Differentially Private Multi-dimensional Time Series Release for Traffic Monitoring

Liyue Fan, Li Xiong, Vaidy Sunderam

Differentially Private Multi-Dimensional Time Series Release for Traffic Monitoring

Liyue Fan, Li Xiong, and Vaidy Sunderam

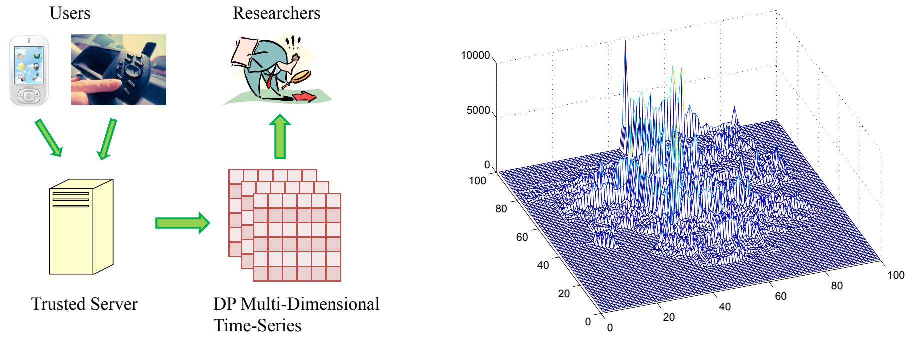
Emory University
Atlanta GA 30322, USA
{lfan3,lxiong,vss}@mathcs.emory.edu

Abstract. *Sharing real-time traffic data can be of great value to understanding many important phenomena, such as congestion patterns or popular places. To this end, private user data must be aggregated and shared continuously over time with data privacy guarantee. However, releasing time series data with standard differential privacy mechanism can lead to high perturbation error due to the correlation between time stamps. In addition, data sparsity in the spatial domain imposes another challenge to user privacy as well as utility. To address the challenges, we propose a real-time framework that guarantees differential privacy for individual users and releases accurate data for research purposes. We present two estimation algorithms designed to utilize domain knowledge in order to mitigate the effect of perturbation error. Evaluations with simulated traffic data show our solutions outperform existing methods in both utility and computation efficiency, enabling real-time data sharing with strong privacy guarantee.*

Keywords: Traffic Monitoring, Multi-Dimensional Time-Series, Differential Privacy

1 Introduction

Sharing real-time traffic data is essential to discovering useful and previously unknown knowledge. As illustrated in Figure 1(a): a wireless service provider gathers data from individual users about their locations, speeds, mobility, etc. The aggregated data, e.g. the number of users present at certain locations during a given time period, can be shared with third party researchers to be mined for commercial interest, such as popular places, as well as public interests, such as congestion trends. Figure 1(b) provides a snapshot of aggregated traffic data at a single time stamp. As is shown, the two-dimensional space is partitioned by a 100×100 grid. For each cell in the 2D space, Figure 1(b) plots the number of users within its extent at the given time stamp. Since the spatial distribution of wireless users could change over time due to movement, such a snapshot is needed at every time stamp in order to perform real-time data mining tasks. However, the privacy of individual users may be affected if their private data is shared with untrusted third parties. The goal of our work is to enable the server/data holder to share useful multi-location aggregates continuously (multi-dimensional time series) while preserving individual privacy.



(a) Privacy-Preserving Data Sharing (b) A Snapshot of 2D Time Series

Fig. 1. Traffic Data Monitoring

The current state-of-the-art paradigm for privacy-preserving data publishing is *differential privacy* [1], denoted as “DP” in Figure 1(a). Differential privacy requires that the aggregate statistics reported by a data publisher be perturbed by a randomized algorithm $\mathcal{A}$, so that the output of $\mathcal{A}$ remains roughly the same even if any single tuple in the input data is arbitrarily modified. This ensures that given the output of $\mathcal{A}$, an adversary will not be able to infer much about any single tuple in the input, and thus privacy is protected.

Despite the large number of methods on differentially private data publication [4–6, 10, 11, 15–17], there does not currently exist an approach to sharing multi-dimensional time series data. We summarize our challenges below:

- Due to the data correlation between time stamps, a straightforward application of the standard differential privacy mechanism at every time stamp leads to an overall perturbation error of $\Theta(T)$ by composition theorem [12], where T is the length of the time series, which severely limits the utility of the published data when T is large.
- Another challenge is data sparsity in the spatial domain. As shown in Figure 1(b), the majority of cells in the 2D space have very low to zero frequency. In reality, the total number of cells can be very large with respect to the total number of users. The data sparsity poses great challenge for privacy-preserving techniques since the perturbation noise is likely to dominate the released value in presence of a small set of users.
- Furthermore, the monitoring application requires that private, released data is provided in real-time. Therefore, existing techniques that require time-series transformation or prohibitive computation time are not applicable to performing real-time tasks.

Our Contributions. In this paper, we propose a real-time framework and two estimation algorithms to address the above challenges in multi-location traffic monitoring with differential privacy. Domain knowledge, such as road network and density, is utilized by our solutions to model the auto-correlation of individual cells over time as well as correlation between neighboring cells. The temporal estimation algorithm establishes an internal time series model for each individual cell and performs posterior estimation to improve the utility of shared aggregate per time stamp. The spatial estimation algorithm builds a spatial indexing structure based on Quadtree to group similar cells together and to reduce the impact of data sparsity. Our solutions provide a strong privacy guarantee. Both algorithms outperform baseline solution as well as state-of-the-art methods in sharing time series or static multi-dimensional data, providing real-time data release without compromising the utility of shared data.

The rest of the paper is organized as follows: Section 2 provides the problem definition, preliminaries on differential privacy, and the baseline solution. Section 3 presents the technical details of our proposed solutions, i.e. temporal estimation and spatial estimation. Section 4 presents a set of empirical results. Section 5 reviews previous works related to data sharing methods with differential privacy. Section 6 concludes the paper and states possible directions for future work.

2 Problem Statement and Preliminaries

2.1 Problem

In the traffic monitoring application we consider, a set of objects are moving in a two-dimensional space and a central server is collecting information about their locations over time. We adopt a fine-grained 2D grid that partitions the space G into $w \times w$ cells, where w is a constant number called *resolution.* We further assume the expected collection time span is T and denote k as the discrete time index where $0 \leq k < T$. For each cell c in G, we define the *frequency series* of c as $\mathbf{X}^c = \{x_k^c | \; 0 \leq k < T\}$, where x_k^c represents the number of objects within its extent at time stamp k. A multi-dimensional time series $\mathbf{X}^G$ can be defined as the set of frequency series of every cell c in G, i.e. $\mathbf{X}^G = \{\mathbf{X}^c | \; c \in G\}$. A *snapshot* of the spatio-temporal database $\mathbf{X}_k^G$ is defined as the set of cell frequencies at time k, i.e. $\mathbf{X}_k^G = \{x_k^c | \; c \in G\}$. The same terms for the released data set $\mathbf{R}^G$ can be defined similarly.

Problem 1 *Given a multi-dimensional time series $\mathbf{X}^G$ where $G = w \times w$ cells, for each snapshot $\mathbf{X}_k^G$, release in real-time a sanitized version $\mathbf{R}_k^G$ such that the overall release $\mathbf{R}^G$ satisfies α-differential privacy, where α is a user-specified privacy level.*

Note that sharing $\mathbf{R}^G$ will enable a variety of data mining tasks. Therefore we use a generic utility metric, i.e. relative error, to measure the usefulness of the released series for each cell c:

Definition 1 (Utility Metric). The utility of a published series $\mathbf{R}^c = \{r_k^c\}$ can be measured by the *average relative error*, denoted as E^c, against the original time-series $\mathbf{X}^c = \{x_k^c\}$.

$$E^c = \frac{1}{T} \sum_{k=0}^{T-1} \frac{|r_k^c - x_k^c|}{max\{x_k^c, \delta\}} \tag{1}$$

where δ is a user-specified constant (also referred to as *sanitary bound* as in [14]) to mitigate the effect of excessively small query results, e.g. $0's$. Here we set $\delta = 1$ throughout the entire time-series for all cells.

2.2 Differential Privacy

The privacy guarantee provided by our solutions is *differential privacy* [1]. Simply put, a mechanism is differentially private if its outcome is not significantly affected by the removal or addition of a single user. An adversary thus learns approximately the same information about any individual user, irrespective of his/her presence or absence in the original database.

Definition 2 (α-Differential Privacy [1]). A non-interactive privacy mechanism $\mathcal{A}$ gives α-differential privacy if for any dataset D_1 and D_2 differing on at most one record, and for any possible anonymized dataset $\widetilde{D} \in Range(\mathcal{A})$,

$$Pr[\mathcal{A}(D_1) = \widetilde{D}] \leq e^\alpha \times Pr[\mathcal{A}(D_2) = \widetilde{D}] \tag{2}$$

where the probability is taken over the randomness of $\mathcal{A}$.

The privacy parameter α, also called the *privacy budget* [12], specifies the degree of privacy offered. Intuitively, a lower value of α implies stronger privacy guarantee and a larger perturbation noise, and a higher value of α implies a weaker guarantee while possibly achieving higher accuracy. We will examine the privacy-utility tradeoff in the experiment section.

Laplace Mechanism. Dwork et al. [5] show that α-differential privacy can be achieved by adding i.i.d. noise $\tilde{N}$ to each query result $q(D)$:

$$\tilde{q}(D) = q(D) + \tilde{N} \tag{3}$$

$$p(\tilde{N} = x) = \frac{1}{2\lambda} e^{-|x|/\lambda}\ , \quad \lambda = GS(q)/\alpha \tag{4}$$

The magnitude of $\tilde{N}$ conforms to a Laplace distribution in Equation (4) where $GS(q)$ represents the *global sensitivity* [5] of a query q. In the traffic monitoring application, each aggregate value is a *count* query and $GS(count) = 1$. Later on in this paper, we denote the Laplace distribution with 0 mean and λ scale as $Lap(0, \lambda)$.

Composition. The composition properties of differential privacy provide privacy guarantees for a sequence of computations, e.g. a sequence of *count* queries.

Theorem 1 (Sequential Composition [12]). Let $\mathcal{A}_i$ each provide α_i-differential privacy. A sequence of $\mathcal{A}_i(D)$ over the dataset D provides $(\sum_i \alpha_i)$-differential privacy.

Algorithm 1 Laplace Perturbation Algorithm(LPA)

Input: Raw data series $\mathbf{X}^G$, privacy budget α
Output: Released data series $\mathbf{R}^G$

```
1: for each cell $c \in G$ do
2:     for each time stamp $k$ do
3:         $r_k^c \leftarrow$ perturb $x_k^c$ by $Lap(0, \frac{T}{\alpha})$;
```

2.3 Baseline Solution

A baseline solution to sharing differentially private multi-dimensional time series is to apply the standard Laplace perturbation at each time stamp to every frequency series. For any c, if every released aggregate satisfies α/T-differential privacy, by Theorem 1 the released frequency series guarantees α-differential privacy. We summarize the baseline algorithm in Algorithm 1 and Line 3 represents the Laplace mechanism to guarantee α/T-differential privacy for each released aggregate. Empirical studies of the LPA algorithm against our proposed solutions are included in Section 4.

3 Proposed Solutions

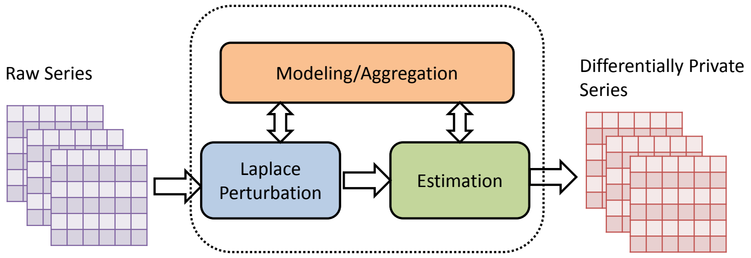


Fig. 2. Differentially Private Traffic Monitoring Framework

In this section, we present our proposed solutions for privacy-preserving traffic monitoring. Figure 2 provides a high-level overview of the system framework. At every time stamp, the input multi-dimensional data is perturbed by the *Laplace Perturbation* mechanism to guarantee differential privacy. Then the perturbed data can be post-processed by the *Estimation* module to produce a more accurate, released version. Domain knowledge, such as road network and population density, is utilized by *Modeling/Aggregation*, which in return interacts with the perturbation component as well as the estimation method in use. Below we describe in detail two separate estimation algorithms: one is to perform time-wise estimation for each individual cell, while the other is to perform spatial aggregation and estimation over the entire 2D space.

3.1 Temporal Estimation

For each cell c in space G, we can apply our recently proposed filtering-based posterior estimation technique [9] to the cell frequency series $\mathbf{X}^c$. The key idea is to utilize an internal time series model for the frequency series and to estimate the true aggregate values based on the Laplace perturbed values. The additional innovation in this paper is that we model different types of cells according to the domain knowledge on the road networks. Below we briefly show how to model the cell frequency series and refer interested readers to our work [9] for further implementation details.

Note that the internal model of cell frequencies depends on many factors, such as location, overall population, road network, etc. Here we simply classify each cell as *sparse* or *dense* based on road network connections and assume the same internal model for cells within each category. For each cell c, its frequency series $\mathbf{X^c}$ can be represented by the following process model:

$$x_{k+1}^c = x_k^c + \omega^c \ , \quad p(\omega_c) \sim \mathbb{N}(0, Q^c) \tag{5}$$

which states that the count values of consecutive time stamps should be consistent except for a white, Gaussian noise ω^c. In particular, ω^c is called the process noise and it follows a normal distribution. Q^c value indicates the level of variation between adjacent time stamps. Intuitively, sparse cells exhibit little variation since very few objects travel within them, therefore we should specify a small Q^c value for such cells. On the other hand, higher Q^c should be assigned for dense cells since they are visited more frequently in reality.

The noisy observation, which is obtained from the Laplace Perturbation mechanism, can be modeled as follows:

$$z_k^c = x_k^c + \nu \ , \quad \nu \sim Lap(0, 1/\alpha_0) \tag{6}$$

where ν, called the measurement noise, corresponds to the Laplace noise and is independent of c. The differential privacy budget for each traffic count is $\alpha_0 = \alpha/T$, since the overall privacy budget α is uniformly allocated to each time stamp.

For posterior estimation purpose, it is sufficient and computationally attractive to approximate ν by a white Gaussian error according to [9]:

$$\nu \sim \mathbb{N}(0, R) \ . \tag{7}$$

Therefore here we adopt the above Gaussian approximation for every cell and use the Kalman filter based filtering technique [9] for posterior estimation.

The outline of the temporal estimation algorithm is presented in Algorithm 2. For every time stamp k and each cell c, we derive a predicted frequency with the *Predict* procedure. Upon receiving the noisy observation, we can derive a posterior estimate with the *Correct* procedure, by linear combination of prediction and observation. The derivation of posterior estimate as well as *Predict* and *Correct* steps can be found in [9] and therefore omitted here for brevity.

Algorithm 2 Temporal Estimation Algorithm

```
Input: Raw data series $\mathbf{X}^G$, privacy budget $\alpha$
Output: Released data series $\mathbf{R}^G$
for each timestamp $k$ do
    for each cell $c \in G$ do
        $prior \leftarrow c.\mathbf{Predict}(k)$ ;
        $z_k^c \leftarrow$ perturb $x_k^c$ by $Lap(0, \frac{T}{\alpha})$;
        $posterior \leftarrow c.\mathbf{Correct}(k, prior, z_k^c)$;
        $r_k^c \leftarrow posterior$;
```

The advantage of temporal estimation approach is that it utilizes the internal time series model and the observations to form an educated guess, which is shown in [9] to greatly improve the accuracy of released data per time stamp. As for complexity, we can see that the computation time requirement is $O(w^2)$ for every time stamp where w is the spatial resolution, since only $O(1)$ operations are performed for each cell.

3.2 Spatial Estimation

When every cell is perturbed individually, data sparsity imposes great utility challenge, i.e. high relative error due to perturbation. We thus are motivated to group similar cells to overcome the data sparsity issue. Considering the spatial correlation among cells, it is very likely that neighboring cells are connected by the same roads therefore are more similar to each other. To utilize this heuristic, we propose to aggregate similar cells into partitions according to spatial vicinity and perform estimation within each partition assuming uniformly distributed objects within the partitions.

We propose a top-down space partitioning approach based on Quadtree due to several considerations. One advantage of Quadtree is its efficiency: it recursively partitions a 2D space into 4 quadrants disregard the actual object distribution in the space. Another advantage of Quadtree is that it doesn't incur any extra privacy cost due to its independence from data. In contrast, the kdTree structure proposed by Cormode et al [4] does require extra privacy budget spent on finding the "private median". Since the privacy budget for each time stamp is very limited, we believe that Quadtree is more suitable in the multi-dimensional time series scenario.

We outline the spatial aggregation algorithm based on Quadtree in Algorithm 3. Line 5 checks every node/partition for the splitting condition. Line 6 splits a partition into four equal quadrants. The *node*.homogeneous() method returns true if all the cells within the partition belong to the same category. Again, each cell is pre-classified as *sparse* or *dense* based on domain knowledge. We stop splitting a partition if it is homogeneous. Otherwise, as long as the predefined depth threshold d is not violated, we further split the partition in the hope of reducing the class impurity in each child partition. The value of d represents

Algorithm 3 QuadTreeAgg Algorithm

Input: 2D grid G, depth threshold d
Output: QuadTree index structure QT

```
$QT.root \leftarrow G$;
$queue$.add($QT.root$) ;
while ! $queue$.empty() do
    $node \leftarrow queue$.remove() ;
    if ! $node$.homogeneous() and $node.depth < d$
        $node$.split() ;
        $queue$.add($node.children$) ;
```

Algorithm 4 Spatial Estimation Algorithm

Input: Raw data series $\mathbf{X}^G$, depth threshold d, privacy budget α
Output: Released data series $\mathbf{R}^G$

```
1: $QT \leftarrow$ QuadTreeAgg($G, d$); # initialize the quadtree index
2: for each timestamp $k$ do
3:     for each partition $p \in QT$ do
4:         $p_k \leftarrow \sum_{c \in p} x_k^c$ ;
5:         $\tilde{p}_k \leftarrow$ perturb $p_k$ by $Lap(0, \frac{T}{\alpha})$;
6:         $r_k^c \leftarrow \tilde{p}_k / p.size(), c \in p$ ;
```

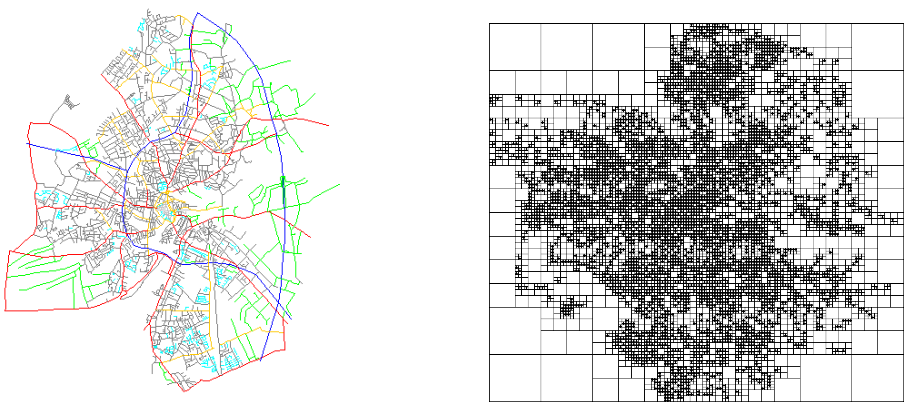
the aggregation level. Setting $d = 0$ implies that all cells are aggregated in one partition. Since the uniform assumption within the partition does not hold, high estimation error will be incurred. On the other hand, a higher value of d implies that many partitions will be further split to produce homogeneous regions so as to reduce estimation error. However, due to data sparsity, very few moving objects will fall into each partition when it is small. Therefore, the perturbation error will dominate the released data in that case. Clearly the optimal d value depends on the spatial distribution of cells. We will examine the impact of d in the experiment section.

Once the Quadtree index structure of the space G is established, we assume uniform data distribution within each partition and estimate each cell frequency with average partition frequency. The spatial estimate algorithm is described in Algorithm 4. For each time stamp k, a partition count is aggregated from cells for every partition (Line 4). It is then perturbed by the Laplace mechanism to guarantee differential privacy (Line 5) and the average noisy count is used to estimate the frequency of each cell within the partition (Line 6). The intuition is that the cells within each partition have similar density. Therefore by uniformly distributing the noisy partition count to each cell, we reduce the magnitude of perturbation error applied to each cell without compromising the accuracy.

One advantage of the spatial estimation algorithm is that it relies on simple and practical assumptions. The complexity is also $O(w^2)$ for each time stamp since every cell is visited $O(1)$ times. Although it takes extra time to build the spatial index for initialization, we see it as a one-time cost which can be done off-line. The runtime of the spatial estimation is reduced because only one perturbation noise is needed for every partition at every k (Line 5). In contrast, both the baseline LPA algorithm and the temporal estimation algorithm will generate one perturbation noise for each cell at every k. We will study their runtime performance in the next section.

4 Evaluation

We implemented the proposed algorithms as well as alternative methods in Java with JSC[1] for simulating the statistical distributions. All experiments were conducted using a 2.90GHz Intel Core i7 PC with 8GB RAM.



(a) Map of Oldenburg

(b) Partitions by Quadtree

Fig. 3. Overview of Data Set

Data Set. We generated synthetic traffic data with the Brinkhoff generator [2]. The input of the generator is the road map of Oldenburg in Germany[2] (Figure 3(a)),which contains 6,105 nodes and 7,035 edges, and the output is a set of moving objects on the road network. We created the data set with 100 discrete timestamps, with 500,000 objects at the beginning and 25,000 new objects introduced at every time stamp. The starting positions and destinations of the moving objects are selected randomly by the generator (see [2] for detailed network-based techniques). Once an object reaches its destination, it disappears from the map. At the server side, we use a 2D grid with 1024 × 1024 cells to

[1] http://www.jsc.nildram.co.uk

[2] http://iapg.jade-hs.de/personen/brinkhoff/generator/

record the locations of the moving objects, with each cell representing approximately 20 × 20 square meters' range in reality. We assign each cell a class label, i.e. *sparse* or *dense*, based on the presence of roads within its extent. Roughly 95% cells are labeled *sparse*, indicating only the rest 5% have been visited by the moving objects. Figure 3(b) visualizes the partition result achieved by the Quadtree-based algorithm with the depth threshold $d = 8$. It can be seen that spares regions around map edge are contained in larger partitions and densely connected regions in map center are further split into smaller partitions. We will evaluate the set of *sparse* cells and the set of *dense* cells separately since they exhibit very different dynamics over time.

Comparison. We compare our proposed solutions against the state-of-the-art methods which are summarized below:

- **DFT** is the Fourier Perturbation Algorithm recently proposed by Rastogi and Nath [13] for sharing single time series. It first performs the Discrete Fourier Transform on an input time series and retains only the first l DFT coefficients. Those coefficients are then perturbed by the Laplace mechanism to guarantee differential privacy. Finally, the Inverse Discrete Fourier Transform is performed on the perturbed coefficients to produce a released series. The number of coefficients to preserve, i.e. l, is a user-specified parameter. In our empirical study, we set $l = 20$ according to their recommendation [13].
- **kd-hybrid** is proposed by Cormode et al [4] as their best method to achieve differentially private space decomposition with static data. Without the help of a grid, *kd-hybrid* builds a mixture index over the 2D data space that begins with kd-tree and switches to quad-tree at a certain level. They slightly modified the kd-tree algorithm, changing the fanout rate to 4 in order to reduce the privacy budget consumption. According to their studies, *kd-hybrid* is most reliable among several representative differentially private space partitioning methods. They reported the optimal parameter setting empirically with the height set to 8 and the switch level set to 4.

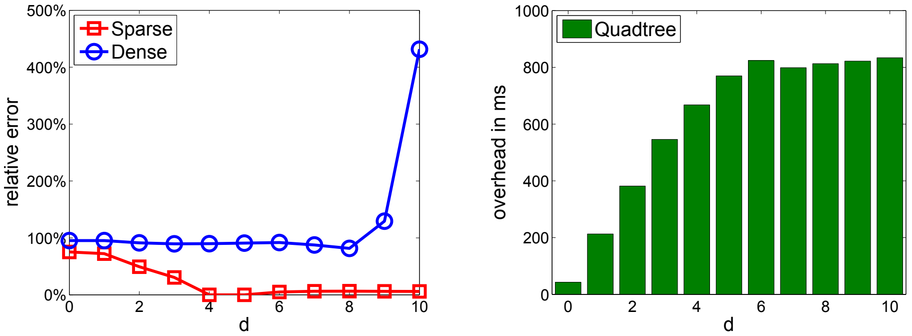
Since the DFT algorithm can be only performed with the complete series, it is not compatible to real-time applications. However, we include it in our evaluation since it serves as a good, off-line reference for utility. As for the *kd-hybrid* algorithm, there are two limitations. One is its high privacy cost since the algorithm iteratively spends budget on finding "private medians" for every data snapshot. The other limitation is its high computation cost: application of the *kd-hybrid* method requires constructing the index structure at every time stamp. Experiments with the author's provided implementation take hours for each iteration, since the domain size and the number of objects in our data set are extremely large. We conclude that the *kd-hybrid* method is too expensive for the continuous, real-time applications and therefore do not include the results in the remaining section.

Table 1. Parameter Settings

Symbol	Description	Default Value
α	Total Privacy Budget	1
w	Resolution for Each Dimension	1024
T	Length of Multidimensional Time Series	100
Q_{sparse}	Process Noise for *Sparse* Cells	10^{-2}
Q_{dense}	Process Noise for *Dense* Cells	10^{3}
R	Gaussian Measurement Noise	10^{6}
d	Depth Threshold for Quadtree	8

4.1 Parameter Impacts

The default parameter setting, unless otherwise noted, is summarized in Table 1. Note that Q_{sparse} and Q_{dense}, which correspond to Q^c in Equation (5) for *sparse* and *dense* cells, can be chosen by domain users and our default setting may not be optimal. As for R from Equation (7), we set its value according to our previous studies [8], which shows that the optimally R is proportional to T^2/α^2.

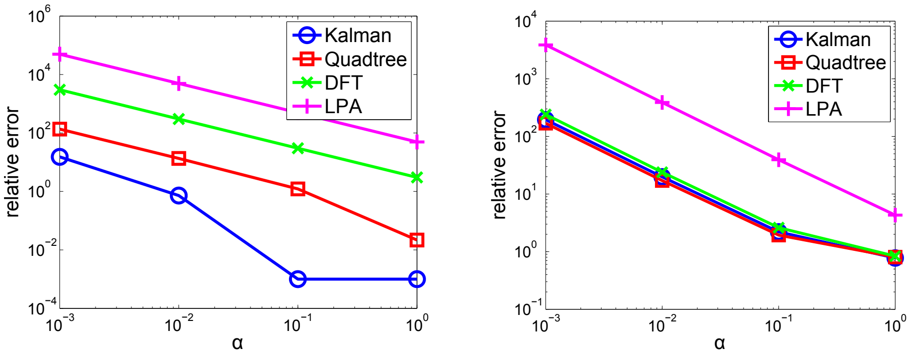


(a) Utility vs. Depth Threshold

(b) Overhead vs. Depth Threshold

Fig. 4. Impact of Depth Threshold on Quadtree-based Spatial Estimation

We study the impact of the depth threshold d used in Algorithm 3 in terms of utility as defined in Equation (1) and runtime. Intuitively, the larger value d takes, the finer partitions the algorithm results in, especially along the border of sparse and dense regions. However, it also incurs a higher overhead to construct the index as we can expect. Figure 4(a) plots separately the utility of released series for sparse cells and dense cells when varying the depth threshold d. For each class of cells, we plot the median relative error to avoid the extremely small or large values. As we increase the threshold value, the error for sparse cells gradually drops to 0 between $d = 0$ and $d = 4$ and remains stable when d value is further increased. This is due to the fact that majority sparse cells are located together (on map edge) and will not take too many splits to be separate from dense cells (in map center). Increasing the d value can help separating those sparse cells on the boarder line. However, the utility of majority sparse cells is not affected since those on the boarder line only count for a very small percentage. On the other hand, dense cells require more splits to achieve optimal separation ($d = 8$). When further split ($d > 8$), the perturbation noise greatly impacts their utility due to data sparsity. Figure 4(b) shows the overhead for constructing the aggregation index when varying the d value. It takes at most 0.9 second and we note that it is a one-time cost. As we expect, a higher depth threshold requires more construction time (from $d = 0$ to $d = 6$). However, when $d > 6$ the overhead does not grow since there are only very few partitions that do not meet the homogeneous requirement at depth 6. As can be seen in Figure 3(b), the densely connected areas in the map are split into finer partitions compared to less populous areas on the map edge.

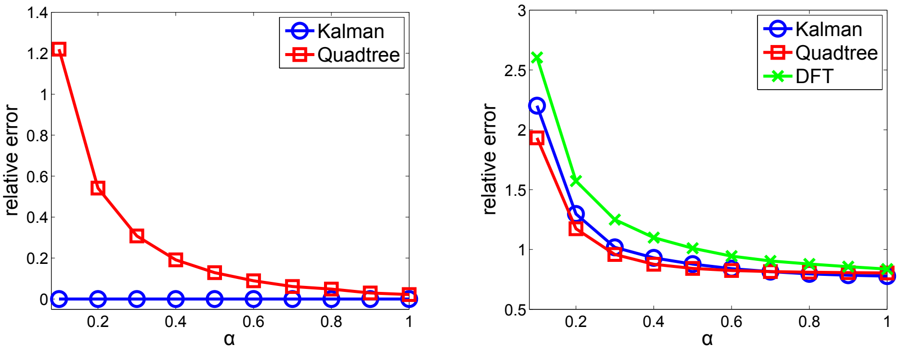


(a) Utility vs. Privacy with *Sparse* Cells (b) Utility vs. Privacy with *Dense* Cells

Fig. 5. Utility of Individual Cells: Comparison of All Methods

4.2 Utility Performance

Utility vs. Privacy. Here we examine the trade-off between utility and privacy. Our proposed solutions, i.e. `Kalman` and `Quadtree`, are compared against the baseline LPA and state-of-the-art DFT algorithm, in terms of utility of individual cells. Figure 5(a) and Figure 5(b) plot the utility of sparse cells and dense cells respectively when varying the overall privacy budget, i.e. α value, from $[10^{-3}, 10^{0}]$. As we can see, the baseline LPA algorithm results in highest relative error in both figures. The DFT algorithm results in high relative error with sparse cells even with high privacy cost ($\alpha = 1$), due to the perturbation and reconstruction error. Our solutions `Kalman` and `Quadtree` outperform both LPA and DFT especially with sparse cells, as `Quadtree` only results in 10% error and `Kalman` produces 0% error when $\alpha = 1$. As for the dense cells, both `Kalman` and `Quadtree` slightly outperforms DFT, which is supposed to be optimal. When $\alpha = 1$, DFT results in 83% error due to lack of smoothness in the original frequency series, while our solutions provide comparable utility to DFT and real-time data release. Figure 6(a) and Figure 6(b) provide a closer look at the utility curves within a more practical range of privacy budget $\alpha \in [0.1, 1]$. DFT and LPA are not plotted in one or both figures because the errors they result in are prohibitive. For sparse cells, `Kalman` provides optimal performance even under small privacy budget ($\alpha = 0.1$), thanks to the accurate modeling. `Quadtree` is able to approach 0% error as α value increases. For dense cells, we observe that `Quadtree` provides the best utility in the same privacy budget range. We conclude that both our proposed solutions outperforms existing methods, allowing for real-time data sharing without compromising the utility.



(a) Utility vs. Privacy with *Sparse* Cells (b) Utility vs. Privacy with *Dense* Cells

Fig. 6. Closer Look at $\alpha \in [0.1, 1]$

Utility of Range Queries. Here we evaluate our solutions with range queries, where each query is a square window that covers a neighborhood of $m \times m$ cells. For each m value, we randomly generate 100 queries of size $m \times m$, evaluate each method with the same set of queries, and plot the average relative error. Note that when $m = 1$, each set query consists of one cell only and therefore the set query error is equivalent to individual cell error. Our findings are summarized in Figure 7. Our temporal estimation algorithm based on the Kalman filter clearly outperforms `Quadtree` and `LPA` with smaller query windows ($m \leq 100$). For all three methods, the relative error shows a growing trend to different extent as the query set size increases, mainly due to the data sparsity in the space. When $m = 500$, we observe that the error of `Kalman` keeps accumulating while `Quadtree` and `LPA` show reduced relative error. We believe that it is because `Kalman` does not explicitly utilize the spatial correlation between cells. When querying the entire space ($m = 1024$), both `Quadtree` and `LPA` provide good utility because the Laplace noise added to each cell is from a zero-mean distribution and the sum of a large set of such noises is likely to be small. Overall, `Quadtree` outperforms `LPA` by making sound estimation within close-to-uniform partitions.

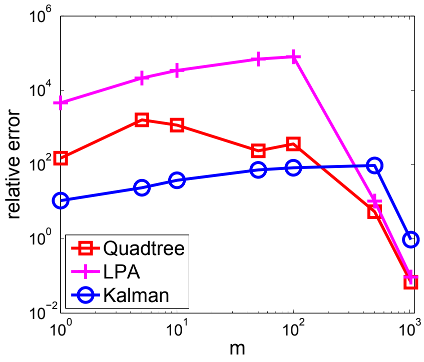


Fig. 7. Utility of Range Queries

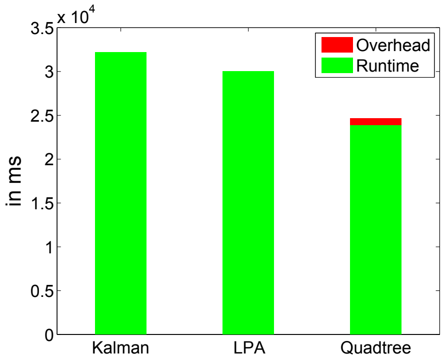


Fig. 8. Runtime Performance

4.3 Runtime Performance

Lastly we compare the runtime performance of our solutions against the baseline since computation time is critical to real-time applications. We measure and plot the runtime for releasing the two-dimensional aggregates for 100 timestamps in order to mitigate random disturbance from the operating system. The results are summarized in Figure 8. As we can see, all three methods take less than 35 seconds to release 100 snapshots of 1024 × 1024 cell frequencies with differential privacy guarantee. Note that the state-of-the-art *kd-hybrid* takes hours to release/evaluate one time stamp. Compared to `LPA`, our solution `Kalman` takes roughly 2 more seconds in total to perform prediction and correction at every time stamp. `Quadtree` turns out to be the most time efficient, even though it has a small overhead in building the spatial index. This is because less perturbation is performed by `Quadtree`, as at every time stamp we only generate one perturbation noise for each partition rather than for each cell as in `LPA`.

5 Related Works

Here we briefly review the most relevant, recent works on differential privacy and time-series data sharing. Dwork et al. [5] established the guideline to guarantee differential privacy for individual aggregate queries by calibrating the Laplacian noise to the global sensitivity of each query. Since then, various mechanisms have been proposed to enhance the accuracy of differentially private data release. Blum et al. [1] proved the possibility of non-interactive data release satisfying differential privacy for queries with polynomial VC-dimension, such as predicate queries. Dwork et al. [7] further proposed more efficient algorithms to release private sanitization of a data set with hardness results obtained.

Several recent works [4, 10, 11, 15–17] study the counting queries on multi-dimensional data, also referred to as histograms or contingency tables, where the multi-dimensional data can be indexed by a tree structure and each level in the tree is an increasingly fine-grained summary/count. Cormode et al [4] propose the class of "private spatial decompositions" and conclude that the hybrid structure *kd-hybrid* provides an accurate yet efficient solution compared to alternatives. When applied to highly self-correlated time-series data, all the above methods, designed to perturb static data, become problematic because of highly compound Laplace perturbation error.

Rastogi and Nath [13] proposed a Discrete Fourier Transform (DFT) based algorithm which implements differential privacy by perturbing the discrete Fourier coefficients. However, this algorithm cannot provide real-time private release in a streaming environment. The recent works [3] [6] on continuous data streams defined the *event-level* privacy to protect an event, i.e. one user's presence at a particular time point, rather than the presence of a user. Our previous work [9] studies the problem of sharing single time-series with user-level differential privacy and we proposed an algorithm with filtering and adaptive sampling to improve the utility of the shared series.

6 Conclusion

We have proposed a real-time framework and two estimation algorithms to address the challenges of differentially private multi-dimensional time series release with application in traffic monitoring. The temporal estimation algorithm establishes a single time-series model for each cell in the space and performs posterior estimation to improve the utility of each released aggregate. The spatial estimation algorithm builds a spatial index by Quadtree and group similar cells together to overcome data sparsity. Domain knowledge is exploited by both estimation methods and is shown beneficial. We observe that the temporal estimation algorithm is highly accurate especially with sparse cells but requires modeling and slightly more running time. On the other hand, the spatial estimation algorithm relies on practical assumptions, demands less computation time, and provides better utility for dense cells and larger range queries. Compared to alternative methods, our solutions outperform the baseline LPA algorithm as well as state-of-the-art methods in both utility and computation efficiency. Future work may include in-depth study of complex spatial-temporal correlation between locations and timestamps.

7 Acknowledgments

This research is supported by NSF under grant CNS-1117763 and AFOSR under grant FA9550-12-1-0240.

References

1. Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 609–618, New York, 2008. ACM.

2. Thomas Brinkhoff. A framework for generating network-based moving objects. *Geoinformatica*, 6(2):153–180, June 2002.
3. T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. In *Proceedings of the 37th international colloquium conference on Automata, languages and programming: Part II*, pages 405–417, Heidelberg, 2010. Springer-Verlag.
4. Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pages 20–31, Washington, DC, 2012. IEEE Computer Society.
5. Cynthia Dwork, Frank Mcsherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *In Proceedings of the 3rd Theory of Cryptography Conference*, pages 265–284, Heidelberg, 2006. Springer-Verlag.
6. Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 715–724, New York, 2010. ACM.
7. Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. pages 381–390, New York, 2009. ACM.
8. Liyue Fan and Li Xiong. Adaptively sharing time-series with differential privacy. *CoRR*, abs/1202.3461, 2012.
9. Liyue Fan and Li Xiong. Real-time aggregate monitoring with differential privacy. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2169–2173, New York, 2012. ACM.
10. Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proc. VLDB Endow.*, 3(1-2):1021–1032, 2010.
11. Chao Li and Gerome Miklau. An adaptive mechanism for accurate query answering under differential privacy. *Proc. VLDB Endow.*, 5(6):514–525, 2012.
12. Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. volume 53, pages 89–97, New York, 2010. ACM.
13. Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 735–746, New York, 2010. ACM.
14. Xiaokui Xiao, Gabriel Bender, Michael Hay, and Johannes Gehrke. ireduct: differential privacy with reduced relative errors. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 229–240, New York, 2011. ACM.
15. Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Trans. on Knowl. and Data Eng.*, 23(8):1200–1214, 2011.
16. Yonghui Xiao, Li Xiong, and Chun Yuan. Differentially private data release through multidimensional partitioning. In *Proceedings of the 7th VLDB conference on Secure data management*, SDM'10, pages 150–168, Heidelberg, 2010. Springer-Verlag.
17. Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Ge Yu. Differentially private histogram publication. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pages 32–43, Washington, DC, 2012. IEEE Computer Society.