Data-Dependent Blowfish Private Algorithm for Histogram Queries

Akbota Anuarbek

1 Introduction

Differential privacy is well known for not only protecting records of individuals in databases from privacy breach, but also for providing a useful tool for measuring the level of privacy. However, under certain circumstances, namely when the sensitivity of the query is very high, differential privacy could be too strong and query answers released under differential privacy may be of a little usefulness due to large amounts of noise added to answers. As a solution to this problem X. He, A. Machanavajjhala, and B. Ding introduced a notion of *Blowfish Privacy* [4] that offers better trade-off between privacy and utility using policies. This is done by defining the set of discriminative pairs, which are values in the domain that adversary should not be able to distinguish between, whereas he is able to distinguish between pairs of values that are not in the set. By losening the definition of neighboring databases they were able to show that sensitivities of queries decrease, thus resulting in smaller errors in query answers. However, general methods for designing algorithms under Blowfish Privacy were not known until S. Haney, A. Machanavajjhala, and B. Ding [3] have proven a *Transformational Equivalence* theorem for data independent mechanisms and for cases, when a policy graph is a tree. The equivalence theorem states that for a mechanism $\mathcal{M}$ that satisfies (ϵ, G)-Blowfish privacy there exists a linear transformation of a workload and a database such that the existing method $\mathcal{M}$ satisfies ϵ-differential privacy for answering the transformed workload on the transformed database. Although applying transformational equivalence to Blowfish mechanisms results in substantial decrease in error compared to differentially private data-independent mechanisms in most cases, some differentially private data-dependent algorithms like DAWA [1] perform better than transformed mechanisms under some workloads, including the histogram workload. What is more, transformational equivalence does not hold for data-dependent mechanisms, when the policy graph is not a tree. This gives a rise to an interesting question of designing data-dependent algorithms for histogram queries under Blowfish Privacy framework.

This paper will first describe previous rearch on the topic of interest, and then continue by introducing some basic notation and theorems to be used in the rest of the paper. It will then describe the overall structure of the algorithm and proceed by introducing two algorithms for private partitioning. Finally, all the described methods will be evaluated on their performance on some real world dataset, and some interesting insights will be given along with the conclusion and future research ideas.

2 Related Work

Haney et al. [3] apply data-dependent algorithm DAWA on a transformed dataset for the case when the policy graph is a tree. Since transformational equivalence can not be applied for data-dependent algorithms under a general policy graph, they propose an approximation by embedding the graph G to its spanning tree G' and then applying transformational equivalence on G'. Formally, the approximate transfromational equivalence states that for any mechanism $\mathcal{M}$ that satisfies (ϵ, G')-Blowfish Privacy, $\mathcal{M}$ also satisfies $(l \cdot \epsilon, G)$-Blowfish Privacy, where l is the maximum distance between any two neighboring vertices of G in G'.

DAWA is a data- and workload-aware algorithm introduced by Li et al. [1]. They consider a histogram query and their algorithm is implemenyted in three steps. In the first step, the domain is privately partitioned into k buckets, such that within each bucket the dataset is approximately uniform. In the second step, counts of each bucket is privately estimated taking into consideration the workload, and finally counts of each value in the domain is reconstructed using the uniformity assumption in the last step.

3 Preliminaries

Consider a dataset D with a single attribute, whose domain is given by $v = \{v_1, ..., v_k\}$ with size k. The dataset can also be represented as a histogram vector $\mathbf{x} \in \mathbb{R}^k$, where each entry $\mathbf{x}[i]$ corresponds to the number of entries in D with the value equal to the i-th value in the domain $\mathbf{v}$. Finally define a workload $\mathbf{W}$ as a $q \times k$ matrix, where each row corresponds to a linear query, and such that $\mathbf{Wx}$ is an answer to the workload. In this paper we are interested in an identity workload $\mathbf{I}_k$, such that each entry of $\mathbf{I}_k\mathbf{x}$ corresponds to a count of some value value in the domain, which indeed gives a histogram.

Definition 3.1 (ϵ-differential privacy [2]) *A mechanism $\mathcal{M}$ satisfies ϵ-differential privacy if for any subset of outputs $S \subset range(\mathcal{M})$ and for any pair of datasets D and D' that differ in the presence/absence of a single entry the following is satisfied:*

$$Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(D') \in S]$$

Definition 3.2 ((ϵ, G)-Blowfish Privacy [4]) *Given a policy graph G, a mechanism $\mathcal{M}$ satisfies (ϵ, G)-Blowfish Privacy if for any subset of outputs $S \subset range(\mathcal{M})$ and for any pair of neighboring datasets under G, i.e. for $(D, D') \in \mathcal{N}(G)$ the following is satisfied:*

$$Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(D') \in S] \quad (1)$$

Definition 3.3 (Transf-l equivalence [3]) *For a policy graph G, there exists a transformation of the workload and database, $(\mathbf{W}, \mathbf{x}) \rightarrow (\mathbf{W}_G, \mathbf{x}_G)$ such that $\mathbf{Wx} = \mathbf{W}_G\mathbf{x}_G$, and a mechanism $\mathcal{M}$ is an (ϵ, G)-Blowfish private mechanism for answering workload $\mathbf{W}$ on input $\mathbf{x}$ iff $\mathcal{M}$ is also an ϵ-differentially private mechanism for answering $\mathbf{W}_G$ on x_G*

Transformational equivalence holds for any mechanism $\mathcal{M}$ under a tree-shaped policy graph G, and when policy graph has cycles the theorem holds only for data-independent mechanisms [3]. In particluar, the transformation is given by $\mathbf{x}_G = \mathbf{P}_G^{-1}\mathbf{x}$ and $\mathbf{W}_G = \mathbf{WP}_G$, where $\mathbf{P}_G$ is a vertex-edge incidence matrix. Its construction is described in detail in [3].

4 Algorithm Steps

The algorithm of interest is an (ϵ, G)-Blowfish private mechanism that takes the identity workload $\mathbf{I}_k$ and the dataset $\mathbf{x}$ as inputs and returns a noisy histogram of $\hat{\mathbf{x}}$. The algorithm runs in three steps described below. Steps 1 and 2 are (ϵ_1, G)- and (ϵ_2, G)-Blowfish private algorithms respectively, whereas the final step does not need any privacy budget being merely based on postprocessing. Then using the sequential composition theorem, the total privacy budget of the algorithm is $\epsilon = \epsilon_1 + \epsilon_2$.

Step 1: Private Partitioning

In the first step the algorithm takes as an input the dataset $\mathbf{x}$ and returns the partitioning of the domain into buckets. Similar to DAWA [1], the partitioning is based on grouping values in the domain that have similar number of counts. However, buckets do not have to be convex, i.e. values in a bucket do not have to be sequential. Although partitioning is based on noisy counts in $\mathbf{x}$, the only thing returned in this step is a partitioning without releasing noisy counts. In the next section we will discuss two (ϵ_1, G)-Blowfish private algorithms for partitioning.

Step 2: Private Estimation of Bucket Counts Given a partitioning of the domain into m buckets, define a new histogram of counts $\mathbf{x}'$ as a vector of size m, where each entry corresponds to the sum of value counts belonging to the same bucket. Also define the new policy graph G' with m vertices, where each vertex also corresponds to a bucket of the partitioning from step 1. G' is basically constructed from G by merging vertices that correspond to values that are grouped within the same bucket. A simple example is illustrated in figure 1. Let $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5]$ be a dataset with a policy graph G illustrated as the upper graph in figure 1. Suppose, x_2 is close in value to x_3, so that step 1 puts x_2 and x_3 inside the same bucket, while assigning all other counts to separate buckets. Then the new dataset is given by $\mathbf{x} = [x_1, x_2 + x_3, x_4, x_5]$ with a policy graph G'.

Figure 2 illustrates construction of G' for one of the common graphs a grid graph which is mostly used with geo-allocation data.

Theorem 4.1 *Answering a workload $\mathbf{I}_m$ on a dataset $\mathbf{x}'$ under (ϵ, G')-Blowfish privacy is equivalent to answering workload $\mathbf{W}_m$ on a dataset $\mathbf{x}$ under (ϵ, G'')-Blowfish privacy, where $\mathbf{W}_m$ is an $m \times k$ matrix with entries $w_{ij} = 1$ if j-th value belongs to bucket i, i.e. $v_j \in b_i$ and 0 otherwise.*

G'' can be constructed from G' by splitting supervertices, so that each splitted vertex will still preserve all the edges incident on the supervertex it was a part of. There is also an edge in G'' between two vertices belonging to the same supervertex in G'. G'' for the above example is given in figure 1.

Proof: It is straightforward to see that $\mathbf{W}_m\mathbf{x} = \mathbf{I}_m\mathbf{x}'$. Now let's consider two datasets

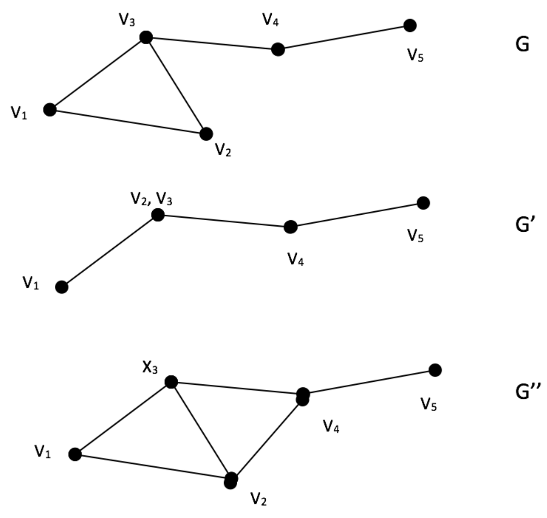


Figure 1

D_1 and D_2 that differ in the value of a single entry, i.e. $D_1 = D \cup \{v_1\}$ and $D_2 = D \cup \{v_2\}$. Suppose D_1 and D_2 are neighbors under G', thus there is an edge between two supervertices u_1 and u_2 each containing vertices v_1 and v_2 respectively. Then by construction of G'' there is an edge between v_1 and v_2 in G'', thus D_1 and D_2 are neighbors under policy graph G'' as well so that equation 1 is satisfied for them. Now suppose D_1 and D_2 are neighbors under G', and there is an edge between v_1 and v_2 in G''. Then there is either (1) an edge between two supervertices u_1 and u_2 each containing vertices v_1 and v_2 respectively or (2) both vertices v_1 and v_2 belong to the same supervertex. In case of (1) D_1 and D_2 are neighbors under G' and equation 1 is again satisfied. In case (2) since the count of the supervertex is the sum of of the counts of vertices belonging to it, the count of the supervertex will not be affected at all. In particular since the workload being considered returns counts of supervertices the outputs are uneffected under both graphs when changing v_1 to v_2 inside the same bucket.

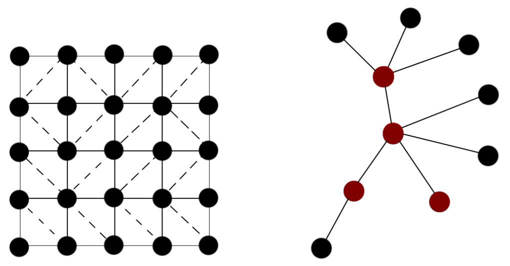


Figure 2: Grid graph

Lemma 4.1 *Answering a workload $\mathbf{I}_m$ on a dataset $\mathbf{x}'$ under (ϵ, G')-Blowfish privacy guarantees (ϵ, G)-Blowfish privacy for answering $\mathbf{W}_m$ on $\mathbf{x}$.*

This trivially follows from theorem 4.1. and the fact that (ϵ, G'')-Blowfish privacy for answering $\mathbf{W}_m$ on $\mathbf{x}$ implies (ϵ, G)-Blowfish privacy for answering $\mathbf{W}_m$ on $\mathbf{x}$.

Thus, the step 2 of the algorithm is implemented under (ϵ_2, G)-Blowfish privacy by using lemma 4.1 and answering a workload $\mathbf{I}_m$ on a partitioned dataset $\mathbf{x}'$ under (ϵ_2, G')-Blowfish privacy. And for estimating bucket counts, i.e. for answering $\mathbf{I}_m$ on $\mathbf{x}'$ under (ϵ_2, G')-Blowfish privacy we will use transformational equivalence theorem. More precisely, we will use laplace mechanism on transformed workload and dataset. Since laplace mechanism is data-independent and the data-dependent partitioning was implemented before applying the transformational equivalence theorem, our algorithm is valid for any type of policy graph.

Step 3: Uniform Expansion

Finally, given noisy counts of buckets, noisy counts of each value is reconstructed by assuming uniformity within each bucket. That is noisy count of each value is the noisy count of the bucket it belongs to over the size of the bucket.

5 Private Partitioning

Under the above algorithm each noisy count has two types of noise added to it. One is based on the noise added in order to privately estimate counts of each bucket, this noise decreases as the number of values in the bucket increases. The other is based on uniformity assumption, and depends on how close are the counts of values belonging to the same bucket. Thus, we seek to increase the number different domain values in each bucket, at the same time ensuring that counts within the same bucket are close enough.

5.1 Partitioning under the first step of DAWA

Thus, one intuitive thing to try is to partition the data under the first step of DAWA. The first step of DAWA is a differentially private algorithm that partitions data into sequential buckets. Utilizing the fact that $\epsilon/2$-differentially private algorithm guarantees (ϵ, G)-Blowfish privacy, we can privately partition our domain under the first step of DAWA using $\epsilon_1/2$ privacy budget.

5.2 Partitioning with Maximum Weighted Matching

The alternative method is reducing the partitioning problem to the *maximum weighted matching problem*. For that let's first define the *minimum cost partition problem*. Since we will be reducing it to the matching problem, the maximum size of buckets equals to 2. For each bucket b define its cost as $cost(b) = \sum_{i \in b} |x_i - \bar{x}_b| + \frac{1}{\epsilon_2}$, where x_i is the count for the domain value v_i and $\bar{x}_b = \sum_{i \in b} \frac{x_i}{|b|}$ is the average count of values in b. Now define B as a valid partition if each of the values in the domain belongs to exactly one buckets in B, and the cost of the partition B is the sum of the individual bucket costs $cost(B) = \sum_{b \in B} cost(b)$. The objective is to find the partition B with minimum cost $cost(B)$.

Now let's implement the reduction. Since we restrict bucket sizes to be no larger than 2, we can rewrite the cost of a bucket with two elements as $cost(b) = |x_i - \bar{x}_b| + |x_j - \bar{x}_b| + \frac{1}{\epsilon_2} = |x_i - x_j| + \frac{1}{\epsilon_2}$ and the cost of a bucket with a single element as $cost(b) = |x_i - \bar{x}_b| + \frac{1}{\epsilon_2} = \frac{1}{\epsilon_2}$. Now consider a complete graph C on k vertices, where each vertex corresponds to a value in the domain. For convenience call this edges primary edges. Given an edge (v_i, v_j) define a weight on that edge to be $w(v_i, v_j) = -(|x_i - x_j| + \frac{1}{\epsilon_2}) + 2K$, where K is some positive constant that insures that all the weights of the graph are positive. Now for each vertex add another secondary vertex, and connect each pair with an edge having a weight $-\frac{1}{\epsilon_2} + K$. This is illustrated in figure 3.

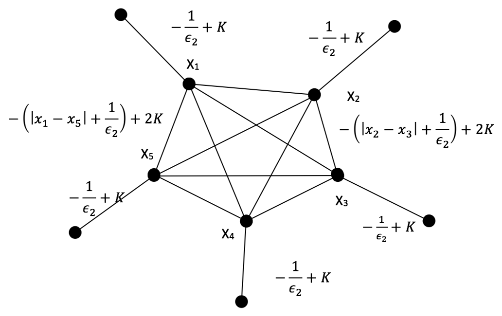

$-\frac{1}{\epsilon_2} + K$
$-\frac{1}{\epsilon_2} + K$
x_1
x_2
$-\left(|x_1 - x_5| + \frac{1}{\epsilon_2}\right) + 2K$
$-\left(|x_2 - x_3| + \frac{1}{\epsilon_2}\right) + 2K$
x_5
$-\frac{1}{\epsilon_2} + K$
x_3
$-\frac{1}{\epsilon_2} + K$
x_4
$-\frac{1}{\epsilon_2} + K$

Figure 3

Theorem 5.1 *Given the graph C defined as above, the maximum weighted matching in C finds the minimum cost partition in the original problem, where each edge incident on two primary vertices corresponds to a bucket with two elements and every edge incident on only one primary edge corresponds to a bucket with a single element corresponding to that vertex.*

Proof: Since every primary vertex corresponds to a value in the domain and each edge corresponds to a bucket, we claim that every primary vertex will be incident on exactly one edge of the maximum weighted matching. This is equivalent to saying that every domain value should belong to exactly one bucket. Our claim is indeed true because (1) no primary vertex will have more than one edge incident on it by the definition of matching, (2) all primary vertices will have at least one edge incident on them because all edge weights are positive and we are maximizing the sum of weights. Now that we have shown that the maximum weighted matching corresponds to a valid partitioning, we just need to show that this partitioning is the one with the minimum cost. This is indeed true because our weights in the graph are equal to the negative of costs plus some positive constant, thus the matching that maximizes the sum of weights is equivalent to the partitioning that minimizes the sum of costs. Also it is important to add a positive constant $2K$ to edges with 2 primary vertices and only K for edges with a single primary vertex in order not to make edges with a single primary vertex more attractive. There is an alternative way of reduction, where instead of adding a positive constant we add additional edges with weight 0 that connect every pair of secondary vertices. Then the problem is to find a maximum weight perfect matching and we do not have to worry about negative weights.

Privacy guarantee:

We showed that the maximum weighted matching in the graph constructed as above indeed finds the minimum cost partitioning. However, more importantly this algorithm has to run privately. For that add a noise $Z_j \sim laplace$ for the weight of each edge, we need to adjust K so that all weights are still positive.

Theorem 5.2 *The algorithm defined as above satisfies ϵ-differential privacy.*

We know that our algorithm finds the maximum weighted matching given noisy weights on edges. Let M be the maximum weight matching given the current instance of the dataset. Then it satisfies the following:

$$\sum_{j \in M} weight(\mathbf{x}, e_j) + z_j >$$

$$\max_{M' \in \mathcal{M} - M} \left(\sum_{k \in M'} weight(\mathbf{x}, e_k) + z_k \right) \quad (2)$$

where $\mathcal{M}$ is the set of all the possible matchings in C and e_j corresponds to some edge in M. Now suppose we add or remove one record from the dataset. Then the weight of exactly one edge changes by one in M, call this edge e_i. Let $\mathcal{M}^+ = \{M | M \in \mathcal{M} \text{ and } e_i \in M\}$ and $\mathcal{M}^- = \mathcal{M} - \mathcal{M}^+$. We claim that the matching M will be selected if (1) it has the highest noisy weight among all the matchings in $\mathcal{M}^+$, and (2) if its noisy weight is higher than noisy weights of all matchings in $\mathcal{M}^-$.

Since all matchings in $\mathcal{M}$ include the edge e_i whether (1) holds does not depend on the outcome of Z_i. Let

$$\phi(\mathbf{x}, \mathbf{z}^{-i}) = \sum_{j \in M - \{i\}} weight(\mathbf{x}, e_j) + z_j >$$

$$\max_{M' \in \mathcal{M}^+ - \{M\}} \left(\sum_{k \in M' - \{i\}} weight(\mathbf{x}, e_j) + z_k \right)$$

be a predicate that is true if the assignment of $\mathbf{z}^{-i}$ makes M the maximum weight matching in $\mathcal{M}^+$ and false otherwise. Since neighboring datasets $\mathbf{x_1}$ and $\mathbf{x_2}$ differ only in the weight of one edge e_i, $\phi(\mathbf{x_1}, \mathbf{z}^{-i}) = \phi(\mathbf{x_2}, \mathbf{z}^{-i})$ for all possible $\mathbf{z}^{-i}$.

Now let ψ be a predicate that is true if (b) holds, thus

$$\psi(\mathbf{x}, \mathbf{z}) = \sum_{j \in M} weight(\mathbf{x}, e_j) + z_j >$$

$$\max_{M' \in \mathcal{M}^-} \left(\sum_{k \in M'} weight(\mathbf{x}, e_k) + z_k \right) =$$

$$z_i > Z(\mathbf{x}, \mathbf{z^{-i}})$$

where

$$Z(\mathbf{x}, \mathbf{z^{-i}}) = \max_{M' \in \mathcal{M}^-} \left(\sum_{k \in M'} weight(\mathbf{x}, e_k) + z_k \right)$$

$$- \sum_{j \in M} weight(\mathbf{x}, e_j) + \sum_{l \in M - \{i\}} z_l$$

For neighboring datasets $\mathbf{x}_1$ and $\mathbf{x}_2$, $Z(\mathbf{x_1}, \mathbf{z^{-i}}) \leq Z(\mathbf{x_2}, \mathbf{z^{-i}}) + 2$, because the weights of matchings in $\mathcal{M}^-$ can increase by at most 1 and the weight of M can decrease by 1.

Then,

$$P(A(\mathbf{x}) = M) = P(\phi(\mathbf{x}, \mathbf{Z}^{-i}) \wedge \psi(\mathbf{x}, \mathbf{Z})) =$$

$$\int \mathbf{I}[\phi(\mathbf{x}, \mathbf{z}^{-i}) \wedge \psi(\mathbf{x}, \mathbf{z})] f_{\mathbf{z}}(\mathbf{z}) d\mathbf{z} =$$

$$\int \mathbf{I}[\phi(\mathbf{x}, \mathbf{z}^{-i})] f_{\mathbf{z}^{-i}}(\mathbf{z}^{-i}) (\int \mathbf{I}[\psi(\mathbf{x}, \mathbf{z})] f_{Z_i}(z_i) dz_i) d\mathbf{z}^{-i}$$

$$= \int \mathbf{I}[\phi(\mathbf{x}, \mathbf{z}^{-i})] f_{\mathbf{z}^{-i}}(\mathbf{z}^{-i}) P(Z_i > Z(\mathbf{x}, \mathbf{z}^{-i})) d\mathbf{z}^{-i}$$

Now for neighbors $\mathbf{x}_1$ and $\mathbf{x}_2$ and any $\mathbf{z}^{-i}$

$$P(Z_i > Z(\mathbf{x_1}, \mathbf{z}^{-i})) \geq P(Z_i > Z(\mathbf{x_2}, \mathbf{z}^{-i}) + 2)$$

$$\geq e^{-2/\lambda} P(Z_i > Z(\mathbf{x_2}, \mathbf{z}^{-i}))$$

by the properties of laplace distribution, where Z is laplace random variable with scale λ.

Now combining all of the above,

$$P(A(\mathbf{x}_1) = M) =$$

$$\int \mathbf{I}[\phi(\mathbf{x}_1, \mathbf{z}^{-i})] f_{\mathbf{z}^{-i}}(\mathbf{z}^{-i}) P(Z_i > Z(\mathbf{x}_1, \mathbf{z}^{-i})) d\mathbf{z}^{-i} \geq$$

$$\int \mathbf{I}[\phi(\mathbf{x}_2, \mathbf{z}^{-i})] f_{\mathbf{z}^{-i}}(\mathbf{z}^{-i}) e^{-2/\lambda} P(Z_i > Z(\mathbf{x_2}, \mathbf{z}^{-i})) d\mathbf{z}^{-i}$$

$$= e^{-2/\lambda} P(A(x_2) = M) = e^{-\epsilon} P(A(x_2) = M)$$

where $\lambda = 2/\epsilon$.

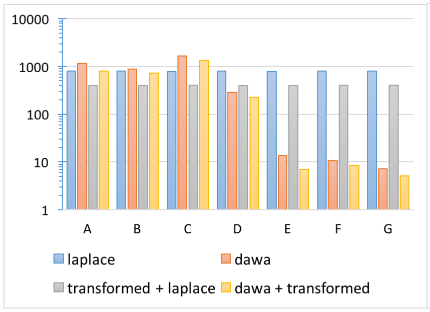
6 Experiments and Results

In this section we implement two algorithms described above together with other famous algorithms on some real and simulated datasets to empirically evaluate and compare their performances. Performances of algorithms are evaluated based on their average mean squared error per query over 5 trials. The real datasets used for evaluation are the same one-dimensional datasets that were used in [3]. All datasets have the same domain size of 4096, but differ in their scale and the percentage of 0 counts. We compare (ϵ, G)-Blowfish private mechanisms against $\epsilon/2$-differentially private algorithms. The policy graph G we are considering here is a line graph. The algorithms were evaluated given different values of $\epsilon \in \{0.001, 0.01, 0.1\}$

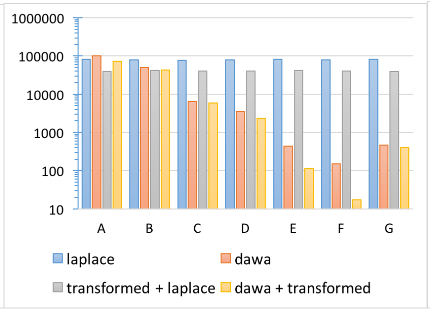
First evaluation was done on the 7 real world datasets described above and implemented for 4 different algorithms. The results are given in figure 4. Here, Laplace and DAWA are straigtforward and stand for $\epsilon/2$-differentially private mechanisms, whereas Transformed+Laplace is an (ϵ, G)-Blowfish private mechanism implemented by applying laplace on a transformed dataset. Finally, DAWA + Transformed is the first method discussed in the paper, it is an (ϵ, G)-Blowfish private mechanism.

Results and explanations: As is illustrated in the figure DAWA + Transformed performs very well, and always has a lower error than DAWA. This is a nice result since previous data-dependent Blowfish algorithms did not perform well under the histogram workload. One interesting thing to note is how the tradeoff between sizes of ϵ_1 and ϵ_2 affects performances of DAWA and DAWA + Transformed. It seems like DAWA performs better if $\epsilon_1 = 0.25\epsilon$, i.e. if only 25% of the privacy budget is utilized for partitioning. By contrast, DAWA + Transformed performs better for $\epsilon_1 = 0.5\epsilon$. However, overall DAWA + Transformed with $\epsilon_1 = 0.5\epsilon$ still outperforms DAWA with $\epsilon_1 = 0.25\epsilon$, we saw this in figure 4 as these values of ϵ_1 were used for the experiments. The possible reason behind this could be the fact that the second step of DAWA + Transformed does not need much privacy budget compared to the second step of DAWA thanks to the transformational equivalence theorem. Thus, DAWA + Transformed can possibly afford to move some of its budget for partitioning. And the reason why DAWA + Transformed performs worse for $\epsilon_1 = 0.25\epsilon$ could possibly be because it does not efficiently acount for the second step while doing the partitioning in the step 1. This is because the first step of DAWA does partitioning assuming that the second step will be ϵ_2-differentially private algorithm and adds average noise to each bucket based on that. However, since the second step of the algorithm is a Blowfish private mechanism. Tuning this parameter is an interesting question for future experiments.

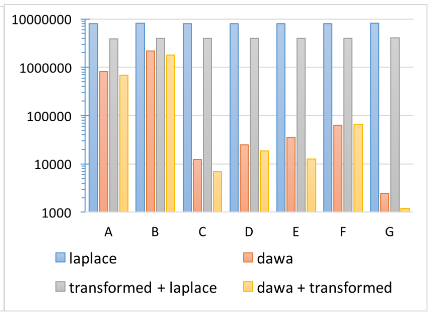
The second algorithm based on matching was evaluated on half-domain of the Income dataset and compared against the four algorithms discussed earlier. The algorithm was also tested on another simulated very high frequecy dataset. Figure 5 illustrates average mean squared errors of all 5 algorithms. As the average mean squared errors indicate, Matching+Transformed does not perform very well. On a real dataset it is only better than laplace, and on the simulated dataset it performs better than 3 algorithms, but still can not perform better than Transformed+Laplace. The main reason behind such behavior is the limit on the bucket size. This illustrates the importance of grouping values in buckets with large sizes in order to bring down the noise.



(a) $\epsilon = 0.1$

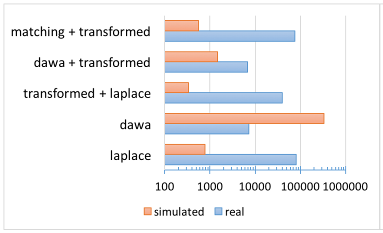


(b) $\epsilon = 0.01$

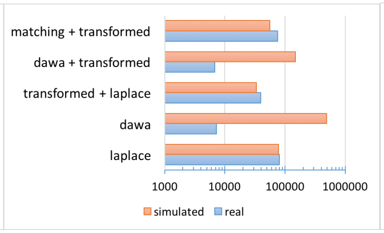


(c) $\epsilon = 0.001$

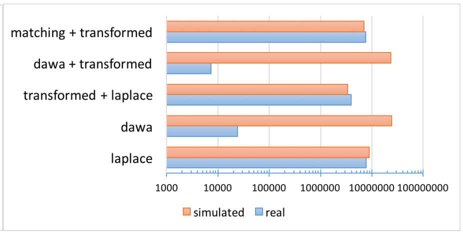
Figure 4: Histogram workload with G_k^1



(a) $\epsilon = 0.1$



(b) $\epsilon = 0.01$



(c) $\epsilon = 0.001$

Figure 5: Histogram workload with G_k^1

7 Conclusion

In this paper we considered two data-dependent Blowfish Private mechanisms. Both of them are based on first privately partitioning the data, creating a new policy grap, and then applying transformational equivalence theorem on new data and a policy graph to estimate bucket counts, and finally applying uniform expansion. By contrast to data-dependent Blowfish methods considered in earlier works, first partitioning and only after that applying transformational equivalence has a number of benefits. First, the algorithm can be applied to data with any policy graph. Secondly, the transformed data is usually less sparse, which makes partitioning less efficient. As expected our first algorithm performed very well. However, the performance of the second algorithm was poor due to the limit on the bucket size. On interesting next step could be applying the second algorithm sequentially, because one advantage of the second algorithm over the first is that it supports non-convex buckets.

References

[1] G. M. C. Li, M. Hay and Y. Wang. A data- and workload-aware algorithm for range queries under differential privacy. *VLDB*, 2014.

[2] K. C.Dwork, F.McSherry and A.Smith. Calibrating noise to sensitivity in private data analysis. *TCC*, 2006.

[3] A. M. S. Haney and B. Ding. Design of policy-aware differentially private algorithms. *VLDB*, 2015.

[4] A. M. X. He and B. Ding. Blowfish privacy: Tuning privacy-utility trade-offs using policies. *SIGMOD*, 2014.