Towards Accurate Histogram Publication under Differential Privacy

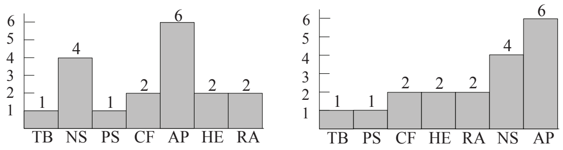
Xiaojian Zhang* Rui Chen† Jianliang Xu‡ Xiaofeng Meng§ Yingtao Xie¶

Abstract

Histograms are the workhorse of data mining and analysis. This paper considers the problem of publishing histograms under differential privacy, one of the strongest privacy models. Existing differentially private histogram publication schemes have shown that clustering (or grouping) is a promising idea to improve the accuracy of sanitized histograms. However, none of them fully exploits the benefit of clustering. In this paper, we introduce a new clustering framework. It features a sophisticated evaluation of the trade-off between the approximation error due to clustering and the Laplace error due to Laplace noise injected, which is normally overlooked in prior work. In particular, we propose three clustering strategies with different orders of run-time complexities. We prove the superiority of our approach by theoretical utility comparisons with the competitors. Our extensive experiments over various standard real-life and synthetic datasets confirm that our technique consistently outperforms existing competitors.

1 Introduction

Obtaining fast and accurate sketches of data distributions is a central problem in data mining and analysis. Histograms are one of the most popular techniques proposed in this context to approximate data distributions. A histogram is a graphical summary of the counts of domain values over a specific domain in a database. For example, if the domain is a set of diseases (e.g., TB, NS, etc), the corresponding histogram is a representation of tabulated counts of the diseases (i.e., number of patients contracting a disease), as illustrated in Figure 1(a).



(a) Original histogram (b) Sorted histogram

Figure 1: Sample histogram and its sorted version

Though making histograms available to researchers may result in knowledge that benefits the general public, it has to be done in a way that does not compromise the privacy of individuals in the database. A standard practice is to safeguard histograms by *differential privacy* [4], one of the strongest privacy models that provides provable privacy guarantees. Intuitively, differential privacy can be satisfied by injecting Laplace noise sufficient for hiding the *maximum* impact of any individual on a histogram. As a result, all record owners can be assured that any privacy breach will not be a result of participating in the database.

There have been some recent works on differentially private histogram publication [2, 15, 13, 9, 5, 17, 16, 1, 7] with the fundamental objective of improving the accuracy of histograms released. Among all these works, the methods based on *clustering* (or *grouping*) [16, 1, 7] provide the most promising accuracy. The general idea of a clustering-based method is to cluster bins with close counts and approximate their counts by the cluster's mean so that we can enjoy reduced Laplace noise. In this process, we encounter two sources of errors: 1) the *approximation error* (AE) due to approximating each bin count by its cluster's mean, and 2) the *Laplace error* (LE) due to Laplace noise injected. The trade-off between the AE and the LE is vital to the final accuracy of a sanitized histogram.

Unfortunately, the existing studies [16, 1, 7] do not balance this trade-off in a desirable way, leaving much room for improvement. Xu et al. [16] and Acs et al. [1] only consider "local" clustering, that is, a bin can only be grouped with adjacent bins, despite the fact that there may exist many bins with close counts far apart. For example, under their methods, in Figure 1(a) the bin CF cannot be in the same cluster as HE and RA unless AP is also in the cluster. Consequently, these methods inevitably incur either large AE (e.g., grouping

*School of Information, Renmin University of China, Beijing, China, xjzhang82@ruc.edu.cn. Henan University of Economics and Law. Part of the work was done when visiting Hong Kong Baptist University

†Department of Computer Science, Hong Kong Baptist University, Hong Kong, ruichen@comp.hkbu.edu.hk

‡Department of Computer Science, Hong Kong Baptist University, Hong Kong, xujl@comp.hkbu.edu.hk

§School of Information, Renmin University of China, Beijing, China, xfmeng@ruc.edu.cn

¶Experiment Center, China West Normal University, Nanchong, China, yingtaoxie@outlook.com

CF, AP, HE and RA together) or large LE (e.g., creating three different clusters)[1]. Furthermore, both methods rely on the exponential mechanism [12] to identify the clusters. When the number of clusters is large, which is the case for most real-life histograms, the formation of clusters becomes imprecise due to the extremely small privacy parameter available for each operation. In addition, specifying the number of clusters as an input is a major hindrance of the method in [16].

Kellaris and Papadopoulos [7] partially address the drawbacks of the above methods by performing a private sorting procedure. Sorting allows "global" clustering. For example, in Figure 1(b), the bins CF, HE and RA can be grouped into a single cluster without including AP. However, their approach does not carefully take into consideration the AE. They simply demand each cluster to be of the same size. In practice, there may not exist a good fixed size that leads to a reasonable trade-off between the two sources of errors (e.g., Figure 1(b)).

Motivated by this, we propose a new clustering framework that publishes more accurate histograms while satisfying differential privacy. Under this framework, we address the aforementioned shortcomings of the existing works. In particular, we propose three clustering schemes with different run-time complexities, including an optimal algorithm, an empirical algorithm and a greedy algorithm. All of these clustering schemes carefully evaluate the trade-off between the AE and the LE in the formation of clusters. We prove the superiority of our approach by providing theoretical accuracy comparisons with our competitors [16, 1, 7]. We also perform an extensive experimental study over various real-life and synthetic datasets, which confirms that our technique consistently outperforms the competitors.

2 Related Work

Numerous recent works [2, 15, 13, 9, 5, 17, 16, 1, 7] have been endeavored to release histograms under differential privacy. Barak et al. [2] consider the problem of releasing differentially private contingency tables. They first apply the Laplace mechanism on the Fourier coefficients of an input dataset, and then use linear programming to generate non-negative contingency tables. Later, in the context of *time-series data*, Rastogi and Nath [13] point out that the magnitude of noise can be reduced by retaining just the first few Fourier coefficients.

Hay et al. [5] make use of the public constraints on the query answers to perform constrained inferences. Xiao et al. [15] apply a wavelet transform on the original histogram and add noise to the wavelet coefficients. Li et al. [9] generalize the methods in [5, 15] under the matrix mechanism. The key idea is to derive the answers to a workload of queries from a different set of queries (known as a query strategy), on which Laplace noise is added. However, the recent work [17] identifies several inherent limits in the matrix mechanism that lower its accuracy in practice, and consequently presents the low-rank mechanism. It answers batch queries based on a low rank approximation of the workload matrix.

Another promising line of research [16, 1, 7] is based on the idea of clustering or grouping. In particular, Acs et al. [1] experimentally show that this line of methods outperforms many aforementioned techniques. Xu et al. [16] propose the NoiseFirst and StructureFirst algorithms. Given the number of clusters in advance, NoiseFirst forms clusters by applying the non-private optimal histogram construction technique [6] over a noisy histogram while StructureFirst finds the cluster boundaries by iteratively applying the exponential mechanism. Their approaches consider only the AE with respect to the non-private optimal histogram and ignore the resultant LE.

Acs et al. [1] improve the methods in [16] by determining the number of clusters on the fly. They design a hierarchical bisection procedure over a histogram to identify good clusters. Kellaris and Papadopoulos [7] consider a slightly different problem setting, in which a record owner may contribute to multiple bins. They first employ sampling to sort the underlying histogram and then divide the bins into groups with a fixed size. Unfortunately, using the same size for all groups normally leads to unbalanced AE. As mentioned in Section 1, all these works [16, 1, 7] have some inherent design limits that prevent them from fully exploiting the benefit of clustering. In Section 5 and Section 6, we provide both a formal theoretical analysis and an experimental evaluation to show that our solution achieves better accuracy.

3 Preliminaries

3.1 Histogram. Consider an attribute X with the value set $\mathcal{V}$ (either *numerical* or *nominal*) in a database D. For each value $v \in \mathcal{V}$, its *count* (or *frequency*) is the number of tuples $t \in D$ with $t.X = v$. The histogram $\mathbf{H}$ over the attribute X consists of a set of *bins*: $\mathbf{H} = \{H_1, H_2, \cdots, H_n\}$, where each bin H_i is associated with a range of values it covers, denoted by $H_i.range$, and is of a *count* (or *frequency*) equal to the sum of the counts of the values covered by H_i (i.e., the number of tuples in $H_i.range$). Normally, for any $i \neq j$, $H_i.range \cap H_j.range = \emptyset$, and $|D| = \sum_{i=1}^{n} H_i$, where $|D|$ is the number of tuples in D. In this paper, we consider *attributed* histograms, that is, each bin's

[1] As will be shown later, the LE is proportional to the number of clusters.

range of values is retained. An attributed histogram can be used to answer a wide range of queries (e.g., range queries).

3.2 Differential Privacy. Let $\mathbf{H}_1$ and $\mathbf{H_2}$ be two histograms derived from two databases D_1 and D_2 that differ by at most one record. We call $\mathbf{H_1}$ and $\mathbf{H_2}$ *neighbors*. It can be observed that any two neighboring histograms differ in exactly one bin with difference on the count at exactly 1. Differential privacy [4] in our problem is formally defined as follows.

Definition 3.1. *A privacy algorithm $\mathcal{A}$ satisfies ϵ-differential privacy, if for any two neighboring histograms $\mathbf{H}_1$ and $\mathbf{H}_2$, and for any $O \in Range(\mathcal{A})$,*

$$\Pr[\mathcal{A}(\mathbf{H}_1) = O] \leq \exp(\epsilon) \cdot \Pr[\mathcal{A}(\mathbf{H}_2) = O]$$

where the probability is taken over $\mathcal{A}$'s randomness. ■

The parameter $\epsilon > 0$ specifies the desired level of privacy. The smaller ϵ, the better privacy protection. Typically, ϵ is small (e.g., $\epsilon \leq 1$).

In the literature, there are two well-established techniques to achieve differential privacy: the *Laplace mechanism* [4] and the *exponential mechanism* [12]. Both are based on the concept of *global sensitivity* of a function f to compute over a histogram. For any two neighboring histograms $\mathbf{H}_1$ and $\mathbf{H}_2$, the global sensitivity of a function $f : \mathbf{H} \to \mathbb{R}^d$ is defined as $\Delta f = \max_{\mathbf{H}_1,\mathbf{H}_2} \|f(\mathbf{H}_1) - f(\mathbf{H}_2)\|_1$.

The Laplace mechanism is used in our solution. Intuitively, it requires to mask the impact of a record owner by adding properly calibrated Laplace noise. More precisely, given a histogram $\mathbf{H}$, a function f, and the privacy parameter ϵ, the noise is drawn from a Laplace distribution with the probability density function $p(x|\lambda) = \frac{1}{2\lambda}e^{-|x|/\lambda}$, where $\lambda = \Delta f/\epsilon$.

Theorem 3.1. [4] *For any function $f : \mathbf{H} \to \mathbb{R}^d$, the mechanism $\mathcal{A}$*

$$\mathcal{A}(\mathbf{H}) = f(\mathbf{H}) + \langle \mathtt{Lap}_1(\frac{\Delta f}{\epsilon}), \ldots, \mathtt{Lap}_d(\frac{\Delta f}{\epsilon}) \rangle$$

gives ϵ-differential privacy, where $\mathtt{Lap}_i(\frac{\Delta f}{\epsilon})$ are i.i.d Laplace variables with scale parameter $\frac{\Delta f}{\epsilon}$. ■

3.3 Utility Metrics. Two popular data analysis tasks conducted over an attributed histogram are examining data distributions and answering range queries [16, 1, 7]. Consequently, we measure the utility of the released histograms in terms of *Kullback-Leibler divergence* (KLD) and *mean squared error* (MSE).

KLD is used to quantify the difference of the probability distributions between the original histogram $\mathbf{H}$ and the sanitized histogram $\widetilde{\mathbf{H}}$. Formally, $KLD(\mathbf{H} \parallel \widetilde{\mathbf{H}}) = \sum_{i=1}^{n} H_i \ln \frac{H_i}{\widetilde{H_i}}$, where n is the number of bins in $\mathbf{H}$. If $\mathbf{H} = \widetilde{\mathbf{H}}$, then $KLD(\mathbf{H}\|\widetilde{\mathbf{H}}) = 0$. We follow the standard convention that $0\ln 0 = 0$.

The utility of the sanitized histogram $\widetilde{\mathbf{H}}$ over a set of range queries $\mathbf{Q} = \{Q_1, Q_2, \cdots, Q_m\}$ is quantified by the MSE as: $MSE(\mathbf{H}, \widetilde{\mathbf{H}}, \mathbf{Q}) = \frac{\sum_{i=1}^{m}(Q_i(\mathbf{H}) - Q_i(\widetilde{\mathbf{H}}))^2}{m}$, where $Q_i(\mathbf{H})$ and $Q_i(\widetilde{\mathbf{H}})$ return the answers of Q_i on $\mathbf{H}$ and $\widetilde{\mathbf{H}}$, respectively.

4 Sanitization Algorithm

In this section, we present our clustering framework that carefully seeks for a better trade-off between the `AE` and the `LE` and thus achieves better accuracy.

4.1 Overview. Our objective is to generate a sanitized histogram that minimizes its error with respect to the original histogram while satisfying ϵ-differential privacy. Let $\mathbb{E}(\cdot)$ be the expected value. Formally, the error of a sanitized histogram $\widetilde{\mathbf{H}}$ is defined as:

$$err(\widetilde{\mathbf{H}}) = \mathbb{E}(\left\|\mathbf{H} - \widetilde{\mathbf{H}}\right\|_2^2) = \mathbb{E}(\sum_{i=1}^{n}(H_i - \widetilde{H_i})^2).$$

Similarly, the error of a bin $\widetilde{H_i}$ is $err(\widetilde{H_i}) = (H_i - \widetilde{H_i})^2$.

As mentioned before, in a clustering-based approach, $err(\widetilde{\mathbf{H}})$ is composed of two types of errors: 1) the *approximation error* (`AE`) due to approximating each bin count by its cluster's mean, and 2) the *Laplace error* (`LE`) due to Laplace noise added. There exists a fundamental trade-off between these two sources of errors: in general forming more clusters decreases `AE` at the cost of larger `LE`. We have to bear this trade-off in mind for designing an effective solution.

Our clustering framework is summarized in Algorithm 1. We first employ a noisy sorting procedure over the input histogram $\mathbf{H}$, then group bins with close counts into the same cluster, and finally replace each bin count with the sum of its cluster's mean and reduced Laplace noise. To minimize the resultant error, each step needs careful design.

4.2 Detailed Algorithm. The success of our solution lies in a good formation of clusters. A natural starting point is to sort all bins based on their counts and then apply clustering over sorted bins. However, the sorting cannot be done based on their true counts; otherwise differential privacy will be violated [7]. For this reason, we obtain the noisy counts of the bins by the Laplace mechanism using a portion of the total privacy parameter ϵ_1 (Line 2).

To generate more accurate sorting results, we would like to mitigate the impact of Laplace noise added. One attempt toward this goal is to employ *sampling* techniques based on the following theorem [10], as suggested by Kellaris and Papadopoulos [7].

THEOREM 4.1. [10] *If an algorithm $\mathcal{A}$ satisfies ϵ-differential privacy, the algorithm $\mathcal{A}^{\beta}$, for any $0 < \beta < 1$, satisfies $\ln(1 + \beta(e^{\epsilon} - 1))$-differential privacy, where $\mathcal{A}^{\beta}$ denotes the algorithm to first sample with probability β and then apply $\mathcal{A}$ to the sampled dataset.* ■

Unfortunately, the *column sampling* technique does not fit our problem setting in its nature because a record owner falls into exactly one bin, while the *row sampling* technique only generates less precise sorting results. To reliably sort two bins H_i and H_j based on their noisy counts, it is natural to require their true difference to be much larger than the magnitude of noise. Therefore, the ratio of their difference $|H_i - H_j|$ to the standard deviation of noise, $\frac{|H_i - H_j|}{\sqrt{2}/\epsilon_1}$, is a good indicator of the sorting quality. When row sampling is applied, the ratio becomes $\frac{\beta|H_i - H_j|}{\sqrt{2}/\ln(1+\beta(e^{\epsilon_1}-1))}$. Since $\frac{|H_i - H_j|}{\sqrt{2}/\epsilon_1} > \frac{\beta|H_i - H_j|}{\sqrt{2}/\ln(1+\beta(e^{\epsilon_1}-1))}$ for any $0 < \beta < 1$, we learn that row sampling only leads to less precise sorting.

In this paper, we use a thresholding strategy to alleviate the influence of noise, following the observation that, in many real-life histograms, there exist many bins with a zero count. Adding additive Laplace noise to these bins creates many artificially non-zero counts, which introduces a major disturbance to the sorting process. The `Threshold` procedure (Line 3) essentially applies a high-pass filter to $\widehat{\mathbf{H}}$ as follows:

$$\widehat{H_i} = \begin{cases} \widehat{H_i} & if\ \widehat{H_i} \geq \theta \\ 0 & otherwise \end{cases},$$

where $\theta = \eta \log(n)/\epsilon_1$ and $\eta > 0$ is a tuning parameter. The selection of θ follows the intuition that the maximal magnitude of n Laplace variables will be $O(\frac{\log n}{\epsilon_1})$ [8]. `Threshold` smooths the noise added to the bins of relatively small counts and leads to a more accurate sorting result. We stress that even if the aforementioned observation does not hold for a particular input histogram, `Threshold` does no harm to the entire algorithm.

Sorting then can be conducted on the smoothed $\widehat{\mathbf{H}}$ using any sorting algorithm (Line 4). Since `Sort` is performed over the differentially private $\widehat{\mathbf{H}}$, it does not incur any extra privacy cost.

Next we perform clustering over the sorted noisy histogram $\widehat{\mathbf{H}}$. To better balance the `AE` and the `LE`, we need a deeper understanding of $err(\widehat{\mathbf{H}})$ by explicitly representing it in terms of these two sources of errors. Let $err(C_i)$ be the error of the cluster C_i.

Algorithm 1 Accurate Histogram Publication (`AHP`)

```
Input: Original histogram $\mathbf{H} = \{H_1, H_2, \cdots, H_n\}$
Input: Privacy parameter $\epsilon$
Output: Sanitized histogram $\widetilde{\mathbf{H}}$
 1: $\epsilon = \epsilon_1 + \epsilon_2$;
 2: $\widehat{\mathbf{H}} = \mathbf{H} + \langle \texttt{Lap}(\frac{1}{\epsilon_1}) \rangle^n$;
 3: $\widehat{\mathbf{H}} = \texttt{Threshold}(\widehat{\mathbf{H}})$;
 4: $\widehat{\mathbf{H}} = \texttt{Sort}(\widehat{\mathbf{H}})$;
 5: $\mathbf{C} = \texttt{Cluster}(\widehat{\mathbf{H}})$;
 6: for each $C_i \in \mathbf{C}$ do
 7:     $\overline{C}_i = \sum_{H_j \in C_i} H_j / |C_i|$;
 8: for each $H_j \in \mathbf{H}$ do
 9:     $\widetilde{H_j} = \overline{C}_i + \frac{\texttt{Lap}(1/\epsilon_2)}{|C_i|}$, where $H_j \in C_i$;
10: return $\widetilde{\mathbf{H}} = \{\widetilde{H_1}, \widetilde{H_2}, \cdots, \widetilde{H_n}\}$;
```

THEOREM 4.2. *Given a cluster C_i generated by Algorithm 1, $err(C_i) = \sum_{H_j \in C_i} (H_j - \overline{C}_i)^2 + \frac{2}{|C_i|(\epsilon_2)^2}$, where $\overline{C}_i$ is the mean of C_i, $\sum_{H_j \in C_i} (H_j - \overline{C}_i)^2$ is the* `AE`*, and $\frac{2}{|C_i|(\epsilon_2)^2}$ is the* `LE`*.* ■

Proof. By definition, $\overline{C}_i = \frac{\sum_{H_j \in C_i} H_j}{|C_i|}$. Since every bin count in C_i is set to $\overline{C}_i + \frac{\texttt{Lap}(1/\epsilon_2)}{|C_i|}$ (Line 9), $err(C_i)$ can be calculated as follows:

$$\begin{aligned} err(C_i) &= \mathbb{E}\left(\sum_{H_j \in C_i} \left(H_j - \overline{C}_i - \frac{\texttt{Lap}(1/\epsilon_2)}{|C_i|}\right)^2\right) \\ &= \sum_{H_j \in C_i} \left(H_j - \overline{C}_i\right)^2 + \mathbb{E}\left(\sum_{H_j \in C_i} \left(\frac{\texttt{Lap}(1/\epsilon_2)}{|C_i|}\right)^2\right) \\ &= \sum_{H_j \in C_i} \left(H_j - \overline{C}_i\right)^2 + \frac{2}{|C_i|\,(\epsilon_2)^2} \end{aligned}$$

This proves the theorem. ■

Consequently, we propose three clustering strategies to instantiate the `cluster` procedure (Line 5) with different orders of run-time complexities.

Optimal Clustering Algorithm. Given the preprocessed noisy histogram $\widehat{\mathbf{H}}$, the first clustering algorithm aims to identify the optimal clusters that minimize $err(\widehat{\mathbf{H}})$. We slightly abuse the term *optimal* in the sense that the optimality is with respect to $\widehat{\mathbf{H}}$, instead of the original histogram $\mathbf{H}$.

Our optimal algorithm is built on the dynamic programming technique proposed in [6], which finds the optimal clusters for a given number of clusters k. However, the `LE` is not a concern in the problem of [6]. Denote the partial histogram containing $\{\widehat{H_i}, \widehat{H_{i+1}}, \cdots, \widehat{H_j}\}$ by $\widehat{\mathbf{H}}_{i,j}$. Define $err^*(\widehat{\mathbf{H}}_{1,j}, m)$ to be the minimum error for

$\widehat{\mathbf{H}}_{1,j}$ using at most m clusters. To correctly reflect the LE, we accordingly modify the recursive rule in dynamic programming as follows:

$$err^*(\widehat{\mathbf{H}}_{1,j}, m) = \min_{1 \le l < j} \{err^*(\widehat{\mathbf{H}}_{1,l}, m-1) + err(\widehat{\mathbf{H}}_{l+1,j})\}.$$

It is easy to verify the correctness of this rule[2]. To find the minimum $err(\widehat{\mathbf{H}})$, we consider all possible numbers of clusters, that is, $k = 1, 2, \cdots, n$, and return the clusters that achieve the minimum error.

This algorithm is guaranteed to find the optimal clusters, but its computational cost is prohibitively high. The run-time complexity of finding the optimal clustering scheme for a given number of clusters k is $O(kn^2)$ [6], and therefore the complexity of our optimal clustering algorithm is $O(n^4)$, making it impractical for most real-life applications.

Empirical Clustering Algorithm. Realizing that the major computational cost of the optimal clustering algorithm comes from examining *all* possible numbers of clusters k, we propose an empirical clustering algorithm, which estimates some good k values based on well-established statistical results. In particular, we consider four rules of thumb:

- Square-root choice: $k = \sqrt{n}$;
- Sturges' formula [14]: $k = \lceil 1 + \log_2 n \rceil$;
- Doane's formula [3]: $k = 1 + \log_2 n + \log_2(1 + \frac{|s|}{\sigma})$, where $s = \left| \frac{\sum_{1 \le i \le n} (\widehat{H_i} - \sum_{1 \le j \le n} \widehat{H_j}/n)^3}{(\sum_{1 \le i \le n} (\widehat{H_i} - \sum_{1 \le j \le n} \widehat{H_j}/n)^2)^{\frac{3}{2}}} \right|$ and $\sigma = \sqrt{\frac{6(n-2)}{(n+1)(n+3)}}$;
- Rice Rule[3]: $k = \lceil 2n^{\frac{1}{3}} \rceil$.

For each k value, we employ the dynamic-programming-based clustering algorithm explained before to find the optimal clusters, denoted by OptimalCluster. We use the best clustering scheme derived from these four k values as the final scheme. Algorithm 2 shows the details of the empirical clustering algorithm.

It can be seen that the empirical clustering algorithm successfully reduces its complexity to $O(n^2\sqrt{n})$. However, the empirical k values normally assume certain underlying data distribution, and may perform poorly if the data are not distributed as assumed. This motivates our third algorithm, which adaptively identifies the best k value by balancing the trade-off between the AE and the LE.

Algorithm 2 Empirical Clustering Algorithm

```
Input: Sorted histogram $\widehat{\mathbf{H}} = \{\widehat{H_1}, \widehat{H_2}, ..., \widehat{H_n}\}$
Output: Clusters $\mathbf{C}$
$\mathbf{K}$= $\{\sqrt{n}, \lceil 1 + \log_2 n \rceil, 1 + \log_2 n + \log_2(1 + \frac{|s|}{\sigma}), \lceil 2n^{\frac{1}{3}} \rceil\}$;    //see main text for detail
for each $k_i \in \mathbf{K}$ do
    $\mathbf{C}_i$ = OptimalCluster($\widehat{\mathbf{H}}, k_i$);
    Calculate $err(\widehat{\mathbf{H}}, \mathbf{C}_i)$;
return $\mathbf{C} = \underset{\mathbf{C}_i}{\text{argmin}}(err(\widehat{\mathbf{H}}, \mathbf{C}_i))$;
```

Greedy Clustering Algorithm. Instead of asking for a k value in advance, our greedy clustering algorithm iteratively decides whether to add the next bin $\widehat{H_j}$ into the current cluster C_i. The decision is guided by the resulting error: if merging $\widehat{H_j}$ into C_i leads to lower error, we add $\widehat{H_j}$ into C_i; otherwise, we create a new cluster. The detail is given in Algorithm 3.

When we add $\widehat{H_j}$ to C_i, the resulting error $err(C_i \cup \widehat{H_j})$ is the sum of the AE and the LE of the new cluster $C_i \cup \widehat{H_j}$. We have $err(C_i \cup \widehat{H_j}) = \mathtt{AE}(C_i \cup \widehat{H_j}) + \frac{2}{(|C_i|+1)(\epsilon_2)^2}$. When we do not add $\widehat{H_j}$ to C_i, the error is $err(C_i) + err(\widehat{H_j})$, where $err(C_i)$ is the total error of C_i and $err(\widehat{H_j})$ is the total error on $\widehat{H_j}$. It is easy to observe that $err(C_i) = \mathtt{AE}(C_i) + \frac{2}{|C_i|(\epsilon_2)^2}$. However, the calculation of $err(\widehat{H_j})$ needs more efforts.

Keeping $\widehat{H_j}$ out of C_i implies $n - j + 1$ possible clusters in which $\widehat{H_j}$ may reside, and therefore $err(\widehat{H_j})$ is a variable depending on the actual formation of the cluster containing $\widehat{H_j}$, which is *not* known at this moment. Fortunately, it is sufficient to make the correct decision if we can learn the lower bound of $err(\widehat{H_j})$, denoted by $err^*(\widehat{H_j})$. Let $C_{j,l}$ be the potential cluster containing $\{\widehat{H_j}, \widehat{H_{j+1}}, \cdots, \widehat{H_l}\}$, where $j \le l \le n$. We get $err^*(\widehat{H_j}) = \min_l\{(\widehat{H_j} - \overline{C}_{j,l})^2 + \frac{2}{|C_{j,l}|^2(\epsilon_2)^2}\}$.

To efficiently calculate $\overline{C}_{j,l}$, we maintain a prefix sum array $\mathbf{P}$ of length n with $\mathbf{P}[i] = \sum_{m=1}^{i} \widehat{H_m}$. Then we have $\overline{C}_{j,l} = \frac{\mathbf{P}[l] - \mathbf{P}[j-1]}{l-j+1}$. We further speed up the calculation of $err^*(\widehat{H_j})$ by making use of the monotonicity of $(\widehat{H_j} - \overline{C}_{j,l})^2$ and $\frac{2}{|C_{j,l}|^2(\epsilon_2)^2}$: with the increase of l, $(\widehat{H_j} - \overline{C}_{j,l})^2$ monotonically increases while $\frac{2}{|C_{j,l}|^2(\epsilon_2)^2}$ monotonically decreases.

We consider the possible clusters containing $\widehat{H_j}$ in the ascending order of their sizes. As soon as $(\widehat{H_j} -$

[2] Note that the faster implementation proposed in [6] cannot be used here because we do *not* always have $err(\widehat{\mathbf{H}}_{i,j}) \ge err(\widehat{\mathbf{H}}_{i,l}) + err(\widehat{\mathbf{H}}_{l+1,j})$ for $0 \le i < l < j \le n$.

[3] Online statistics education: A multimedia course of study (http://onlinestatbook.com).

Algorithm 3 Greedy Clustering Algorithm

```
Input: Sorted histogram $\widehat{\mathbf{H}} = \{\widehat{H_1}, \widehat{H_2}, ..., \widehat{H_n}\}$
Output: Clusters $\mathbf{C}$
$\mathbf{C} = \emptyset$;
$i = 1$;
$j = 2$;
$C_i = \{\widehat{H_1}\}$;
while $j \leq n$ do
    Calculate $err(C_i \cup \widehat{H_j})$;
    Calculate $err(C_i) + err^*(\widehat{H_j})$;
    if $err(C_i \cup \widehat{H_j}) < err(C_i) + err^*(\widehat{H_j})$ then
        $C_i = C_i \cup \widehat{H_j}$;
    else
        $\mathbf{C} = \mathbf{C} \cup C_i$;
        $i++$;
        $C_i = \{H_j\}$;
    $j++$;
return $\mathbf{C}$;
```

$\overline{C}_{j,l+1})^2 - (\widehat{H_j} - \overline{C}_{j,l})^2 \geq \frac{2}{|C_{j,l}|^2(\epsilon_2)^2} - \frac{2}{(n-j+1)^2(\epsilon_2)^2}$, we can conclude that $err^*(\widehat{H_j}) = (\widehat{H_j} - \overline{C}_{j,l})^2 + \frac{2}{|C_{j,l}|^2(\epsilon_2)^2}$. The rationale behind this stop condition is that once the increase of the AE between $C_{j,l+1}$ and $C_{j,l}$ is greater than the *maximum* decrease of the LE, we have missed the minimum point. Note that $\frac{2}{(n-j+1)^2(\epsilon_2)^2}$ is the minimum LE we can achieve on $\widehat{H_j}$.

Example 3.1. Let $\widehat{\mathbf{H}} = \{1, 1, 3, 3, 4, 6, 7\}$ be the sorted noisy histogram. Assume that the current cluster $C_1 = \{1, 1\}$ and $\epsilon_2 = 0.5$. We consider whether to add $\widehat{H_3} = 3$ to C_1. We have $err(C_1 \cup \widehat{H_3}) = \mathtt{AE}(C_1 \cup \widehat{H_3}) + \frac{2}{(2+1)(\epsilon_2)^2} = \frac{16}{3}$ and $err(C_1) = 4$. To calculate $err^*(\widehat{H_3})$, we consider 5 potential clusters, and obtain $err^*(\widehat{H_3}) = \frac{1}{9} + \frac{8}{9} = 1$ when the potential cluster containing $\widehat{H_3}$ is $\{3, 3, 4\}$. Since $err(C_1 \cup \widehat{H_3}) > err(C_1) + err^*(\widehat{H_3})$, we keep the cluster $C_1 = \{1, 1\}$ as it is. Similarly, we find $C_2 = \{3, 3, 4\}$ and $C_3 = \{6, 7\}$. ■

Once the clusters have been identified, for a bin $H_j \in C_i$, its noisy count is set to $\overline{C}_i + \frac{\mathtt{Lap}(1/\epsilon_2)}{|C_i|}$ (Line 9 of Algorithm 1). The run-time complexity of Algorithm 1 when coupled with the greedy clustering strategy is $O(n^2)$, where n is the number of bins in $\mathbf{H}$.

5 Analysis

In this section, we provide the critical analysis on the privacy and utility of our solution (referred to as AHP).

Privacy Analysis. The theorem below shows that AHP is ϵ-differentially private.

THEOREM 5.1. AHP *satisfies ϵ-differential privacy.* ■

Proof. Line 2 of Algorithm 1 employs the Laplace mechanism to calculate the noisy count of each bin. Let this set of count queries be $\mathbf{Q}_1$. By definition of neighboring histograms, we have $\Delta\mathbf{Q}_1 = 1$ and therefore Line 2 satisfies ϵ_1-differential privacy. Line 9 of Algorithm 1 employs the Laplace mechanism to calculate the noisy mean of each cluster. Since the size of each cluster has been known, this is equivalent to asking for the noisy sum of each cluster. Let this set of queries be $\mathbf{Q}_2$. Similarly, we have $\Delta\mathbf{Q}_2 = 1$ and hence Line 9 satisfies ϵ_2-differential privacy. Using the *sequential composition* property [11] stated below, we learn that Line 2 and Line 9 together satisfy $(\epsilon_1 + \epsilon_2)$-differential privacy.

LEMMA 5.1. [11] *Let each $\mathcal{A}_i$ provide ϵ_i-differential privacy. A sequence of $\mathcal{A}_i(D)$ over the database D provides $\sum \epsilon_i$-differential privacy.*

The rest lines of AHP are based on differentially private results, and they do not incur any extra privacy cost [5]. So AHP as a whole satisfies ϵ-differential privacy. ■

Utility Analysis. We first give the error of the sanitized histogram generated by AHP.

THEOREM 5.2. *Given an input histogram $\mathbf{H}$ and its sanitized version $\widetilde{\mathbf{H}}$ computed by* AHP*,*

$$err(\widetilde{\mathbf{H}}) = \sum_{C_i \in \mathbf{C}} \sum_{H_j \in C_i} \left(H_j - \overline{C}_i\right)^2 + \sum_{C_i \in \mathbf{C}} \frac{2}{|C_i|(\epsilon_2)^2}. \blacksquare$$

The proof of Theorem 5.2 directly follows Theorem 4.2. Now we show that the utility of AHP is better than our state-of-the-art competitors: NoiseFirst (referred to as NF) [16], StructureFirst (SF) [16], P-HPartion (PHP) [1] and GS [7]. Since all these methods are *data-dependent*, it is very difficult to give rigorous quantitative comparisons among them. However, we can still elaborate why AHP leads to better utility.

Comparison with NF and SF. We first focus on the error of a particular cluster C_i. From [16], we learn that $err_{\mathtt{NF}}(C_i) = \mathtt{AE}(C_i) + \frac{2}{\epsilon^2}$ and $err_{\mathtt{SF}}(C_i) > \mathtt{AE}(C_i) + \frac{2}{(\epsilon_2)^2}$. In SF, ϵ is divided into ϵ_1 and ϵ_2, where ϵ_1 is used to form clusters while ϵ_2 is used to generate noisy counts. When C_i contains the same bins under all three methods, we can directly observe that $err_{\mathtt{AHP}}(C_i) = \mathtt{AE}(C_i) + \frac{2}{|C_i|(\epsilon_2)^2} < err_{\mathtt{SF}}(C_i)$ if ϵ_2's are equal in both methods and that $err_{\mathtt{AHP}}(C_i) < err_{\mathtt{NF}}(C_i)$ if $|C_i| > (\frac{\epsilon}{\epsilon_2})^2$. In practice, ϵ_2 is normally set to $\frac{\epsilon}{2}$, and therefore as long as $|C_i| > 4$, which is the case for most clusters, $err_{\mathtt{AHP}}(C_i) < err_{\mathtt{NF}}(C_i)$.

Based on the above analysis, we can discuss the error of the entire sanitized histogram. In general, due to the sorting procedure and the better clustering strategy, AHP obtains larger cluster sizes and lower AE within each cluster (this is especially true for SF due to the extremely small privacy parameter used to identify each cluster). Larger cluster sizes imply a smaller number of clusters and hence smaller LE. This indicates that AHP achieves smaller total error.

Comparison with PHP. Similarly, PHP uses ϵ_1 to form the clusters and ϵ_2 to generate the final noisy counts. The error of a particular cluster C_i under PHP is also $err_{\texttt{PHP}}(C_i) = \texttt{AE}(C_i) + \frac{2}{|C_i|(\epsilon_2)^2}$. So the difference of error between PHP and AHP is solely determined by the quality of the clusters. Analogously, the sorting procedure and the better clustering scheme of AHP generate smaller number of clusters with smaller AE. This guarantees that AHP achieves smaller total error.

Comparison with GS. As indicated before, column sampling does not fit our problem setting and row sampling only leads to less precise sorting results. Here we discuss the version of GS with *no* sampling. The sorted noisy histogram in GS is at most as precise as that in AHP. The utility difference between GS and AHP is determined by the clustering procedure. Let the fixed cluster size chosen by GS be w. Below we establish the critical condition under which a cluster can be decomposed into two clusters with smaller total error.

THEOREM 5.3. *For a cluster $C_i = \{H_1, H_2, \cdots, H_w\}$ with $H_1 \leq H_2 \leq \cdots \leq H_w$, if $H_{j+1} - H_j \geq \frac{\sqrt{2(w^2-w+1)}}{\epsilon_2(w-1)}$ for some $1 \leq j < w$, then $err(C_i) \geq err(C_i^1) + err(C_i^2)$ where $C_i^1 = \{H_1, H_2, \cdots, H_j\}$ and $C_i^2 = \{H_{j+1}, H_{j+2}, \cdots, H_w\}$.*

Proof. By definition, we have $err(C_i) = \texttt{AE}(C_i) + \frac{2}{w(\epsilon_2)^2}$, $err(C_i^1) = \texttt{AE}(C_i^1) + \frac{2}{j(\epsilon_2)^2}$ and $err(C_i^2) = \texttt{AE}(C_i^2) + \frac{2}{(w-j)(\epsilon_2)^2}$. To prove the theorem, we need to prove that $\texttt{AE}(C_i) - \texttt{AE}(C_i^1) - \texttt{AE}(C_i^2) \geq \frac{2}{j(\epsilon_2)^2} + \frac{2}{(w-j)(\epsilon_2)^2} - \frac{2}{w(\epsilon_2)^2}$. For the right hand side (RHS) of the inequality, we have:

$$RHS = \frac{2}{(\epsilon_2)^2}\left(\frac{w}{j(w-j)} - \frac{1}{w}\right) \leq \frac{2}{(\epsilon_2)^2} \cdot \frac{w^2-w+1}{w(w-1)}.$$

Let $H_{j+1} - H_j = \delta$. For the left hand side (LHS), we have two facts: $\overline{C}_i - \overline{C}_i^1 \geq \frac{\delta \cdot (w-j)}{w}$ and $\overline{C}_i^2 - \overline{C}_i \geq \frac{\delta \cdot j}{w}$. Then we have:

$$\begin{aligned} LHS &= j \cdot \left(\overline{C}_i - \overline{C}_i^1\right)^2 + (w-j) \cdot \left(\overline{C}_i - \overline{C}_i^2\right)^2 \\ &\geq j \cdot \frac{\delta^2 (w-j)^2}{w^2} + (w-j) \cdot \frac{j^2 \delta^2}{w^2} \\ &= j(w-j) \cdot \frac{\delta^2}{w} \\ &\geq (w-1) \cdot \frac{\delta^2}{w} \end{aligned}$$

Table 1: Experimental dataset characteristics.

Dataset	|H|	Number of zero count	Mean	Variance	Count Range
Location	7,725	4,720	24.13	4,764.57	[0, 467]
Social Network	11,342	0	59.49	2995	[1, 1,678]
Search Log	32,768	17,082	10.25	577.31	[0, 496]
NetTrace	65,536	63,318	0.39	91.01	[0, 1,423]

To make $LHS \geq RHS$, it is sufficient to require

$$(w-1) \cdot \frac{\delta^2}{w} \geq \frac{2}{(\epsilon_2)^2} \cdot \frac{w^2-w+1}{w(w-1)},$$

which leads to $\delta \geq \frac{\sqrt{2(w^2-w+1)}}{\epsilon_2(w-1)}$. ■

Theorem 5.3 suggests that even if w in GS is chosen to be the optimal, we can still find a better clustering scheme with non-uniform cluster sizes if the critical condition holds. To give an intuitive impression on the magnitude of $\frac{\sqrt{2(w^2-w+1)}}{\epsilon_2(w-1)}$, let $\epsilon_2 = 0.5$ and $w = 100$[4]. We get $\frac{\sqrt{2(w^2-w+1)}}{\epsilon_2(w-1)} \approx 2.84$, suggesting that many clusters identified by GS could be further decomposed to obtain smaller total error. In contrast, AHP adaptively identifies the clusters that result in the minimal total error, regardless of their sizes.

6 Experimental Evaluation

This section experimentally evaluates the utility of AHP in terms of the two utility metrics introduced in Section 3.3, namely the *KLD* of data distributions and the *MSE* of range queries. Due to the space limit, we only report the results of AHP integrating the greedy clustering strategy (Algorithm 3), which achieves the best trade-off between utility and efficiency. We compare AHP with our competitors: NF [16], SF [16], PHP [1] and GS [7]. AHP is implemented in C++, and all experiments were performed on an Intel Core 2 Duo 2.94GHz CPU with 4GB RAM.

Our experiments are based on four standard real-life and synthetic datasets used in [5, 1, 16]. *Location* is a half-synthetic dataset that involves a total population of 186,471 distributed in 1,340 meshblocks (or bins). It is generated from the 2006 Census Meshblock Dataset of New Zealand[5]. *Social Network* records the friendship relations among 11K users in a social network website. Each user's record contains her number of friends. *Search Log* is a synthetic dataset generated by integrating *Google Trends* data and *American Online* search logs during the period between 2004 and 2010. Each bin contains the number of the keyword "Obama"

[4] $w = 100$ is one of the best group sizes used by GS.

[5] http://www.stats.govt.nz/Census/2006CensusHomePage/MeshblockDataset.aspx

Table 2: KLD on four datasets (ϵ=1)

Datasets	AHP	NF	SF	PHP	GS
Location	0.018	0.843	1.242	0.621	1.451
Social Network	0.071	1.001	2.051	0.801	2.611
Search Log	0.054	1.582	2.085	0.132	2.115
NetTrace	0.153	2.839	3.835	0.904	4.816

Table 3: KLD on four datasets (ϵ=0.1)

Datasets	AHP	NF	SF	PHP	GS
Location	0.203	1.381	2.983	0.805	1.539
Social Network	0.309	2.753	2.054	0.602	3.251
Search Log	0.103	2.467	3.457	0.159	4.157
NetTrace	0.572	4.726	4.359	2.026	6.917

issued within a 90-minute interval. *NetTrace* describes the IP-level network traces collected from a university intranet. Each bin gives the number of external hosts connected to an internal host. We summarize the characteristics of these four datasets in Table 1.

Data Distribution. In the first set of experiments, we demonstrate the utility of AHP for data distribution in terms of KLD. Tables 2-4 present the KLD for all datasets under various privacy parameters ($\epsilon = 0.01$, 0.1 and 1 [16, 1]). It can be observed that AHP performs significantly better than the other methods in all cases. It is worth mentioning that *Social Network* and *NetTrace* are originally sorted, and therefore the sorting procedure of our solution does not really help. In this case, the improvement of KLD is solely due to the greedy clustering strategy. It confirms that our clustering strategy indeed identifies better clusters.

Range Query. In the second set of experiments, we examine the performance of all methods on range queries with respect to varying range sizes and different privacy parameters. We follow the evaluation scheme from previous works [16, 1] and report their accuracy in terms of MSE. The results are shown in Figures 2-5. All figures exhibit the same trend: MSE increases when ϵ decreases or when the query range size increases, which conforms to the theoretical analysis.

We can observe that AHP outperforms all competitors in all cases. In particular, on *Location* and *Search Log* that are *not* originally sorted, the improvement of MSE is substantial (more than 8 times). This suggests that the sorting procedure is very important to unsorted datasets. On *Social Network* and *NetTrace* that are originally sorted, AHP still achieves smaller MSE, though its MSE is close to PHP when ϵ is relatively large (e.g., 0.1 and 1). This is because PHP inherently fits sorted histograms best, where its "local" clustering becomes "global".

7 Conclusion

In this paper, we investigated the problem of publishing histograms under differential privacy. Based on the observation that the existing clustering (or grouping) based schemes do not fully exploit the power of clustering, we introduced a new clustering framework. In particular, we proposed three different clustering strategies. All of them effectively balance the trade-off between the approximation error and the Laplace error, and therefore achieve better accuracy. We prove the superiority of our solution by both theoretical analysis and extensive experiments on different standard real-life and synthetic datasets.

Table 4: KLD on four datasets (ϵ=0.01)

Datasets	AHP	NF	SF	PHP	GS
Location	0.467	2.588	4.948	1.076	3.803
Social Network	0.825	3.099	6.058	1.019	5.112
Search Log	0.189	5.584	5.340	0.86	7.046
NetTrace	1.229	6.926	6.841	2.280	8.187

Acknowledgement This research is partially supported by the grants from Natural Science Foundation of China (61379050, 91224008), National 863 High-tech Program (2013AA013204), Specialized Research Fund for the Doctoral Program of Higher Education(20130004130001), and RGC/GRF HKBU (210811).

References

[1] G. Acs, C. Castelluccia, and R. Chen. Differentially private histogram publishing through lossy compression. In *Proc. of ICDM*, pages 1–10, 2012.

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

[3] D. P. Doane. Aesthetic frequency classifications. *American Statistician*, 30:181–183, 1976.

[4] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. of TCC*, pages 265–284, 2006.

[5] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proc. of VLDB Endow*, 3(1):1021–1032, 2010.

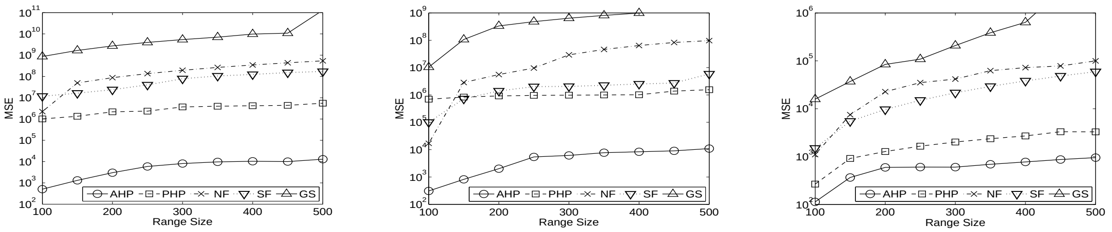
[6] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *Proc. of VLDB*, pages 275–286, 1998.

[7] G. Kellaris and S. Papadopoulos. Practical differential privacy via grouping and smoothing. *Proc. of VLDB Endow*, 6(5):301–312, 2013.

[8] J. Lei. Differentially private m-estimators. In *Proc. of NIPS*, pages 361–369, 2011.

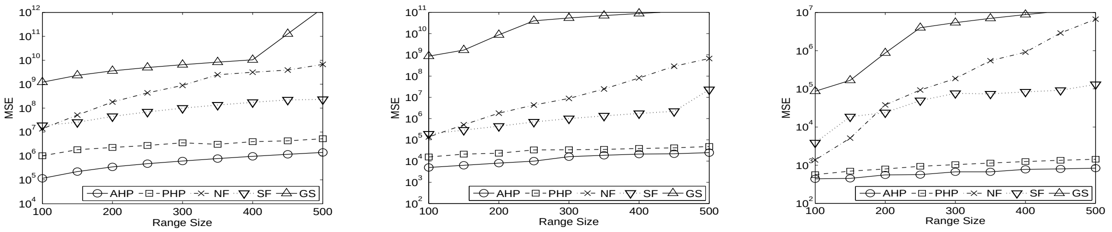
[9] C. Li, M. Hay, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Proc. of PODS*, pages 123–134, 2010.

[10] N. Li, W. H. Qardaji, and D. Su. On sampling, anonymization, and differential privacy or, k-



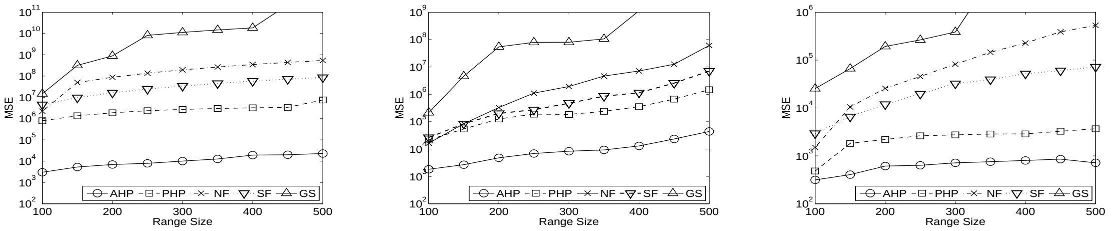
(a) $\epsilon = 0.01$ (b) $\epsilon = 0.1$ (c) $\epsilon = 1$

Figure 2: MSE on *Location* under different ϵ values. *y-axis* is in log-scale



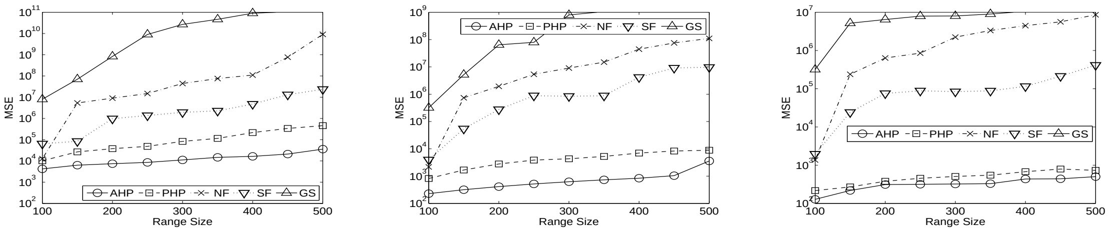
(a) $\epsilon = 0.01$ (b) $\epsilon = 0.1$ (c) $\epsilon = 1$

Figure 3: MSE on *Social Network* under different ϵ values. *y-axis* is in log-scale



(a) $\epsilon = 0.01$ (b) $\epsilon = 0.1$ (c) $\epsilon = 1$

Figure 4: MSE on *Search Log* under different ϵ values. *y-axis* is in log-scale



(a) $\epsilon = 0.01$ (b) $\epsilon = 0.1$ (c) $\epsilon = 1$

Figure 5: MSE on *NetTrace* under different ϵ values. *y-axis* is in log-scale

anonymization meets differentail privacy. In *Proc. of ASIACCS*, pages 32–33, 2012.

[11] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proc. of SIGMOD*, pages 19–30, 2009.

[12] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proc. of FOCS*, pages 94–103, 2007.

[13] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation end encryption. In *Proc. of SIGMOD*, pages 735–746, 2010.

[14] H. A. Sturges. The choice of a class interval. *Journal of the American Statistical Association*, 21:65–66, 1926.

[15] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transform. In *Proc. of ICDE*, pages 225–236, 2010.

[16] J. Xu, Z. Zhang, X. Xiao, and G. Yu. Differentially private histogram publicaiton. In *Proc. of ICDE*, pages 32–43, 2012.

[17] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. Low-rank mechanism: optimizing batch queries under differential privacy. *Proc. of VLDB Endow*, 5(11):1352–1363, 2012.