the first bin is not sufficient to return accurate result, even when there is no Laplace noise added on the histogram. Therefore, in some cases, it renders more accurate results if some bin adopts a non-uniform scheme of count computation.

In this section, we discuss how to implement such non-uniform scheme on both NoiseFirst and StructureFirst methods respectively. Two different strategies are employed in NoiseFirst and StructureFirst methods because of the different properties of the algorithms.

A. NoiseFirst Method

In Algorithm 1, after finding the structure of the histogram, NoiseFirst method always uses the average count to replace all the original noisy counts. Another option is to keep using the dirty data instead of the average counts in some of the bins. To ensure that a better selection is made, we propose to compare the estimations on the errors of the two options and adopt the one with smaller expected error. In particular, for every bin B_i, we assume that $[l, r]$ is the interval B_i covers in the data domain, and c is the average count over the noisy counts. NoiseFirst uses the average count for B_i only when

$$SSE(\hat{D}, l, r) < \frac{4(r-l)}{\epsilon^2}$$

Otherwise, the dirty counts $\{\hat{x}_l, \ldots, \hat{x}_r\}$ are kept as the original values. Intuitively, when the structural error of the histogram is small, our scheme prefers to use the average counts in the bin. It is because the average count is capable of eliminating noise on consecutive dirty counts. When the original counts are very different from each other in the bin, averaging over all counts does not help reduce the errors. In such situations, the dirty counts may perform better.

B. StructureFirst Method

Unfortunately, the non-uniform strategy used for NoiseFirst method is not applicable in StructureFirst method, since StructureFirst calculates the histogram on the original counts. To apply non-uniform scheme, we mainly borrow the idea from [6]. Generally speaking, after constructing all the bins, StructureFirst runs the boosting algorithm [6] on every bin, with differential privacy parameter ϵ_2.

Given a bin B_i covering $[l, r]$, Laplace noise with magnitude $Lap\left(\frac{2}{\epsilon_2}\right)$ is added on every count x_j for $l \leq j \leq r$, as well as the sum $s_i = \sum_{l \leq j \leq r} x_j$. Assume $\overline{x}_j$ is the result noisy count and $\overline{s}_i$ is the noisy sum. Our algorithm runs normalization on $\{\overline{x}_l, \ldots, \overline{x}_r, \overline{s}_i\}$ and returns a new group of counts $\{x'_l, \ldots, x'_r, s'_i\}$ satisfying that $s'_i = \sum_{l \leq j \leq r} x'_j$. These counts are used to approximate the original counts in the database. By [6], this scheme is always consistent with ϵ_2-differential privacy. Therefore, the complete modified algorithm of StructureFirst is still fulfilling ϵ-different privacy.

There is some connection between such method and [6]. Our StructureFirst method with this non-uniform count assignment scheme can be considered as an adaptive version of the boosting algorithm in [6]. In the original boosting algorithm, the user must specify the fan-out of the tree structure before running the algorithm. This fan-out decides how the algorithm partitions the count sequence and how the normalization is run bottom-up and top-down. In our histogram technique, an adaptive partitioning is used instead, which is supposed to better capture the distribution of the data. This is the underlying reason that our StructureFirst method outperforms the original boosting algorithm. In the following section, some empirical studies will verify the advantages of our proposal.

VI. Experiments

This section experimentally compares the effectiveness of the proposed solutions NoiseFirst and StructureFirst for range-count queries with three state-of-the-art methods, referred to as *Dwork* [3], *Privelet* [8] and *Boost* [6]. Specifically, our implementation of NoiseFirst uses the noisy histograms produced by Dwork as inputs; in StructureFirst, the parameter F i.e., maximal possible count in a histogram bin, is fixed to $\frac{10^4 * NumOfRecords}{DomainSize}$. Meanwhile, the implementation of Boost follows the same settings in [6], and employs binary trees as the synopsis structure as in [6]. Similarly, StructureFirst also uses a binary tree inside each bin of the histogram, in order to ensure fairness in our comparisons. Akin to [6], we evaluate the accuracy of the algorithms over range queries with varying length and location. In particular, given a query length L, we test all its possible range queries and report the average *square error*. To evaluate the effectiveness under different privacy requirements, we test all methods with three popular values of ϵ: 0.01, 0.1 and 1. All methods are implemented in C++ and tested on an Intel Core2 Duo 2.33 GHz CPU with 2GB RAM running Windows XP. In each experiment, every algorithm is executed 10 times, and its average accuracy are reported.

The experiments involve four data sets: (i) *Age* contains $100,078,675$ records, each of which corresponds to the age of an individual, extracted from the IPUM's census data of Brazil. The ages range from 0 to 101. Differential privacy guarantees the hardness for adversaries to infer any individual's true age from published statistics. (ii) *Search Logs* [6] is a synthetic data set generated by interpolating *Google Trends* data and *America Online search logs*. It contains $32,768$ records, each of which stores the frequency of searches (ranging from 1 to 496) with the keyword "Obama" within a 90 minutes interval, from Jan. 1, 2004 to Aug. 9, 2009. DP ensures that the adversary cannot derive if any specific person has searched the keyword "Obama" at a particular time. (iii) *NetTrace* [6] contains the IP-level network trace at a border gateway in a major university. Each record reports the number of external hosts connected to an internal host. There are $65,536$ records with connection number ranging from 1 to 1423. The application of DP protects the information of individual connections. Finally, (iv) *Social Network* [6] records the friendship relations among 11K students from the same institution, derived from an online social network website. Each record contains the number of friends of certain student. There are $32,768$ students, each of which has at most 1678 friends. DP protects the sensitive information of individuals' connections from adversaries.

To generate histograms, we transform all four data sets to statistical counts on every possible value. On *Age*, for example, each x_i indicates the number of individuals aged i ($1 \leq i \leq 101$). We observed that the distribution of *Age* is very different from the other three, in that the counts therein are more evenly distributed over the entire domain, while all the other data sets exhibit a high degree of skewness.

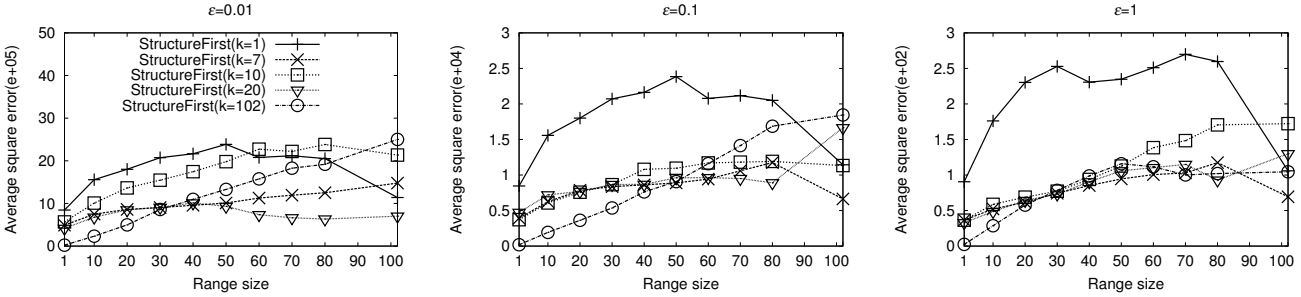


Fig. 5. Varying k on *Age*

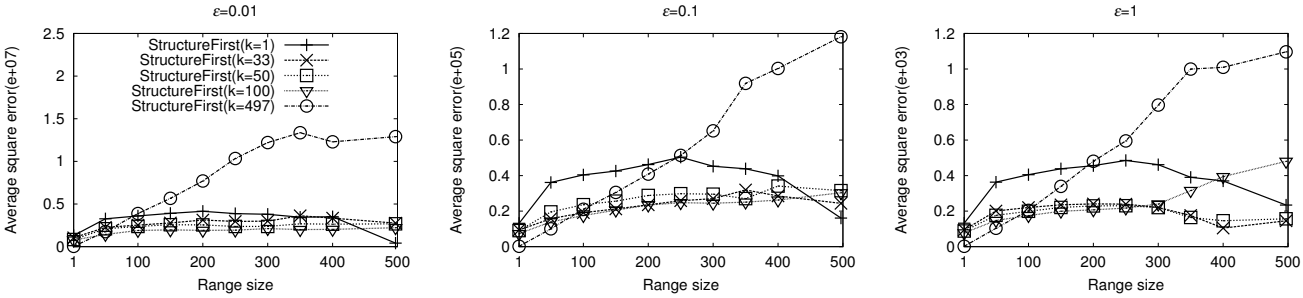


Fig. 6. Varying k on *Search Logs*

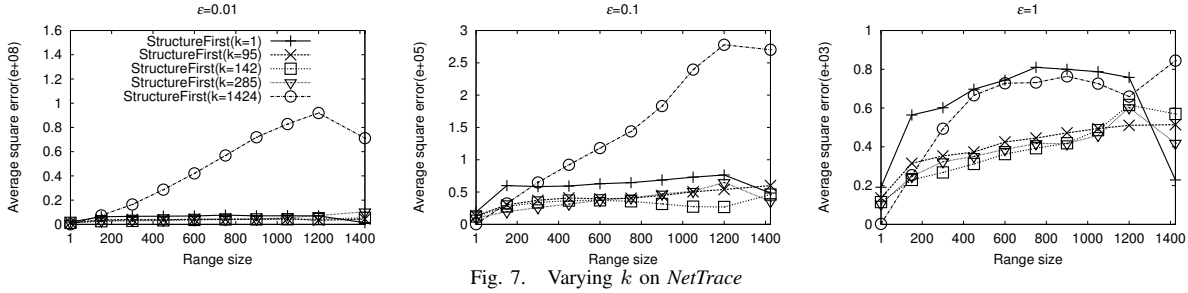


Fig. 7. Varying k on *NetTrace*

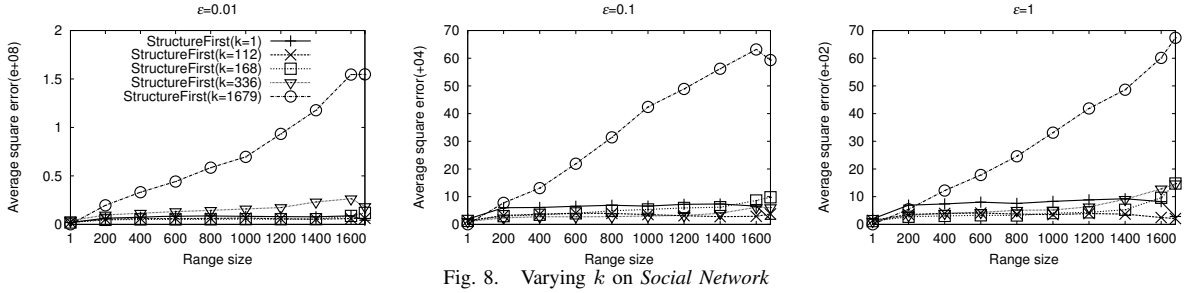


Fig. 8. Varying k on *Social Network*

A. Impact of Parameter k on StructureFirst

This subsection evaluates the effectiveness of StructureFirst with varying the number of bins k in the resulting histogram. Specifically, for each data set with domain size n, we test five different values of k: $\{1, \frac{n}{15}, \frac{n}{10}, \frac{n}{5}, n\}$. Figures 5 - 8 illustrate the average square error of StructureFirst with different values of k and different query range size on all the four data sets. There are several important observations we made from the results.

First, when k equals the domain size n, StructureFirst reduces to method Dwork [3] that adds random Laplace noise with magnitude $\frac{1}{\epsilon}$ to every count in the dataset. In this case, the average square error of StructureFirst increases linearly with the query range size, regardless of the settings on ϵ. Note that the histogram structure is fixed to one bin per count in the original statistics; hence, there is no need to spend privacy budget on the histogram structure.

Second, when $k = 1$, StructureFirst is equivalent to the Boost method. Similar to the case for $k = n$, the histogram structure is fixed, and the expenditure of privacy budget on the histogram is $\epsilon_1 = 0$. From the figures, we can see that StructureFirst's performance follows the properties of Boost, with more accurate query results on queries with the minimum and maximum lengths.

Third, when $1 < k < n$, StructureFirst achieves better performance compared to the cases with extreme values,

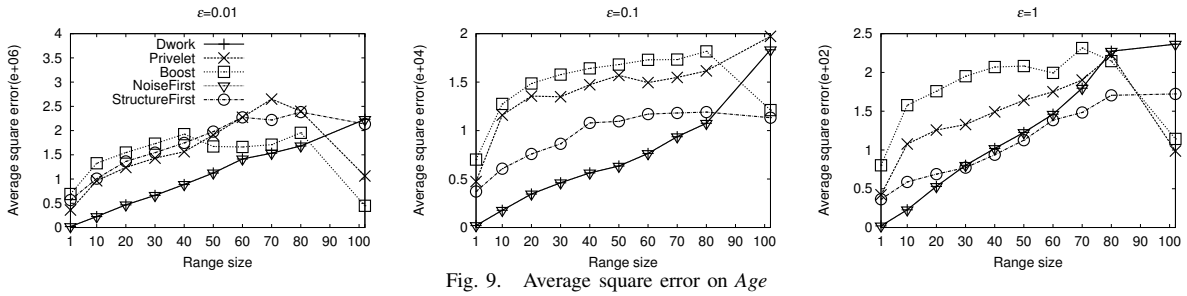


Fig. 9. Average square error on *Age*

especially on queries with intermediate length. This shows that our StructureFirst method successfully constructs histograms adaptively to the data distribution, reducing the error for queries covering different bins on the data domain.

Finally, on all data sets, StructureFirst achieves its highest accuracy when k is around $n/10$. In such cases, every bin consists of 10 counts by average, offering a balanced results for all types of queries with different length. Moreover, the performance of binary tree and the normalization technique from [6] also works well, since every bin only involves 10 random counts in the processing by average. Therefore, we employ $k = n/10$ in the rest of the paper.

B. Evaluation on all methods

We now evaluate the performance of all solutions for range count queries, starting with unit-length queries. The errors of such queries is equivalent to the *Sum of Squared Error* divided by the domain size n. The results are listed in Table I, with best performance on specific data set and ϵ value marked in bold. Clearly, NoiseFirst achieves best accuracy on *Search Log*, *NetTrace* and *Social Network* data sets. Dwork remains the best method on *Age* data set. The reasons for their relative performance are explained as follows. Since *Age* is more evenly distributed, it is more difficult for NoiseFirst to find a good histogram structure after Dwork adding random noise to the counts. On the other hand, because the last three data sets have a large number of small counts, NoiseFirst is more effective on merging consecutive small counts into bins and eliminating the impact of the random noise in the bins. StructureFirst does not show competitive results for queries with unit length, mainly because the extension with the binary tree normalization technique from [6]. The performance of StructureFirst on such queries is thus closer to that of Boost. In generally, the results imply that NoiseFirst is the better for queries with a short range. To the best of our knowledge, it is also the first method outperforming Dwork in such settings.

Next we evaluate the performance of all methods on queries with varying ranges, reporting their accuracy in terms of average square error. The results are shown in Figures 9-12. Note that StructureFirst sets $k = \frac{n}{10}$ in all settings, based on the conclusion drawn in Section VI-A. On all data sets, the average square error of NoiseFirst and Dwork increases linearly with the query range. Both of them significantly outperform the other three algorithms on small range sizes. As the query length grows, the performance gap between NoiseFirst and Dwork expands, though NoiseFirst is better by a small margin on most of the query ranges. These results again verify the advantage of NoiseFirst for short ranges, which is consistent with the results in Table I.

The accuracy of both Privelet and Boost is high for queries with very long or very short ranges; for other queries, they tend to incur rather high errors. This is due to the binary tree structures used in their algorithms (note that the Haar wavelet used in Privelet is essentially a binary tree). Hence, very short queries favor both methods, as they only need to access a small number of nodes close to the leaf level. Very long queries, on the other hand, are answered mainly using a small number of nodes close to the top of the tree, leading to high accuracy. When queries with length in neither extremes, these methods return poorer results, since a large number of nodes from different levels of the tree are involved in query processing.

StructureFirst produces more accurate estimates than both Privelet and Boost for most queries. The main reason are (i) that StructureFirst utilizes the count similarities amongst consecutive bins; (ii) that StructureFirst avoids building a large tree over the whole data domain; reducing the number of nodes required to answer a range query; and (iii) that bins in a domain partition tend to have similar counts, and, thus, building trees separately on each partition benefits from both the consistency and the similarities amongst bins. Another interesting observation concerning StructureFirst is that when the range size is as large as the domain size, Boost has better accuracy than StructureFirst. This is because Boost performs the consistency inference on the whole domain, and hence, it has a better overview on range counts on the complete domain.

C. Summary

Based on experimental results shown above, we conclude that NoiseFirst usually returns more accurate results for unit-length range count queries than all the state-of-the-art methods. For the error of unit-length queries has direct impacts on the histogram shape, therefore NoiseFirst provides us a possibility to make a better visualization of the published data. StructureFirst, on the other hand, has apparent superiority with most settings of the query lengths. Thus, a query executer can provide the users more precise results by querying the DP-complaint histogram published by the StructureFirst method.

VII. Related Work

Numerous techniques have been proposed for publishing various types of data while achieving differential privacy (see

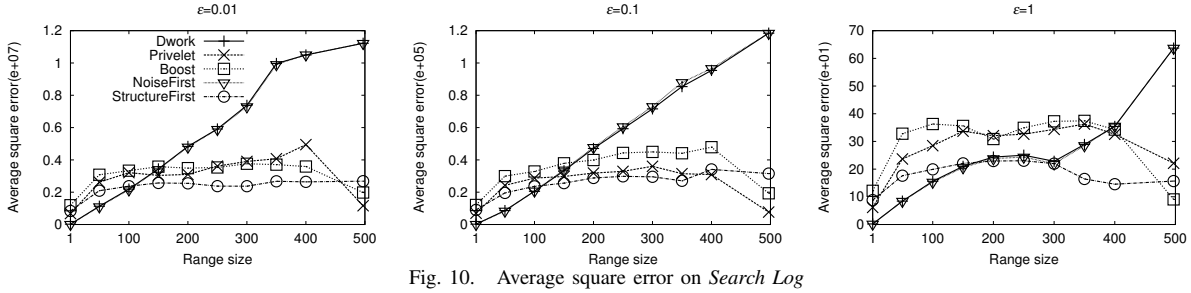


Fig. 10. Average square error on *Search Log*

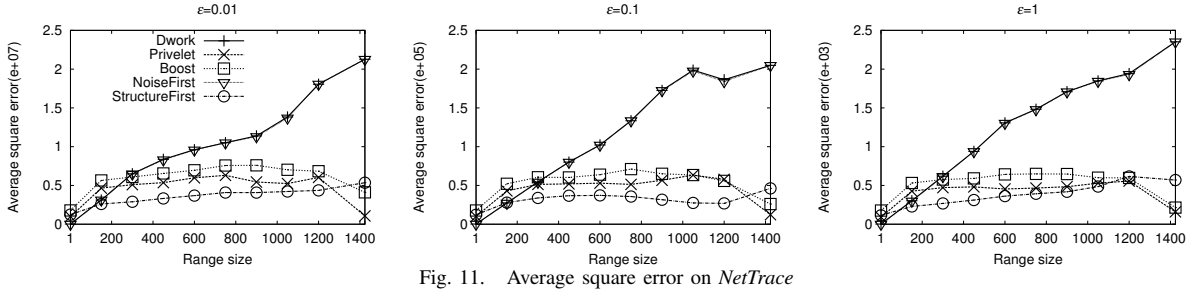


Fig. 11. Average square error on *NetTrace*

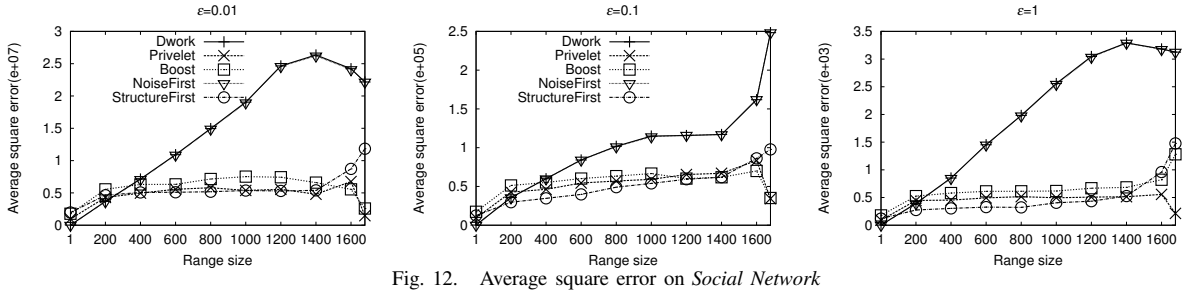


Fig. 12. Average square error on *Social Network*

Data set	Age			Search Logs			NetTrace			Social Network		
ϵ	0.01	0.1	1	0.01	0.1	1	0.01	0.1	1	0.01	0.1	1
Dwork	**19,574**	**201**	**2.11**	19,507	206	2.01	19,858	195	2.06	19,810	195	2.06
Privelet	360,774	4,745	42	678,763	6,975	62	978,884	9,987	91	972,103	9,426	92
Boost	691,220	7,008	80	1,191,886	121,15	122	1,776,939	17,704	176	1,781,207	17,300	178
NoiseFirst	20,637	203	**2.11**	**12,491**	**143**	**1.72**	**12,495**	**121**	**1.38**	**12,494**	**124**	**1.51**
StructureFirst	566,212	3715	36	839,892	9,051	88	1,240,337	12,192	115	1,970,914	12,430	119

TABLE I

COMPARISON OF THE AVERAGE SQUARE ERRORS ON QUERY WITH UNIT LENGTH, I.E. $\left(\frac{SSE}{n}\right)$

[13] for surveys). For example, Bhaskar et al. [14] investigate how frequent itemsets from transaction data can be published. Friedman et al. [15] devise methods for constructing decision trees. Korolova et al. [16] and Götz et al. [17] present methods for publishing statistics in search logs, while McSherry and Mahajan [18] develop techniques for network trace analysis.

Among the existing approaches, the ones most related to ours are by Blum et al. [19], Hay et al. [6], Xiao et al. [8], and Li et al. [7]. Specifically, Blum et al. [19] propose to construct one-dimensional histograms by dividing the input counts into several bins, such that the sum of counts in each bin is roughly the same. The bin counts are then published in a differentially private manner. This approach, however, is shown to be inferior to the method by Hay et al. [6] in terms of the variance of the noise in range count query results.

Hay et al.'s method works by first (i) computing the results of a set of range count queries (with Laplace noise injected), and then (ii) refining the noisy results by exploiting the correlations among the queries. The results obtained thus can then be used to answer any range count queries, and the variance of noise in the query answers is $O(\log^3 n)$, where n is the number of counts in the input data. Hay et al.'s method, as with our solutions, is designed only for one-dimensional data. Meanwhile, Xiao et al. [8] develop a wavelet-based approach that can handle multi-dimensional datasets, and it achieves a noise variance bound of $O(\log^{3d} n)$, where d is the dimensionality of the data set. As shown in our experiments, however, both Hay et al.'s and Xiao et al.'s approaches are outperformed by our techniques in terms of query accuracy.

Li et al. [7] propose an approach that generalizes both Hay et al.'s and Xiao et al.'s techniques in the sense that it can achieve optimal noise variance bound for a large spectrum

of query workloads. In contrast, Hay et al.'s and Xiao et al.'s techniques only optimizes the accuracy of range count queries. Nevertheless, Li et al.'s approach incurs significant computation cost, and hence is inapplicable on large data sets.

In addition, there exist several techniques that address problems similar to (but different from) ours. Barak et al. [20] and Ding et al. [21] propose methods for releasing *marginals*, i.e., projections of a data set onto subsets of its attributes. The core ideas of their methods are to exploit the correlations among the marginals to reduce the amount of noise required for privacy protection. However, neither Barak et al.'s nor Ding et al.'s method can be applied for our problem, as we consider the release of *one* histogram instead of *multiple* marginals.

Xiao et al. [22] devise a differentially private approach that optimizes the relative errors of a given set of count queries. The approach targets the scenario where the count queries overlap with each other (i.e., there exist at least one tuple that satisfies multiple queries), in which case adding less noise in one query result may necessitate a larger amount of noise for another query, so as to ensure privacy protection. Under this setting, Xiao et al.'s approach calibrates the amount of noise in each query result, such that queries with smaller (larger) answers are likely to be injected with less (more) noise, which leads to reduced relative errors. This approach, however, is inapplicable for our problem, since the count queries concerned in a histogram are mutually disjoint.

Rastogi and Nath [23] develop a technique for releasing aggregated results on time series data collected from distributed users. The technique injects noise into a time series by first (i) deriving an approximation of the time series and then (ii) perturbing the approximation. Our histogram construction algorithm is similar in spirit to Rastogi and Nath's technique, in the sense that our algorithm (i) approximates a set D of counts with several bins and (ii) perturbs the bin counts. One may attempt to apply Rastogi and Nath's technique for histogram construction, by regarding the set D of counts as a time series. This approach, however, would lead to highly suboptimal results, since Rastogi and Nath's technique assumes that all information in a time series concerns the same user, in which case changing one user's information could completely change all counts in D.

VIII. Conclusion

In this paper, we present a new differential privacy mechanism supporting arbitrary range queries on numeric domains. Utilizing commonly used histogram techniques, our mechanism generates randomized and biased estimations on the counts in the database. Our method is effective on reducing the sensitivities of the synopsis, thus dramatically cutting the amount of noise added on the synopsis. Experiments on real data sets validate the advantages of our proposals over all state-of-the-art solutions.

IX. Acknowledgement

Jia Xu and Ge Yu are supported by the National Basic Research Program of China (973 Program) under grant 2012CB316201, the National Natural Science Foundation of China (No. 60933001 and No. 61003058), and the Fundamental Research Funds for the Central Universities (No. N100704001). Zhenjie Zhang and Yin Yang are supported by SERC Grant No. 102 158 0074 from Singapore's A*STAR. Xiaokui Xiao is supported by Nanyang Technological University under SUG Grant M58020016 and AcRF Tier 1 Grant RG 35/09, and by the Agency for Science, Technology and Research (Singapore) under SERG Grant 1021580074.

References

[1] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig, "Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays," *PLoS Genetics*, vol. 4, no. 8, 2008.

[2] R. Wang, Y. Li, X. Wang, H. Tang, and X. Zhou, "Learning your identity and disease from research papers: Information leaks in genome wide association study," in *ACM CCS*, 2009.

[3] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC*, 2006, pp. 265–284.

[4] C. Dwork, F. McSherry, and K. Talwar, "The price of privacy and the limits of LP decoding," in *STOC*, 2007, pp. 85–94.

[5] C. Dwork, G. N. Rothblum, and S. P. Vadhan, "Boosting and differential privacy," in *FOCS*, pp. 51–60.

[6] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the accuracy of differentially private histograms through consistency," *PVLDB*, vol. 3, no. 1, pp. 1021–1032, 2010.

[7] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor, "Optimizing linear counting queries under differential privacy," in *PODS*, 2010, pp. 123–134.

[8] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," in *ICDE*, 2010, pp. 225–236.

[9] S. Kotz, T. Kozubowski, and K. Podgórski, *The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance*. Birkhäuser Publication, 2001, pp. 23–23.

[10] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel, "Optimal histograms with quality guarantees," in *VLDB*, 1998, pp. 275–286.

[11] S. Guha, N. Koudas, and K. Shim, "Approximation and streaming algorithms for histogram construction problems," *ACM TODS*, vol. 31, no. 1, pp. 396–438, 2006.

[12] T.-H. H. Chan, E. Shi, and D. Song, "Private and continual release of statistics," in *ICALP (2)*, 2010, pp. 405–417.

[13] C. Dwork, "Differential privacy: A survey of results," in *TAMC*, 2008, pp. 1–19.

[14] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta, "Discovering frequent patterns in sensitive data," in *KDD*, 2010, pp. 503–512.

[15] A. Friedman and A. Schuster, "Data mining with differential privacy," in *KDD*, 2010, pp. 493–502.

[16] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas, "Releasing search queries and clicks privately," in *WWW*, 2009, pp. 171–180.

[17] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke, "Publishing search logs - a comparative study of privacy guarantees," in *TKDE*, in press.

[18] F. McSherry and R. Mahajan, "Differentially-private network trace analysis," in *SIGCOMM*, 2010, pp. 123–134.

[19] A. Blum, K. Ligett, and A. Roth, "A learning theory approach to non-interactive database privacy," in *STOC*, 2008, pp. 609–618.

[20] 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*, 2007, pp. 273–282.

[21] B. Ding, M. Winslett, J. Han, and Z. Li, "Differentially private data cubes: optimizing noise sources and consistency," in *SIGMOD*, 2011, pp. 217–228.

[22] X. Xiao, G. Bender, M. Hay, and J. Gehrke, "ireduct: differential privacy with reduced relative errors," in *SIGMOD*, 2011, pp. 229–240.

[23] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *SIGMOD*, 2010, pp. 735–746.