Spatial Statistic Data Release Based on Differential Privacy

Sujin Cai[1,2], Xin Lyu[1*] and Duohan Ban[1]
[1] College of Computer and Information, HoHai University
Nanjing, China
[e-mail: lvxin.gs@163.com]
[2]School of Information Engineering, Yancheng Teachers University
Yancheng, China
[e-mail: sujin_cai@126.com]
*Corresponding author: Xin Lyu

Received October 15, 2018; revised January 4, 2019; accepted April 17, 2019; published October 31, 2019

Abstract

With the continuous development of LBS (Location Based Service) applications, privacy protection has become an urgent problem to be solved. Differential privacy technology is based on strict mathematical theory that provides strong privacy guarantees where it supposes that the attacker has the worst-case background knowledge and that knowledge has been applied to different research directions such as data query, release, and mining. The difficulty of this research is how to ensure data availability while protecting privacy. Spatial multidimensional data are usually released by partitioning the domain into disjointed subsets, then generating a hierarchical index. The traditional data-dependent partition methods need to allocate a part of the privacy budgets for the partitioning process and split the budget among all the steps, which is inefficient. To address such issues, a novel two-step partition algorithm is proposed. First, we partition the original dataset into fixed grids, inject noise and synthesize a dataset according to the noisy count. Second, we perform IH-Tree (Improved H-Tree) partition on the synthetic dataset and use the resulting partition keys to split the original dataset. The algorithm can save the privacy budget allocated to the partitioning process and obtain a more accurate release. The algorithm has been tested on three real-world datasets and compares the accuracy with the state-of-the-art algorithms. The experimental results show that the relative errors of the range query are considerably reduced, especially on the large scale dataset.

Keywords: LBS, Differential Privacy, Data Partitioning, Data Release, Range Query

1. Introduction

In recent years, the continuous development of communication and LBS technology has brought great convenience to people's lives. The living facilities such as shopping malls and restaurants, etc., can be found through LBS applications. A large amount of collected location data can be used by these applications, and potential business value is obtained from mining and analyzing. The users' privacy will inevitably be leaked if the data owner releases the actual data directly. Therefore, how to preserve the privacy of released data is an urgent problem that needs to be resolved. The existing privacy protection methods of data release mostly make use of anonymity and generalization techniques, such as k-anonymity [1], l-diversity [2], and t-closeness [3], all of which require an assumption of the background knowledge of the attackers. However, in the big data environment, there are so many channels for attackers to obtain information that it is difficult to confirm their background knowledge accurately. Differential privacy [4][5] is one of the strongest privacy guarantees that is based on a solid theoretical foundation for mathematics, and differential privacy assumes the worst-case of the attacker's background knowledge, that is, the attacker possesses the whole database except one tuple, but it is indistinguishable on the output whether the tuple is in the dataset or not. The main methods of differential privacy are to add noise to the query or release results, but it would make the data worthless if too much noise was added, and privacy guarantees could not be provided if an excessively small amount of noise was added, so the research key point is how to balance the preservation of privacy and data availability.

For some certain uses, the location service providers or government agencies do not need to know the exact location of the users, they need only to obtain the statistical results, i.e., how many people are there on a certain range at a certain time, that is, spatial statistics [6][7]. For example, the government departments analyze the traffic volume of a certain section of the road and formulate corresponding policies to regulate this traffic. Spatial data are multidimensional and can be released by partitioning the region and generating an index tree. The partitioning methods can be classified into two categories: data-dependent partition (such as the KD-tree [8]) and data-independent partition (such as the quad-tree). A data-dependent partition needs to divide the space based on the distribution of data such as choosing the median as the splitting point. However, the process of choosing the median will lead to privacy leakage. So we should allocate a part of the private budget to protect the process, continuing to split the budget and assigning it to each layer of the index tree. When the tree is high or the fan out is large, too much noise will be introduced, making the results inaccurate. In general, the author sets the height of the tree empirically. The other partitioning method is data-independent partition where the partitioning is based on the space range and does not refer specifically to the basic data. Data-independent is efficient and accurate when the data distribute uniformly, but the results are not ideal if datasets distribute unevenly.

This paper presents a novel algorithm to solve the current problems. The main contributions are as follows: A new data-dependent partition is made on a synthetic dataset according to the noisy counts in the grid cells. The partition granularity is computed by minimizing the sum of noise error and non-uniformity error. We propose a post-processing method that can reduce the relative error significantly by applying consistent constraint on the noisy count.

The rest of our paper is organized as follows: Section 2 reviews the related work. Preliminaries are introduced in Section 3. We present our differentially private partition strategy and the post-processing in Section 4. We provide the security analysis in Section 5. The experiments and results are presented in Section 6. We give the final conclusions in Section 7.

2. Related Work

The general methods of release for statistical data always involve the partition of the dataset and building the index structures according to the partition results, then releasing the index structure. In recent years, the privacy preserving of data releasing has received considerable attention. Researchers have proposed some influential release methods based on differential privacy [9-14,16,19-23]. We briefly review the relevant work here and discuss the differences between our work and existing work.

A few researchers have studied data-dependent partition with differential privacy [9-12]. Inan [9] proposed the Adaptive-KD tree algorithm, which can be used to partition both numerical and nonnumerical attributes. The mean is used to replace the traditional median as the splitting points when the numerical attribute is partitioned by KD-tree. However, [10] points out that the medians selected by this method are not accurate, especially in the skew data. Hence, they proposed the KD-Standard algorithm, which selects the private median using the exponential mechanism [15] that preserves the privacy of choosing the true median. They also proposed a geometric budgeting strategy that increased the privacy budget geometrically from root to the leaf nodes that will increase the accuracy of the released data. Hien To [11] built a two-level data-dependent tree, called the h-tree. As the height of the tree is low, the budget allocated to each level only needs to split twice. However, the fan out of the tree is quite large, and the budget needs to continue splitting for each leaf node. The DiffPart [12] algorithm proposed by Chen splits the data using the Taxonomy Tree that is based on generalization. This algorithm is suitable for set-valued data and supports top-k frequent pattern mining. However, this algorithm only supports count queries, and it does not consider the semantic association between different items, thus leading to the low availability of released data.

Several researchers [13,16] have studied the mechanism for data-independence. Qardaji [13] proposed the UG and AG algorithms. The UG algorithm uses uniform grids to partition the dataset, and the AG algorithm employs a two-layer partitioning strategy. The first layer is coarse-grained, adding noise to each cell, then continuing fine-grain partitioning if the noisy count of grid cells exceeds the threshold. The algorithm is data-independent partitioning thus has high efficiency, but it can not be partitioned heuristically according to the actual distribution of data, and has poor results in skewed data. The PrivTree [16] partitions the spatial domain using the complete Quad-Tree, then publishes the noisy count and the domain information of nodes. This algorithm adopts hierarchical decomposition but does not depend on the predefined tree height. The algorithm computes a biased count for nodes with a decaying factor, and then uses a constant amount of noise to judge whether to continue partitioning a node, as well as the sparse vector technique (SVT) [17][18] that is used to calculate the partition threshold.

The rest of the researchers [14,19,20] have studied hybrid partitions or other conditions. The DPCube [14] algorithm proposed by Xiao has two steps. First, the algorithm imposes a fixed grid over the original dataset, then generates a synthetic dataset based on the noisy

count of the cell. Second, the algorithm partitions the synthetic dataset using an innovative KD-Tree. Then, the resulting partitioning keys are used to split the original dataset. A new metric is proposed to determine whether the nodes that will minimize the non-uniformity error are close to uniform. Maryam [19] provided a method that focuses on counting planar spatial regions where the users visited most frequently. They leveraged the Euler characteristic with differential privacy for the first time to address the problem of duplicate counting when a planar body spaned multiple grid cells, and a novel constrained inference uses the least absolute deviations to increase the utility of the data. However, they used fixed grids to partition the region instead of adaptive partitioning, according to the dataset distributions. X. Zhang [20] proposed a three-layer adaptive grid decomposition strategy, called the STAG. The algorithm sampled the spatial data as the object of partitioning, and the data are decomposed by fixed grids. Some of the cells whose counts are over the threshold will continue to be split, and some of the cells whose counts are below the threshold will be merged.

To summarize the existing work, data-independent partition does not rely on the distribution of the data. Data-independent is efficient but has relatively larger errors when the data are not evenly distributed. In data-dependent partitioning, the choice of tree height determines when to stop splitting. However, this important parameter is always selected by experience. To address the above decomposition problems, we built a two-level tree that can be regarded as a data-dependent grid partition. The grid size is determined by minimizing the sum of the noise error and the non-uniformity error. This method is more accurate than setting tree height or fan out by experience. Traditional grid-based partitioning is data-independent and only obtains accurate results from a uniformly distributed database. Nevertheless, almost all real-life datasets are distributed unevenly. Our algorithm is data-dependent partitioning and chooses the private median as the splitting point depending on the actual data distribution. It combines the advantages of both grid and tree partition strategy, and has satisfactory results on the real world datasets.

3. Preliminaries

3.1 Differential Privacy

Definition 1 (ε-Differential privacy). Any neighboring datasets D and D' that have the same data structure and have only one record difference between them, that is, $\left|D'\Delta D\right|\le 1$. Given a randomized algorithm A, $Range(A)$ is the value range of A, if the output $O\left(O\in Range(A)\right)$ of algorithm A in datasets D and D' satisfies the inequality $\Pr\left[A(D)=O\right]\le e^{\varepsilon}\times\Pr\left[A(D')=O\right]$, then A satisfies ε - Differential privacy.

$\Pr[\cdot]$ denotes the probability of a user's privacy breach. Parameter ε is the private budget that specifies the degree of privacy protection. The algorithm A using smaller ε will obtain a higher degree of privacy protection.

Definition 2 (Global sensitivity). For a query function $f:D\to R^d$ and any neighboring datasets D and D', the global sensitivity of function f is $\Delta f=\max\limits_{D,D'}\left\|f(D)-f(D')\right\|_p$. Where R is the real number space mapped by dataset D, d denotes the query dimension of function f, and p is used to measure the norm distances of Δf, and generally, $p=1$.

Definition 3 (Laplace mechanism). Let $f: D \to R^d$ denotes a query function over a dataset *D*. Its global sensitivity is Δf, and a random algorithm *A* satisfies the ε-differential private if its output is $A(D) = f(D) + Lap(\Delta f / \varepsilon)$, where $Lap(\Delta f / \varepsilon)$ is a random variable sampled from the Laplace distribution, the amount of noise is proportional to Δf and inversely proportional to the private budget ε. The ε is smaller, the larger noise will be injected and the privacy will be protected more strictly, vice versa.

Sometimes, a preserving privacy algorithm will apply differential privacy more than once, or the analyses will operate on the disjoint subsets of data. On these occasions, the privacy guarantee depends on the sequential and parallel composition. The private budget can reasonably be allocated through the two theorems below, making the entire algorithm achieve ε - Differential privacy protection.

Theorem 1. Sequential Composition. Let $A_i (1 \le i \le n)$ be a set of random algorithms, each providing $\varepsilon_i (1 \le i \le n)$ - differential privacy. Then, the sequence of all algorithms satisfies $\sum_{i=1}^{n} \varepsilon_i$ - Differential privacy.

Theorem 2. Parallel Composition. If $D_i (1 \le i \le n)$ are the disjointed subsets of the original dataset D and $A_i (1 \le i \le n)$ is a set of random algorithms, each provides $\varepsilon_i (1 \le i \le n)$ -differential privacy for each D_i. Then the sequence of these algorithms satisfies $Max(\varepsilon_i)$ -Differential privacy.

3.2 Range Query

Definition 4 (Range Query). Given a spatial range *R* and a spatial dataset *D* of the moving objects at time *t*, the range query *Q* on the dataset is represented as: $Q(D) = \{x | x \in D, x \in range(R)\}$, where the $range(R)$ denotes the moving objects in the spatial range *R*.

3.3 Availability Metrics

For released data, it is necessary to improve the availability of data while protecting the privacy of the users. At present, the metrics of data availability mainly include relative error, absolute error, Euclidean distance, etc. In this paper, we consider the relative error as the measure of availability. For a range query *Q*, we use $Q(D)$ to denote the actual answer of Q on original dataset D, and $Q(D')$ indicates the answer of the same query on the released dataset D', and the formalized definition of relative error is as follows:

$$Error = \frac{|Q(D') - Q(D)|}{\max(Q(D), s)} \tag{1}$$

where *s* is a threshold that avoids the denominator becoming zero.

4. DPIH Partition

In the traditional data-dependent partitioning, the private budget should be divided into two parts. One part is used to protect the partitioning process, such as median selection, and the other part is used to add noise to the resulting count. The budget allocated to protect the partitioning can be represented as: $\varepsilon^m = \varepsilon / 2$, we called the median budget. Then, the total median budget needs to be divided by the tree height *h*, thus, the median budget for each level of the tree is: $\varepsilon^m / h = \varepsilon / 2h$. As expounded on Definition 3, the private budget is smaller, the added noise will be larger. When the tree height is high, a large amount of noise will be added. Obviously, the added noise will affect the accuracy of choosing the median and cause the released data to be worthless.

The H-tree [11] is a two-level tree that uses data-dependent partition. Since the height of the tree is low, the budget allocated to each level only need to split twice. However, the total budget should be cut into two parts: one part is to protect the node counts, called the count budget ε^c, and the other part is to protect the splitting process, called the median budget ε^m. Then, the count budget should still split into two parts for two levels of the tree, i.e. $\varepsilon^c = \varepsilon_1^c + \varepsilon_2^c$, and the median budget needs to continue splitting $2\log_2 m$ times for choosing the median (*m* is the partitioning granularity), that is, $\varepsilon^m = \log_2 m\left(\varepsilon_1^m + \varepsilon_2^m\right)$. Although the number of H-tree budget splits is less than KD-tree, it will still affect the accuracy of the partition results and the availability of published data when *m* is quite large. We improved the H-Tree and propose the DPIH-tree partition algorithm, injecting a constant amount of noise into the original dataset and generating a synthetic dataset according to the noisy count. Since the partition does not perform on the original data, there is no need to allocate an additional private budget to protect the partitioning process Hence, more budget can be used for node counts. The [10] points out that the query accuracy will be significantly improved if more budget is allocated for node count than for medians. The experiments show that our algorithm performs particularly better than the state-of-the-art methods on million-level datasets.

We provide an overview of our partitioning strategy: First, DPIH decomposes the entire space domain with fixed grids, injects noise into each grid cell and synthesizes a dataset according to the noisy count. Second, DPIH partitions the synthetic dataset using the IH-tree (Improved H-Tree), then partitions the original dataset using the splitting points that are obtained from the former step.

The details of the DPIH (Differential Privacy Improved H-Tree) partitioning algorithm are as follows: 1) Partition the original dataset using fixed uniform grids, and the grid size is β; 2) Add Laplace noise to each grid cell using private budget $\alpha\varepsilon$ and synthesize a new dataset based on the noisy count of each cell; 3) Partition the synthetic dataset with Algorithm 2 (IH-Tree partitioning algorithm); 4) Partition the original dataset using the splitting keys that returned from step 3; 5) Add noise using the rest of the private budget $(1-\alpha)\varepsilon$, releasing the noisy count of each cell.

In Algorithm 2 (IH-Tree partition algorithm), it is necessary to determine which dimension will be splitted first. We calculated the variance of the spatial points in each dimension respectively and selected the larger one to split first, because the larger variance means the data points distribute in a more dispersed manner in this dimension, and it will have better resolution to partition in this dimension. Then, choose the medians to partition

the spatial domain in the first dimension until *m* data blocks are obtained. Further split each block into *m* sub-blocks in the second dimension in the same way. There is a naive method to select *m-1* sequential medians to generate *m* partitions on each dimension, but obviously, it is inefficient if *m* is quite large. In our algorithm, a recursive manner is used to partition the dataset, that is, the synthetic dataset D_c is partitioned into D_{c_1} and D_{c_2} with the median, then the subsets D_{c_1} and D_{c_2} are recursively partitioned until return *m* blocks. However, the value of *m* is not always the power of 2, hence, we iterative split the domain into $2^{\lfloor \log_2 m \rfloor}$ blocks, calculate the variance of these blocks in the first dimension, sort them in descending order and choose the first $m-2^{\lfloor \log_2 m \rfloor}$ blocks to continue partitioning. Therefore, we partition the domain into *m* blocks in the first dimension, then we split each block in the second dimension similarly. Finally, we obtain $m \times m$ cells.

Algorithm 1. DPIH partitioning algorithm:

Input: Size of grids β, Private budget ε, dataset *D*

Output: Release the domain information and noisy count of each leaf node

1. Partition the original dataset into β uniform grid cells;
2. Add Laplace noise with budget $\alpha\varepsilon$ for each cell, synthesize a new dataset D_c based on the noisy count;
3. Partition D_c by **Algorithm 2** (IH-Tree Partitioning Algorithm);
4. Partition the original dataset *D* according to the splitting keys which obtained from the step 3;
5. Add noise to each cell using the rest budget $(1-\alpha)\varepsilon$ and release the domain information and noisy count of each leaf node.

Algorithm 2. IH-Tree partitioning algorithm:

Input: dataset D_c, granularity size *m*

Output: partition keys

1. Calculate the variance of the spatial points for each dimensions in the original dataset *D* , and select the larger one to divide first;
2. Obtain *m* data blocks by dividing the first dimension;

(Partition the dataset into D_{c_1} and D_{c_2} by the median of the first dimension, then partition D_{c_1} and D_{c_2} iteratively until *m* blocks are obtained)

3. For each *m* blocks:

Get *m* sub-blocks in the second dimension using the method of step 2;

4. Return the partition keys.

4.1 Synthetic Dataset Construction

In Algorithm 1, we slice the original dataset into β grid cells, add noise into each cell using the private budget $\alpha\varepsilon$, and synthesize a new dataset based on the noisy count. We refer to the method of synthesizing a dataset proposed in [9]: If the noisy count is *s* more than the actual count, add *s* data points randomly in that cell; if the noisy count is *s* less than the true count, delete *s* data points randomly; if the noisy count is negative, modify the noisy count to 0. Algorithm2 (IH-Tree partition) is performed on this synthetic dataset. Since the division is not in the original dataset, it can save the median budget that needs to be allocated

to protect the partitioning process in traditional methods.

4.2 Granularity

The first step of the Algorithm 1 needs to partition the original dataset into β grid cells and add Laplace noise based on the $\alpha\varepsilon$ budget for each cell. If β is excessively large, too much noise will be introduced, affecting the availability of the released data; If β is undersize, the perturbation error will be reduced, but the non-uniformity errors will increase, as well as, exposing the distribution of the actual data, and the privacy guarantee cannot be enforced. Unfortunately, [24] proves that the optimal partitioning in multi-dimensional histograms is NP-hard, that is, the optimal size of grids does not exist. In our experiments, we set β to a fixed value of 100, i.e. 10*10 grid cells cover the domain.

In Algorithm 2, synthetic dataset D_c is partitioned into $m \times m$ blocks. We prefer to compute the minimum query estimation error proposed in [13] to determine the size of the partitioning granularity. In the experimental part, we will give a more reasonable proposal based on the experimental results.

The query estimation error has two sources: perturbation error and non-uniform error. Perturbation Error (PE) is the error caused by adding Laplace noise into each cell, therefore, the perturbation error is the difference between noisy count and actual count. The non-Uniformity Error (NE) is generated only when the query rectangle partially intersects the cells. We assume that the data points in these cells are uniformly distributed, and this estimation will produce non-uniformity error.

$$Error(Q) = PE(Q) + NE(Q) \tag{2}$$

Theorem 3. To minimize the query estimation error while the released dataset satisfies ε – differential privacy, the partitioning granularity of DPIH tree is $m = \sqrt{N\varepsilon / c}$, where N is the number of points of the synthetic dataset D_c, ε is the private budget of the algorithm, and c is a constant.

Proof: In the process of releasing two-dimensional spatial statistics, the global sensitivity of the counting query is 1, and the private budget of noise is ε. In the other words, the added noise is a random sample that follows the Laplace distribution $Lap(1/\varepsilon)$ and has a deviation of $2/\varepsilon^2$. Assuming that r is the ratio of the query rectangle to the domain area, then, there are approximately rm^2 cells that are included in the query. Thus, the perturbation error in the query rectangle is $\sqrt{rm^2 \times \left(2/\varepsilon^2\right)} = m\sqrt{2r}/\varepsilon$. The non-uniformity error is only related to the cells partially intersecting the query rectangle, and the non-uniformity error is proportional to the number of data points falling on the cells that are at the edge of the query rectangle. Since the ratio of the query area to the domain is r, the lengths of query rectangle are proportional to $\sqrt{r}$ of the spatial domain length. Thus, the edge of the query contains $m\sqrt{r}$ cells, that is, approximately $m\sqrt{r} \times \left(N/m^2\right) = \sqrt{r} \times \left(N/m\right)$ data points fall on the edge of the query. Assuming that the non-uniformity error, on average, is proportional to the total density of data points in the query border, then the non-uniformity error is $N\sqrt{r}/c_0 m$ for some constant c_0. To

minimize the sum of these two errors, namely, $\min\left(\left(N\sqrt{r}/c_0 m\right)+\left(m\sqrt{2r}/\varepsilon\right)\right)$, we should set $m=\sqrt{N\varepsilon/c}$, where $c=\sqrt{2}c_0$. □

EXAMPLE 4.2. Given m=5, we choose the longitude as the first partitioning dimension, and the medians are selected as the splitting points. Then, the domain is split into five blocks, i.e., $C1-C5$. The smaller interval means the data points in this region are denser. Furthermore, we split each $C1-C5$ into five partitions on the latitude using the same method as above. Finally, we obtain 25 cells. The partition effect of the above example is shown in **Fig. 1**. The red rectangle indicates a query range.

Fig. 2 demonstrates the index structure of this DPIH-tree. The root of the tree represents the whole domain, and includes five partitions that correspond to $C1-C5$. The second level of the tree is the further partitioning of each $C1-C5$. The height of the tree is 2, and the fan out of the tree is 5. The red parts represent the nodes that intersect the query box.

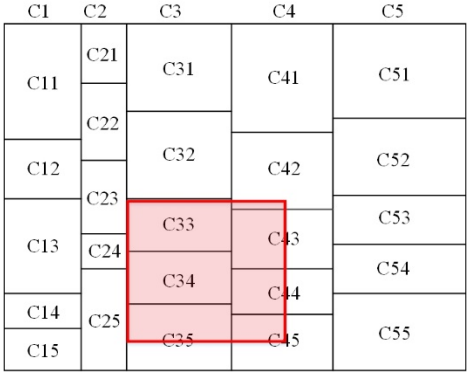


Fig. 1. Division effect diagram

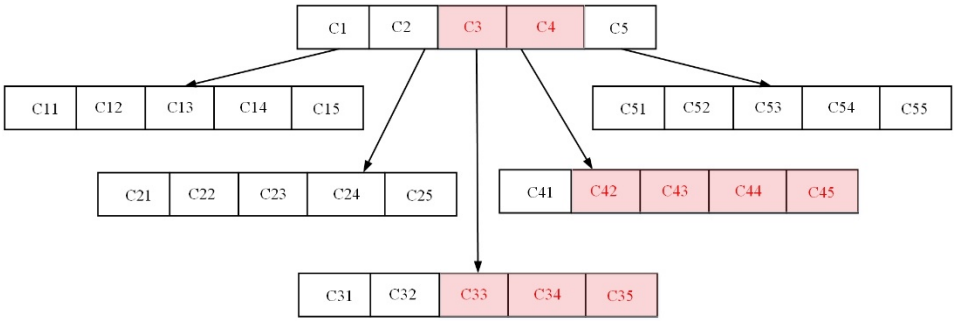


Fig. 2. Index diagram

4.3 Post Processing

In this section, we present a post processing to improve the accuracy of the range query using constrained inference. The general methods to obtain the query count are: if the node is completely contained in the query rectangle, return its noisy count; if the node intersects the query range, we assume that the data points in the node are distributed uniformly, and the count of the intersection is node counts multiplied by the ratio of the intersection part to the query rectangle. [13] proposes a post-processing method based on the least squares estimate(OLS), but the algorithm is relatively complex. We apply a constrained inference with similar effects based on its idea. As shown in **Fig. 1**, the count of the node C_1 can be

its own count Y_{C_1}, or the sum of its five children $Y_{C_{11}}+Y_{C_{12}}+Y_{C_{13}}+Y_{C_{14}}+Y_{C_{15}}$. However, if the count is set to 5/6 of its own count plus 1/6 of its five children's count, i.e. $5/6Y_{C_1}+1/6(Y_{C_{11}}+Y_{C_{12}}+Y_{C_{13}}+Y_{C_{14}}+Y_{C_{15}})$, the perturbation error will be minimized $PE(Q)=(5/6)Var(Y_{C_1})$. In Algorithm 3, we show the details of post processing:

Algorithm 3 Post Processing

Input: *m* nodes in the 1-dimension
1. If the node is not a child:
2. $sum=\sum_{child\in node}Y_{child}$
3. $adjust=(Y_{node}-sum)/(len(children(node))+1)$
4. $Y_{node}=Y_{node}-adjust$
5. for $child\in node$
6. $Y_{child}=Y_{child}+adjust$

5. Security Analysis

The algorithm is composed of two parts, according to the Theorem 1, i.e. sequential composition. The private budget of the combination algorithm ε is the sum of the budget of its sub-algorithms, ε_i. In addition, the study in [25] proves that the constrained inference will not affect the guarantee of the privacy, hence the budget of the algorithm we proposed is $\varepsilon=\varepsilon_1+\varepsilon_2$.

Theorem 4. The DPIH partitioning algorithm satisfies ε-Differential privacy.

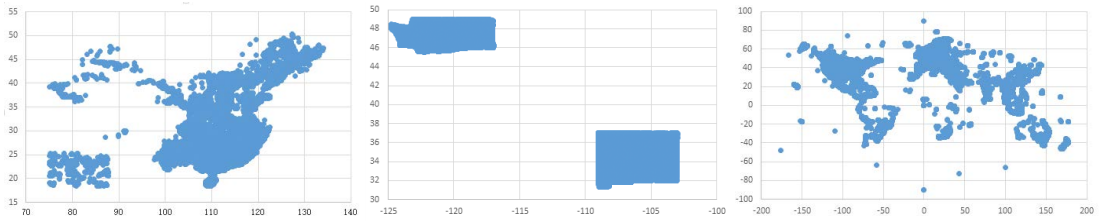
Proof: In Algorithm 1, we first partition the original dataset D into β grids. These grid cells are disjointed. Then, we add Laplace noise with private budget $\alpha\varepsilon$ for each cell. According to the Theorem 2, i.e. parallel composition, the step 2 of the Algorithm 1 satisfies $\alpha\varepsilon$-Differential privacy. The process of partitioning the synthetic datasets D_c (Algorithm 1, step 3) does not consume any private budget. Afterwards, we perform partition on the original dataset *D* using the resulting splitting keys, and inject noise into each cell using the rest budget $(1-\alpha)\varepsilon$. Thus, the step 5 of the Algorithm 1 satisfies $(1-\alpha)\varepsilon$-Differential privacy. According to the Theorem 1, the DPIH partitioning algorithm satisfies ε-Differential privacy. □

6. Experiments

To analyze the accuracy of the query range more comprehensively, experiments are conducted in three real-life spatial datasets: Reservoir, Road[1] and Checkin[2], they have different distribution characteristics and different volumes, the distribution of the three datasets is shown in **Fig. 3** below. The Reservoir dataset has approximately twenty thousand spatial locations that are obtained by searching the keyword "reservoir" on the open platform of the Gaode Map, thus, the distribution of the Reservoir is shaped like a map of China. The

[1] http://www.census.gov/geo/maps-data/data/tiger-geodatabases.html
[2] http://snap.stanford.edu/data/loc-gowalla.html

Road dataset is hundreds of thousands-level, including the coordinates of road intersections in the New Mexico and Washington areas. The distribution of this dataset is quite special. Most data points are in the two relatively concentrated regions, and the remaining area is almost blank. Checkin is derived from a location-based social network where users share their locations. There are approximately 6.4 million tuples in this dataset, which is so huge that experimentation on a personal computer is infeasible. Hence, our experimental results are tested on the cloud.



Reservoir Road Checkin

Fig. 3. Distribution of datasets

We set five different query sizes for each dataset, Q1 is the smallest. Each Qi+1 increases approximately twice in both X and Y directions. Q5 is the largest query rectangle and it covers approximately 1/4 of the domain. The information about the datasets and the query sizes are all shown in **Table 1**. There are 500 queries randomly generated for each query rectangle under each different budget. The average of these relative errors is taken as the result under this query in answering them.

Table 1. Datasets information and query sizes

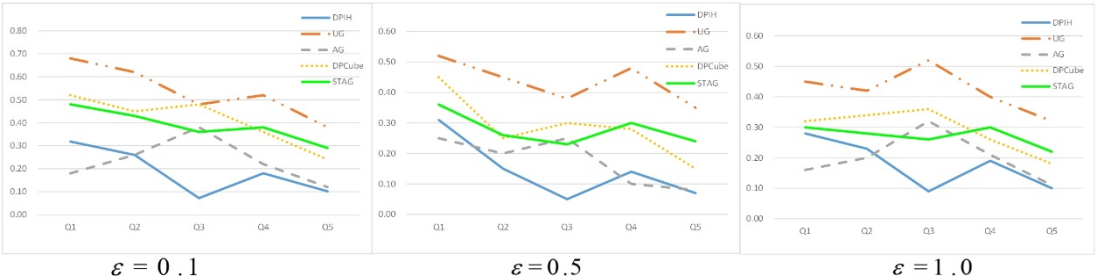
Dataset	Points	Domain range	Q1	Q2	Q3	Q4	Q5
Reservoir	0.02M	59*31	2*2	5*5	10*10	15*8	30*15
Road	0.36M	25*20	1*1	2*2	3*3	6*5	12*10
Checkin	6.44M	360*150	10*5	20*10	45*20	90*40	180*75

The step 1 of the Algorithm 1 splits the domain with β grid cells. We set β as 100, that is, 10*10 grid cells over the total domain. The step 2 adds noise with the private budget $\alpha\varepsilon$ and step 5 adds noise with the rest budget $(1-\alpha)\varepsilon$. In most of the experiments, we set $\alpha = 0.5$, but in the Road dataset, we found that if we set $\alpha = 0.3$, the results will be better. We use line graphs to illustrate the relative error under all query sizes and provide a comparison among different algorithms. The blue line indicates our algorithm, the orange line is UG and the gray line is AG which are all proposed in [13]. The yellow line is DPCube proposed in [14], the green line is STAG proposed in [20].

6.2 Experimental Results

We compare the performance of our proposed DPIH with UG, AG, DPCube and STAG in each three datasets. In differential privacy, ε represents the degree of privacy protection, smaller value of ε indicates higher degree of privacy protection. The relative error defined

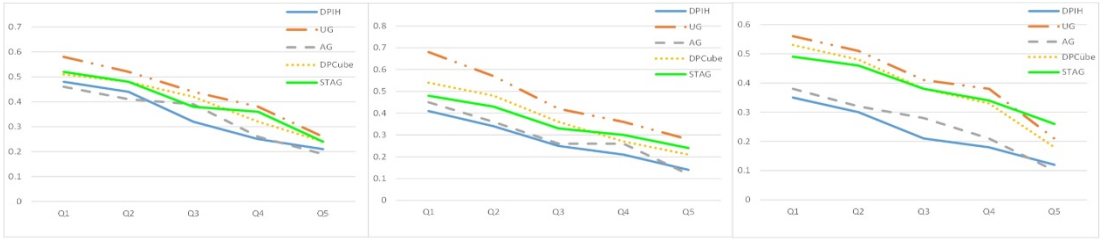
in Section 3.3 is a metric of data availability, it can evaluate the quality of each partition by the accuracy of its answers to range queries, the relative error is smaller indicates the data availability is higher. In our experiments, we set three different ε values, ε=0.1, ε=0.5 and ε=1, we compare the relative errors of our algorithm with the other four algorithms which have some influence on this research field, and the experimental results are shown as follows:



$\varepsilon = 0.1$ $\varepsilon = 0.5$ $\varepsilon = 1.0$

Fig. 4. Compare relative errors of UG, AG, DPCube, STAG and DPIH on the Reservoir dataset

Fig. 4 shows that the relative error of the DPIH algorithm in the query size of Q3 and Q5 is much better than other algorithms under the Reservoir dataset, it performs worse than the AG algorithm in the small query such as Q1, but still better than UG and DPCube and STAG algorithms. We found that when the budget $\varepsilon = 1$, if we adjust the value of m to $m = \sqrt{N\varepsilon / c\sqrt{2}}$, and when $\varepsilon = 0.1$, we adjust m to $m = \sqrt{\left(\sqrt{2}N\varepsilon\right)/c}$, the relative error will be significantly reduced. The reason why our algorithm performs considerably better under Q3 and less than ideally under Q1 is that the cells generated by our algorithm areas not as uniform as the standard grid cells, the areas where data points are densely distributed will be partitioned more times. Nevertheless, the grids are assumed to be uniform when we calculate the value of m. Thus, if the query size is excessively small, more errors would be generated, suggesting that the value of m can be tuned adaptively if the dataset is biased.



$\varepsilon = 0.1$ $\varepsilon = 0.5$ $\varepsilon = 1.0$

Fig. 5. Compare relative errors of UG, AG, DPCube, STAG and DPIH on the Road dataset

The Road dataset is quite special and biased. There are only two densely distributed regions, while the remaining regions are blank. In the algorithm proposed in our paper, we add or delete some spatial points randomly according to the noisy count of the cell when synthesizing a new dataset. It will generate certain errors when large blank areas appear. We can see that the DPIH performs worse on the Road dataset than on the other two datasets under different ε values and query sizes. STAG performs poor in the Reservoir and the Road datasets because it draws the samples from the datasets in advance with a 1/10 probability, thus it

will generate large sparse area after sampling, and it will produce a lot of errors when partition and add noise on the sparse area. From the observation of **Fig. 5**, under the query range of Q1 and Q5, the relative error of DPIH is slightly higher than the AG algorithm, but still lower than UG and DPCube. However, when the side length of the query rectangle is approximately 1/6~1/4 of the domain, DPIH performs much better than the other algorithms because the Road dataset has large blank areas, more errors will be caused by the process of synthesizing a new dataset in the Road dataset than in the other datasets. Fortunately, if we set $\alpha = 0.3$, in the other words, reduce the budgets allocated to step 2 of algorithm 1 and add more noise to the count, the relative error distinctly decreases.

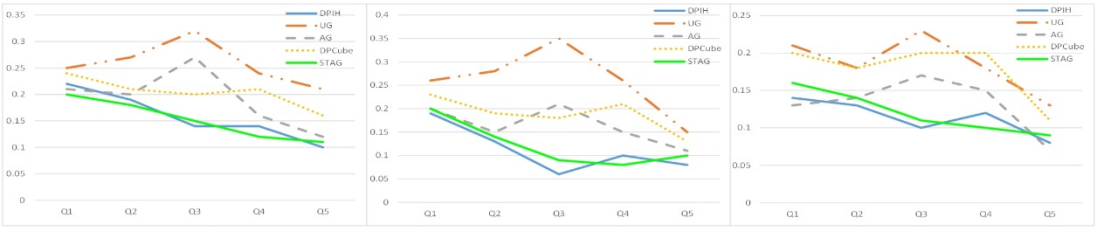


Fig. 6. Compare relative errors of UG, AG, DPCube, STAG and DPIH on the Checkin dataset

Checkin is a million-level dataset and its distribution is not overly clustered or skewed. From the observation of **Fig. 6**, DPIH performs well in this dataset. The cardinality of the dataset is huge, and the range for the partitioning granularity m is between 253 and 800, according to DPIH, the partitioning process doesn't consume the private budget, even if some errors arise from the process of synthesizing the dataset, it can be corrected in the subsequent post processing, thus the advantages of DPIH can be well demonstrated in Checkin. STAG also provides good results in the Checkin dataset, unlike the general partition algorithms, it constructs a three-layer grid partitioning, and it merges the cells into coarse-grained cells if the count of these cells is below the threshold. The relative errors of STAG are lower than DPIH under Q1, Q2, Q4 when $\varepsilon = 0.1$, but it performs worse than DPIH in almost other cases.

In summary, the data-dependent partitioning algorithm proposed in this paper is applicable to unbiased datasets and performs much better in larger volume datasets. For some skewed datasets such as Road, we can reduce the value of α, namely, allocating more budget to the node count and allocating less budget to synthesize the dataset, better results will be returned. Nevertheless, the data-dependent partition is less efficient than the data-independent partition and needs to be tested on a server or cloud if the dataset is million-level.

7. Conclusions

In this paper, we propose a novel multidimensional data-dependent partitioning algorithm based on differential privacy. It is necessary to allocate additional privacy budget for choosing the median in the traditional data-dependent partition. In our algorithm, we first synthesize a new dataset according to the noisy count, then partition the domain on the synthetic dataset. It can save the private budget allocated to protect the partitioning process, and obtain a more accurate release. In the experimental tests on three real-life datasets, our algorithm gets satisfactory results, especially on the million-level datasets, the relative errors

of DPIH are almost smaller than current methods that means DPIH provides more accurate results under the three different degrees of privacy protection. However, our algorithm is data-dependent, so it is difficult to conduct experiments on a personal computer if the dataset is million-level. We plan to extend the idea of partitioning to sequential or dynamic spatial data, and study more efficient and accurate data-dependent partitioning algorithms in the future.

Acknowledgments

This work is partially supported by "The National Key Research and Development Program of China" (Grant No. 2018YFC0407105, 2016YFC0400910); "NSF of Jiangsu Higher Education Institutions of China" (Grant No. 17KJB520043).

References

[1] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no.05, pp. 557-570, 2002. Article (CrossRef Link)

[2] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond K-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 24, 2007. Article (CrossRef Link)

[3] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *Proc. of IEEE 23rd International Conference on Data Engineering*, pp. 106-115, 2007. Article (CrossRef Link)

[4] C. Dwork, "Differential privacy," in *Proc. of the 33rd International Colloquium on Automata, languages and programming*, pp. 1-12, 2006. Article (CrossRef Link)

[5] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. of the 3th Theory of Cryptography*, pp. 265-284, 2006. Article (CrossRef Link)

[6] L. Fan, L. Xiong, and V. Sunderam, "Differentially private multi-dimensional time series release for traffic monitoring," *Data and Applications Security and Privacy XXVII*, pp. 33-48, 2013. Article (CrossRef Link)

[7] H. To, G. Ghinita, and C. Shahabi, "Privgeocrowd: A toolbox for studying private spatial crowdsourcing," in *Proc. of the 31st IEEE International Conference on Data Engineering*, pp. 1404-1407, 2015. Article (CrossRef Link)

[8] H. Samet, J. Gray, "Foundations of multidimensional and metric data structures," *Morgan Kaufmann*, vol. 45, no. 7, pp. 1165-1177, 2006.

[9] A. Inan, M. Kantarcioglu, G. Ghinita, E. Bertino, "Private record matching using differential privacy," in *Proc. of the 13th International Conference on Extending Database Technology*, pp.123-134, 2010. Article (CrossRef Link)

[10] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu. "Differentially private spatial decompositions," in *Proc. of IEEE 28th International Conference on Data Engineering (ICDE)*, pp. 20-31, 2012. Article (CrossRef Link)

[11] H. To, L. Fan, "Differentially Private H-Tree," in *Proc. of the 2nd ACM SIGSPATIAL Workshop on Privacy in Geographic Information Collection and Analysis*, 2015. Article (CrossRef Link)

[12] R. Chen, N. Mohammed and B.C.M. Fung, "Publishing set-valued data via differential privacy," in *Proc. of the 37th Conference of Very Large Databases (VLDB)*, vol. 4, no.11, pp.1087-1098, 2011.

[13] W. Qardaji, W. Yang, and N. Li, "Differentially private grids for geospatial data," in *Proc. of IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 757-768, 2013. Article (CrossRef Link)

[14] Y. Xiao, L. Xiong, and C. Yuan, "Differentially private data release through multidimensional partitioning," in *Proc. of the 7th VLDB Workshop on Secure Data Management*, pp. 150-168, 2010. Article (CrossRef Link)
[15] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proc. of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 94-103, 2007. Article (CrossRef Link)
[16] J. Zhang, X. Xiao, X. Xie, "PrivTree: a differentially private algorithm for hierarchical decompositions," in *Proc. of the 36th ACM International Conference on Management of Data*, pp. 155-170, 2016. Article (CrossRef Link)
[17] J. Lee, C. W. Clifton, "Top-k frequent item sets via differentially private fp-trees," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 931-940, 2014. Article (CrossRef Link)
[18] R. Chen, Q. Xiao, Y. Zhang, "Differentially private high-dimensional data publication via sampling-based inference," in *Proc. of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 129-138, 2015. Article (CrossRef Link)
[19] M. Fanaeepour, B. I. P. Rubinstein, "Differentially private counting of users' spatial regions," *Knowledge And Information System*, vol. 54, no. 1, pp. 5-32, 2018. Article (CrossRef Link)
[20] X. Zhang, K. Jin, X. Meng, "Private Spatial Decomposition with Adaptive Grid," *Computer Research and Development*, vol. 55, no. 6, pp. 1143-1156, 2018. Article (CrossRef Link)
[21] P. Xiong, T. Zhu, W. Niu, "A differentially private algorithm for location data release," *Knowledge And Information System*, vol. 47, no. 3, pp. 647-669, 2016. Article (CrossRef Link)
[22] H. To, G. Ghinita, L. Fan, "Differentially Private Location Protection for Worker Datasets in Spatial Crowdsourcing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, pp. 934-949, 2017. Article (CrossRef Link)
[23] K. Al-Hussaeni, B. C. M. Fung, F. Iqbal, "Differentially private multidimensional data publishing," *Knowledge And Information System*, vol. 56, no. 3, pp. 717-752, 2018. Article (CrossRef Link)
[24] S. Muthukrishnan, V. Poosala, T. Suel, "On Rectangular Partitionings in Two Dimensions: Algorithms, Complexity, and Applications," in *Proc. of International Conference on Database Theory (ICDT)*, pp. 236–256 , 1999. Article (CrossRef Link)
[25] F. McSherry, "Privacy integration queries: an extensible platform for privacy-preserving data analysis," in *Proc. of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 19-30, 2009. Article (CrossRef Link)

Sujin Cai received the B.E. degree in computer science and technology from Soochow University and the M.E. degree in computer science and technology from Nanjing University of Science and technology, China, respectively. She is currently a Ph.D Candidate in computer science and technology at Hohai University, China. Her research interests include Security and Privacy Preserving.

Xin Lyu is a lecturer in college of computer and information at Hohai University. His research interests include Cryptography, Network Information Security and Privacy-Preserving Theory and Technology. He has published over 50 papers in refereed international journals and conference proceedings.

Duohan Ban received the B.E. degree in computer science and technology from Hohai University, Jiangsu, China. She is currently a master student in computer science and technology at Hohai University, Jiangsu, China. Her research interests include Image Encryption and Network Security.