Differentially Private Spatial Decomposition of 2D Data for Range Queries

Ruan Pretorius
School of Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa

***Abstract*—Data sets that contain the geographic location of individuals or their activities can be used to enhance business intelligence and traffic flow as well as the process of determining locations and layout of transport systems, political boundaries, and facilities.**

This paper was concerned with the challenge of producing accurate answers to range queries on two-dimensional geospatial data sets while still preserving the privacy of the data set participants. To do this, three differentially private algorithms were selected for spatial decomposition of two different data sets. These spatial decomposition results were then processed to answer 1,000 randomly generated range queries of different sizes. Finally, the relative errors produced in answering these range queries were compared to those obtained from Zhang et al. [1] using the same algorithms on the same data sets.

Overall, considering the results of this study and Zhang et al., the PrivTree algorithm outperformed both the uniform grid and the simple quadtree algorithms for alomst all cases tested. The reason for the superiority of PrivTree was that its spatial decomposition adapts to the data set, growing more densely into more dense parts of the data set.

***Index Terms*—differential privacy, spatial decomposition, range queries, quadtree, uniform grid**

I. Introduction

Data sets that contain the geographic location of individuals or their activities can be used to enhance business intelligence and traffic flow. They can also be used in the process of determining the location and layout of transport systems, political boundaries, and facilities [2], [3].

The information contained in such data sets are clearly valuable to researchers. However, publishing data sets like these can pose a threat to the privacy of the individuals whose data is contained within them. This paper is concerned with the challenge of publishing geospatial data that can be used to accurately answer queries for research purposes while protecting the privacy of the individuals who participated in the data sets.

More specifically, this paper is concerned with producing accurate answers to query type known as a *range query*. A range query is a type of query that returns the amount of data points contained within a specific region. Range queries have particular importance when dealing with geospatial data [2].

Funded by the DSI-NICIS National e-Science Postgraduate Teaching and Training Platform (NEPTTP)

To address the challenge of preserving both privacy and data utility, the paradigm of differential privacy can be used. Differential privacy is a strong privacy guarantee that ensures the answer to a query has very little difference when applied to a data set that differs by the participation of any one individual. This guarantees that no additional information about an individual can be revealed by their participation in the data set.

In this paper, two differentially private algorithms for answering range queries are considered and compared. These are the *Uniform Grid* (UG) and the *PrivTree* algorithms as described in [3] and [1]. These algorithms use different spatial decomposition methods to divide the input data space into smaller sub-regions before computing a noisy count of data points contained within each sub-region. Both the PrivTree and the UG algorithm takes as input a set of two-dimensional coordinates of geospatial data and returns the noisy counts of data points contained in its sub-regions. These noisy count outputs can than be processed for answering range queries.

The rest of this paper is constructed as follows. Section II contains some priliminaries to prime the reader with the necessary definitions and explanations of concepts contained in subsequent sections. Section III then gives a brief overview of related work which has been done on similar topics. This is followed by section IV that describes the data sets that were used in this study and section V that describes the methodology. An exposition and discussion of the results is then given in section VI followed by the conclusions in section VII.

II. Preliminaries

This section contains the necessary definitions and concepts of differential privacy, spatial decomposition, and range query processing to prime the reader for further sections of this paper.

A. Differential Privacy

Differential privacy (DP) was earlier described on an intuitive level as a strong privacy guarantee that ensures the answer to a query has very little difference when applied to a data set that differs by the participation of any one individual. A more formal definition is given next in definition 1.

Definition 1 (ϵ-Differential Privacy [3]). *A randomised algorithm $\mathcal{A}$ satisfies ϵ-differential privacy when the following equation holds for any two neighbouring data sets D and D', and any $S \in Range(\mathcal{A})$:*

$$\Pr[\mathcal{A}(D) = S] \leq e^{\epsilon} \Pr[\mathcal{A}(D') = S]. \quad (1)$$

Note that equation (1) can also be written in the following format which will also be used in this paper:

$$\ln \left(\frac{\Pr[\mathcal{A}(D) = t]}{\Pr[\mathcal{A}(D') = t]} \right) \leq \epsilon \quad (2)$$

In this paper, neighbouring data sets refer to any pair of data sets D and D' that differ by a single record t such that $D = D' + t$ or $D' = D + t$.

One of the most popular mechanisms used to ensure an algorithm $\mathcal{A}$ satisfies ϵ-DP is the *Laplace mechanism* defined next.

Definition 2 (Laplace Mechanism [2]). *Let $q(D)$ be a numeric query function applied to D. The Laplace mechanism publishes a noisy query answer $\mathcal{L}(D) = q(D) + X$, where X is a random variable drawn from the Laplace distribution $X \sim \text{Lap}\left(\frac{\sigma(q)}{\epsilon}\right)$. Here, $\sigma(q)$ is the global sensitivity of the function q. It is the maximum difference in output between $q(D)$ and $q(D')$ for neighbouring databases D and D' (denoted $D \simeq D'$):*

$$\sigma(q) = \max_{D \simeq D'} |q(D) - q(D')|. \quad (3)$$

Note from equation (3), for any counting query such as the range count query, the global sensitivity is unity ($\sigma(q) = 1$) since the maximum possible difference for counting records in D and D' is one.

A useful property of differentially private algorithms is the composition lemma which allows for the calculation of the privacy loss when several differentially private algorithms are applied in a sequence:

Lemma 1 (Composition Lemma [1]). *Let $\mathcal{A}_1, ..., \mathcal{A}_k$ be k algorithms such that $\mathcal{A}_i$ satisfies ϵ_i-diffirential privacy $\forall i \in [1, k]$. Then the sequential composition $(\mathcal{A}_1, ..., \mathcal{A}_k)$ satisfies ϵ-diffirential privacy where $\epsilon = \sum_{i=1}^{k} \epsilon_i$.*

The composition lemma is very useful for calculating the overall privacy guarantee of an algorithm which consists of a sequence of steps as will become clear shortly.

B. Spatial Decomposition

Spatial decomposition involves partitioning the domain of a geometrical space into smaller sub-domains [2]. Applied to a two-dimensional geospatial data set D, this involves partitioning the space Ω that contains all the data points of D into several smaller sub-domains v.

The two different types of spatial decomposition considered in this paper are (i) a uniform grid which involves the partitioning of Ω into a grid of equal-sized sub-domains, and (ii) a hierarchical decomposition that recursively partitions Ω into smaller sub-domains until some criteria are met to form a tree-like structure. Usually the criteria for terminating this tree growth are when a minimum amount of data points are contained within a sub-domain or when the tree reaches a specific depth [1].

The hierarchical spatial decomposition method used in this paper is referred to as a *quadtree*. A quadtree is used to recursively partition a domain of two-dimensional data into four equal-sized sub-domains. Unlike the uniform grid, the partitioning of the qudtree is dynamic. The qudtree grows more densely into regions of dense data so that the data points in each leaf node of the quadtree is almost uniformly distributed [1].

C. Range Query Processing

A range query can be passed to a data set D that asks how many points of D are contained within a specific area q. The true count $c(D)$ can be calculated for the case of the uniformly and hierarchically decomposed spaces as follows.

For the tree structure of the hierarchically decomposed space, the tree can be traversed from the root node towards the leaf nodes. Whenever a sub-domain v is fully contained within the query domain q, the count of the data points within v is added to the output answer. For any domain v that partially intersects with q, the child-domains of v can be inspected until one is found that is fully contained in q. When such a child-domain is found, its count can be added to the output answer as described previously. However, when any leaf nodes are reached whose domains still only partially intersect q, only the amount of points contained within q are added to the output answer [1].

In the case of the uniformly partitioned space, each sub-domain can be treated as a leaf node of the tree described previously. The counts of sub-domains that are fully contained within q, can be added to the output answer. For partially intersecting sub-domains, only the amount of points contained within q are added to the final output answer [3].

This paper is mainly concerned with algorithms that can answer differentially private range queries. The essence of this process is similar to the process of answering range queries by giving the true count as described previously. However, the true counts of domains and sub-domains are perturbed by adding a specific amount of noise to produce a noisy count which is returned instead of the true count.

III. Related Work

Qardaji et al. [3] used a uniform grid (UG) to partition the domain of the data space Ω into a grid of $m \times m$ equally-sized sub-domains. Range queries for this method was answered by returning the noisy counts of the sub-domains that were fully and partially contained within the query domain.The steps of UG are shown in algorithm 1.

The accuracy of this method heavily depended on the selection of m (the amount of sub-domains to use). Their study [3] provided a heuristic method for the selection of m that minimises the error of the query answer while still ensuring

Algorithm 1: Uniform Grid (D, λ, m)

Divide Ω into $m \times m$ equally sized sub-domains v;
for *every new sub-domain* v **do**
  compute true count $c(v)$ of D contained in v;
  compute noisy count $\hat{c}(v) = c(v) + \text{Lap}(\lambda)$;
end
return *domains* v*, noisy counts* $\hat{c}(v)$

a the satisfaction of differential privacy. According to [3], m should be selected to be:

$$m = \sqrt{\frac{N\epsilon}{k}} \tag{4}$$

Where N is the cardinality of the data set, ϵ is the total privacy budget, and k is some small constant that depends on the data set. Their experimental results indicated that $k = 10$ works well for data sets of different kinds [3].

Since the global sensitivity of a counting query is unity, the amount of noise added to each sub-domain of the uniform grid should be drawn from a Laplace distribution $\text{Lap}(\lambda)$ with scale parameter $\lambda = \frac{1}{\epsilon}$ in order to satisfy ϵ-DP.

Zhang et al. [1] used a simple quadtree (also from here on referred to as *simple tree*) as well as a slightly adapted version of this quadtree, called PrivTree, for hierarchical spatial decomposition of the domain of the data space Ω into smaller sub-domains.

The simple tree starts off by setting the domain of the quadtree root node v_0 equal to the domain of the entire data space Ω. It then calculates the true count $c(v)$ and noisy count $\hat{c}(v)$ for the domain $v = v_0$. The noisy count is calculated by adding noise from a Laplace distribution $\text{Lap}(\lambda)$ to $c(v)$. Then two criteria are evaluated that determine if the domain v requires further division into smaller sum-domains. The first criterion is whether the noisy count is greater than a given constant ($\hat{c}(v) > \theta$), and the second criterion is whether the current depth of the quadtree is less than a specified depth limit h. When both of these criteria are satisfied, the domain v is then sub-divided into four sub-domains of equal size and the process is repeated for each new sub-domain. The constant θ is given as an input to the algorithm to specify at what point the tree can stop growing more densely into a domain containing θ or less data points [1]. This process is described in algorithm 2.

Algorithm 2 satisfies ϵ-DP for $\lambda \geq \frac{h}{\epsilon}$. This is because, after the insertion (or deletion) of a single record from D, the counts of h tree nodes will be affected. These h nodes will be the nodes that all contain the new record in their domains. From the composition lemma (lemma 1) and the fact that the global sensitivity for a count query is unity, the scale of Laplace noise λ required to satisfy ϵ-DP is:

$$\lambda \geq \sum_{i=1}^{h} \frac{1}{\epsilon} = \frac{h}{\epsilon}$$

However, as described in [1], the fact that the amount of noise added to $c(v)$ and the depth of the quadtree depends on

Algorithm 2: Simple Tree (D, λ, θ, h)

initialise quadtree with root node $v_0 = \Omega$, mark v_0 unvisited;
while *unvisited node* v *exists* **do**
  mark v as visited;
  compute true count $c(v)$ of D contained in v;
  compute noisy count $\hat{c}(v) = c(v) + \text{Lap}(\lambda)$;
  if $(\hat{c}(v) > \theta)$ *and* $(depth(v) < h - 1)$ **then**
    split v into four equal sub-domains;
    mark these four sub-domains unvisited;
  end
end
return *domains* v*, noisy counts* $\hat{c}(v)$

h is problematic. When h is selected to be small, the noise added to $c(v)$ is also small which leads to more accurate query results. However, since h limits the tree depth, this can lead to premature termination of tree growth in denser regions of the data set which in turn leads to the tree not accurately representing the distribution of the data. On the other hand, if h is selected to be larger in an attempt to avoid the latter problem, the noise added to $c(v)$ will also be larger leading to inaccurate query results and a reduction in data utility. Moreover, these problems cannot be solved by using a heuristic method of selecting h that depends on the specific data set since this will release some information about the data set, invalidating the privacy guarantee [1].

In addition to pointing out the problems of using a simple quadtree for spatial decomposition, Zhang et al. also introduced a solution by adapting the simple qadtree algorithm so that the noise injection is independent of h [1]. This adapted algorithm was called PrivTree and is described next.

The PrivTree algorithm is similar to the simple quadtree algorithm. It also takes as input a data set D, a constant θ, and a Laplace noise scale parameter λ and splits the data space Ω into four equal-sized sub-domains when certain criteria are met. However, it does not require a tree depth limit h. Instead it requires another parameter δ. This new parameter is involved in the slightly different way that PrivTree calculates the noisy count of a domain. Unlike the simple quadtree algorithm that obtains the noisy count from directly adding noise to the true count $c(v)$ of a domain v, the PrivTree algorithm includes a few additional steps. Firstly, a biased count $b(v)$ is calculated according to the following formula:

$$b(v) = c(v) - \delta \cdot depth(v)$$

Secondly, this biased count is increased to $\theta - \delta$ if it is lower than this value. This is done by ising the following formula:

$$b(v) = \max\{\theta - \delta, c(v) - \delta \cdot depth(v)\}$$

Thirdly, a noisy version $\hat{b}(v)$ of this biased count $b(v)$ is calculated by adding Laplace noise as follows:

$$\hat{b}(v) = b(v) + Lap(\lambda)$$

Finally, the current domain v is only split into four sub-domains if $\hat{b}(v) > \theta$. The PrivTree algorithm is shown in algorithm 3.

Algorithm 3: PrivTree $(D, \lambda, \theta, \delta)$

```
initialise quadtree with root node v_0 = Ω, mark v_0
 unvisited;
while unvisited node v exists do
    mark v as visited;
    compute true count c(v) of D contained in v;
    compute biased count b(v) = c(v) − δ · depth(v) ;
    adjust b(v) if small: b(v) = max{b(v), θ − δ};
    compute noisy biased count b̂(v) = b(v) + Lap(λ);
    if b̂(v) > θ then
        split v into four equal sub-domains;
        mark these four sub-domains unvisited;
    end
end
return domains v, noisy counts b̂(v)
```

The PrivTree algorithm, as shown in [1], satisfies ϵ-DP for some constant $\gamma > 0$ if the scale parameter λ of the Laplace noise injection and the parameter δ satisfies the following:

$$\lambda \geq \frac{2e^{\lambda} - 1}{e^{\lambda} - 1} \cdot \frac{1}{\epsilon},$$
$$\delta = \gamma \cdot \lambda.$$

The probability of domain v being divided into smaller sub-domains increases as δ decreases. Therefore, if δ is selected to be small enough, the domain splitting process of PrivTree might not converge or ever terminate. To address this issue, Zhang et al. [1] dicided to set $\delta = \lambda \cdot \ln \beta$, where β is the number of sub-domains created on every split. This is called the *fanout* of the tree. For a quadtree grown on two-dimensional data, $\beta = 4$. Setting δ this way ensures that a domain v with a biased count $b(v) = \theta - \delta$ has a $\frac{1}{2\beta}$ probability of being split [1].

Therefore, the privacy guarantee of PrivTree can be rewritten as follows. The PrivTree algorithm, as shown in [1], satisfies ϵ-DP if the following two conditions are satisfied:

$$\lambda \geq \frac{2\beta - 1}{\beta - 1} \cdot \frac{1}{\epsilon},$$
$$\delta = \lambda \cdot \ln \beta.$$

It was found that reasonably good results were obtained when using $\theta = 0$ [1].

This paper revisits the simple quadtree, PrivTree and UG algorithms as described above and compares the accuracy results of range queries to those obtained by Zhang. et al. [1] on the same data sets.

IV. Data Sets

This section contains a description of the two data sets used in this study. These are the same data sets used in the study by Zhang et al. [1]. The first data set was a set of latitude and longitude coordinates of trajectories of taxis in Beijing [4]. The second data set contained latitude and longitude coordinates of users of the Gowalla social networking website [5].

A. Beijing Taxi Data

The full Beijing taxi data set [4] contained the latitude and longitude coordinates of 10,357 taxis that were recorded over a time period of one week. This data set contained over 17 million data points. In order to make it more manageable for this study, the data set was reduced to 30,000 points as in [1]. A scatter plot of the data points of the reduced data set used in this study can be seen in Fig. 1. The street pattern of Beijing can vaguely be seen in this scatter plot.

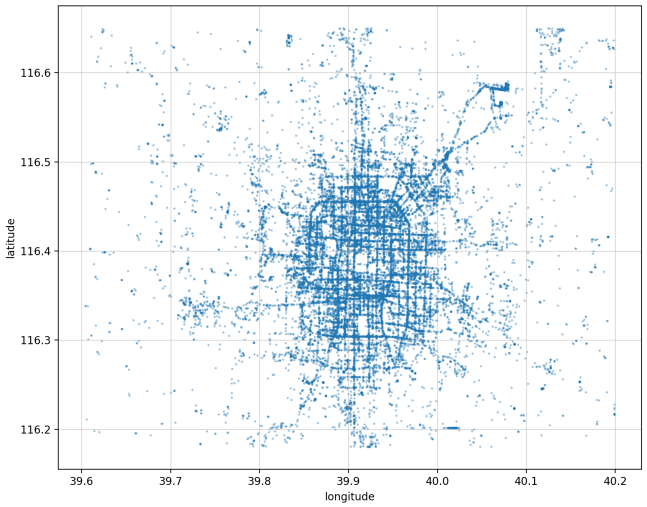

Fig. 1. Scatter Plot of Data Points from The Reduced Beijing Taxi Data Set

B. Gowalla Social Network Data

The full Gowalla social network data set [5] contained the latitude and longitude coordinates of 6,442,890 user check-ins over the period of February 2009 to October 2010. As done in [1], this data set was reduced to make it more manageable for the study. A scatter plot of the reduced data set (561,010 data points) can be seen in Fig. 2. In this scatter plot, the outlines of the continents can be seen vaguely.

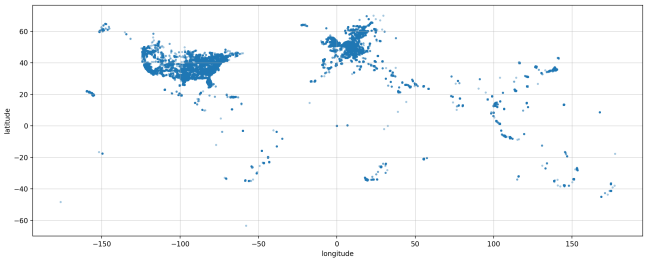

Fig. 2. Scatter Plot of Data Points from The Reduced Gowalla Social Network Data Set

V. Methods

This section describes the methodology followed to obtain the final results. Since the aim of this paper was to replicate and compare the results of Zhang. et al. [1], the methods followed here are closely related.

All the experiments of this study were conducted on a Windows machine with a six-core, 2.6 GHz CPU and 16 GB of RAM at 2,667 MHz . All data set and query processing was done in a Jupyter notebook environment with Python 3.7 using the Numpy and Pandas libraries. All plots were generated in the same environment using the Matplotilib library.

Three of the algorithms used in [1] were also used in this study. These were the UG, simple quadtree, and PrivTree algorithms described previously in algorithms 1, 2, and 3 of section III. All three algorithms were applied to both data sets for spatial decomposition. After spatial decomposition, 1,000 random range queries were generated in the data space Ω of each data set. These queries spanned three size ranges (small, medium, and large) specified in Table I.

TABLE I
Proportion Range of Data Space Ω Covered by Different Range Query Sizes

Range Query Size		
Small	Medium	Large
[0.01%,0.1%)	[0.1%,1%)	[1%,10%)

After the noisy counts $\hat{q}(D)$ of the random range queries were computed for each data set D, the relative error $RE(\hat{q}(D))$ of these counts were calculated using the following equation as in [1] and [3]:

$$RE = RE(\hat{q}(D)) = \frac{\hat{q}(D) - q(D)}{\max\{q(D), \rho\}}$$

Where $q(D)$ is the true count of data points of D contained in the query domain q, $\hat{q}(D)$ is the noisy count of data points of D returned by the query, and ρ is a smoothing parameter equal to 0.1% of the cardinality N of D that prevents division by zero.

In all experiments, the parameters of the algorithms were selected as follows in order to satisfy ϵ-DP as described in section III.

A tree depth limit of $h = 150$ was selected for the simple quadtree algorithm along with $\theta = 1000$, and $\lambda = \frac{h}{\epsilon}$.

For the UG algorithm, $\lambda = \frac{1}{\epsilon}$ and $m = \sqrt{\frac{N\epsilon}{10}}$ were selected.

Finally, for the PrivTree algorithm, the following parameters were selected:

$$\begin{aligned}
&\beta = 4,\\
&\theta = 0,\\
&\lambda = \frac{2\beta - 1}{\beta - 1} \cdot \frac{1}{\epsilon},\\
&\delta = \lambda \cdot \ln \beta.
\end{aligned}$$

Since the scale of Laplace noise λ was dependent on the privacy budget ϵ for all three algorithms, the relative error was expected to change for different values of ϵ. Six different values of ϵ were tested as in [1]. For each algorithm on each data set, the relative error was calculated for $\epsilon = \{0.05, 0.1, 0.2, 0.4, 0.8, 1.6\}$. The results obtained were then compared to those obtained by Zhang et al. [1] which can be seen in the next section.

VI. Experimental Results and Discussion

This section contains an exposition and discussion of the experimental results obtained in this study.

A. Experimental Results

The relative error results of all three different algorithms (PrivTree, UG, and simple quadtree) used in this study and Zhang et al. [1] can be seen in Fig. 3 and 4. These figures contain results for all three different range query sizes (small, medium, and large) and all six privacy budgets ϵ. Fig. 3 contains the results for the Beijing taxi data set and Fig. 4 contains the results for the Gowalla social network data set.

In order to get a sense of the spatial decomposition performed by each of the three different algorithms on the two data sets, the grid of the UG algorithm and the domains of the leaf nodes for the PrivTree and simple quadtree algorithms are shown in Fig. 5 through 10.

Fig. 5, 6, and 7 show the spatial decomposition of the Gowalla social network data set whereas Fig. 8, 9, and 10 show the spatial decomposition of the Beijing Taxi data set.

The spatial decompositions for all combinations of data sets and ϵ could not be shown in order to keep the paper brief. The spatial decomposition figures shown here are for $\epsilon = 1.6$ for all hierarchical decomposition methods (simple quadtree and PrivTree). For the UG algorithm, spatial decompositions with $\epsilon = 0.05$ and $\epsilon = 0.4$ are shown for the Gowalla social network and Beijing taxi data sets respectively.

B. Discussion

Simple Quadtree Algorithm: From the plots of relative error in Fig. 3 and 4 it is clear that the simple quadtree algorithm produced the highest errors out of all the algorithms tested in this study. The relative error for the simple tree was between one and two orders of magnitude larger than any of the other algorithms for all cases tested. One reason for this is due to the simpe quadtree algorithm having a tree depth limit h. This limit prevents the tree from growing into areas where the data is more densely distributed resulting in higher errors in range queries. These areas of dense data points that were not divided into sub-domains can be seen in Fig. 6 and 9. The simple qudtree algorithm was not used in Zhang et al., so no comparison was made. The discussion that follows is concerned with the comparison of results obtained in this study to those obtained by Zhang et al. [1] with the UG and PrivTree algorithms on different data sets.

Beijing Taxi Data Set: For small and medium query sizes on the Beijing taxi data set, results from this study were consistent with those of Zhang et al. for $\epsilon \geq 0.8$. Relative error results were less similar for lower ϵ values.

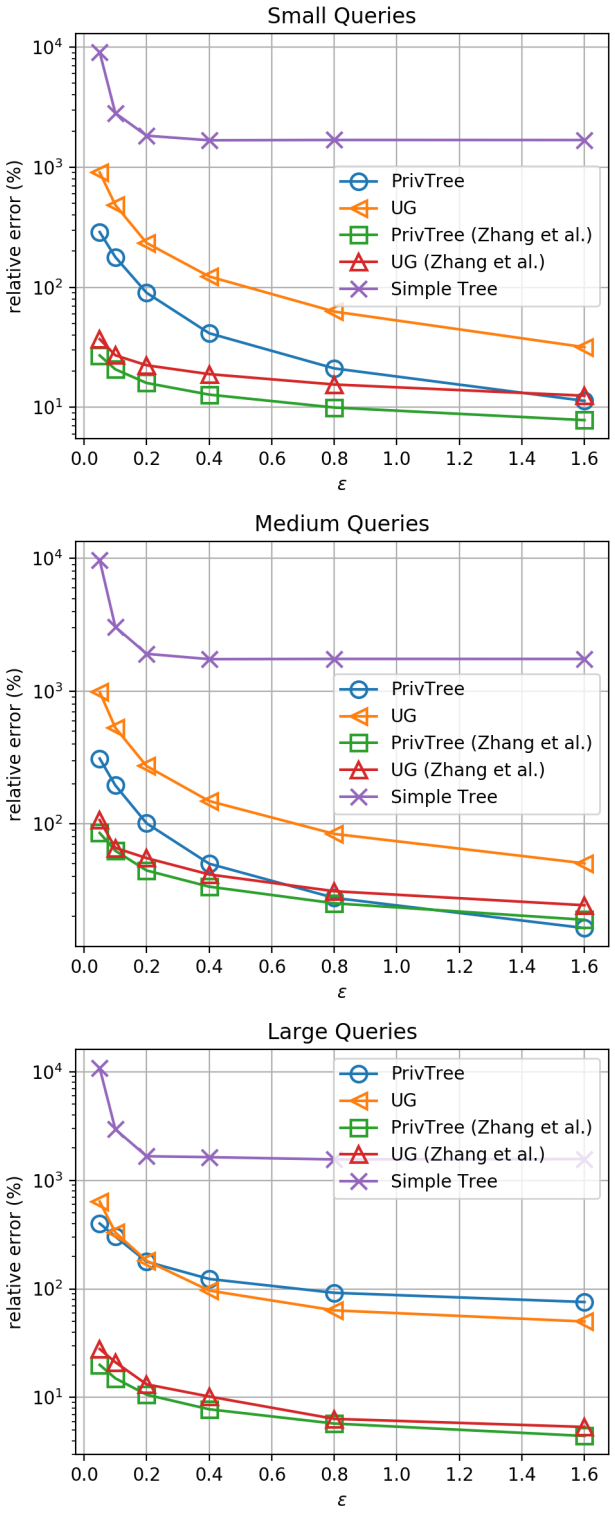

Fig. 3. Relative Error Results from PrivTree, UG, and Simple Quadtree Algorithms of This Study and Zhang et al. [1] for Different Range Query Sizes and Privacy Budgets on Beijing Taxi Data Set

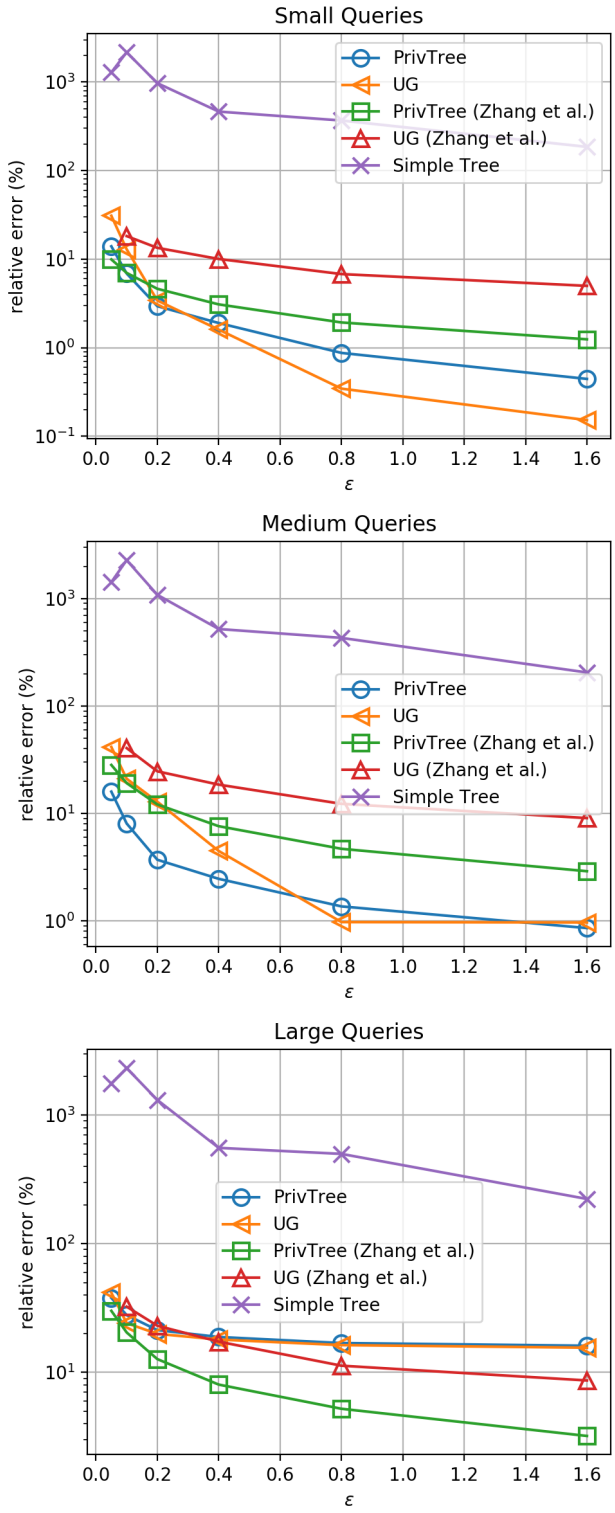

Fig. 4. Relative Error Results from PrivTree, UG, and Simple Quadtree Algorithms of This Study and Zhang et al. [1] for Different Range Query Sizes and Privacy Budgets on Gowalla Social Network Data Set

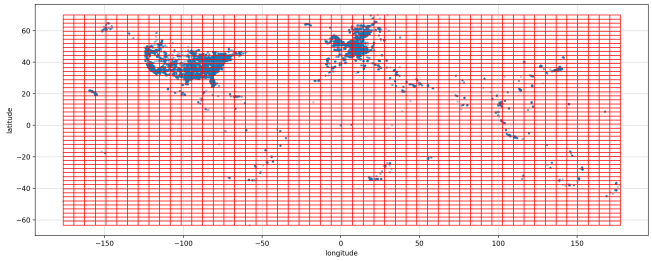

Fig. 5. ϵ-DP Spatial Decomposition of The Gowalla Social Network Data Set Using The UG Algorithm with $\epsilon = 0.05$

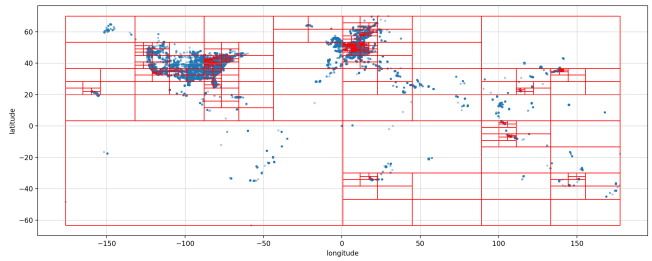

Fig. 6. ϵ-DP Spatial Decomposition of The Gowalla Social Network Data Set Using The Simple Quadtree Algorithm with $\epsilon = 1.6$

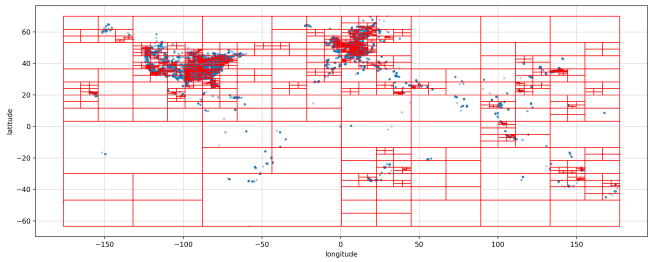

Fig. 7. ϵ-DP Spatial Decomposition of The Gowalla Social Network Data Set Using The PrivTree Algorithm with $\epsilon = 1.6$

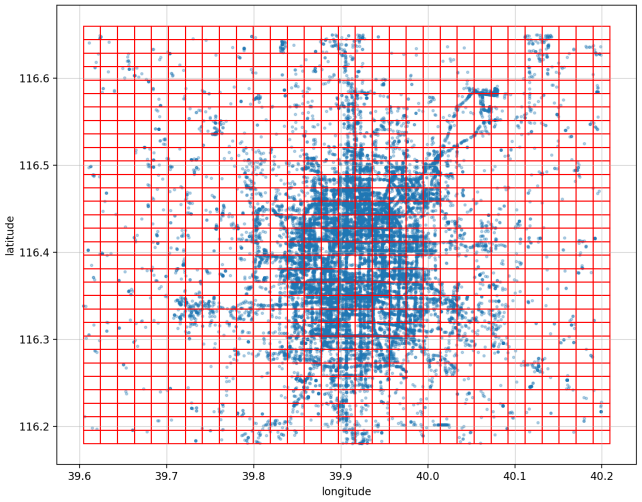

Fig. 8. ϵ-DP Spatial Decomposition of The Beijing Taxi Data Set Using The UG Algorithm with $\epsilon = 0.4$

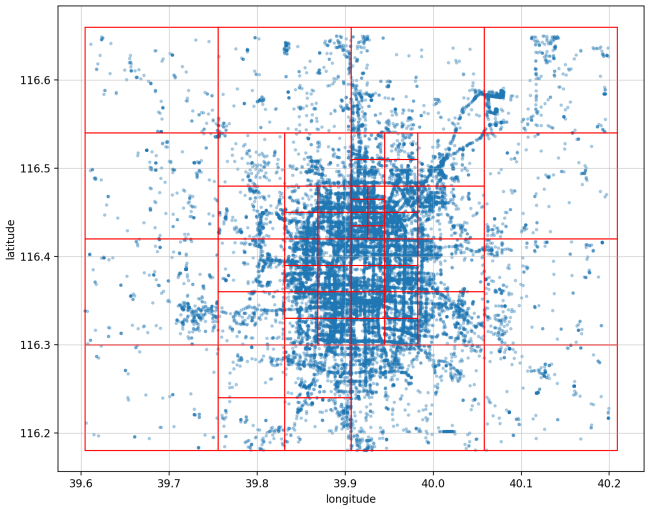

Fig. 9. ϵ-DP Spatial Decomposition of The Beijing Taxi Data Set Using The Simple Quadtree Algorithm with $\epsilon = 1.6$

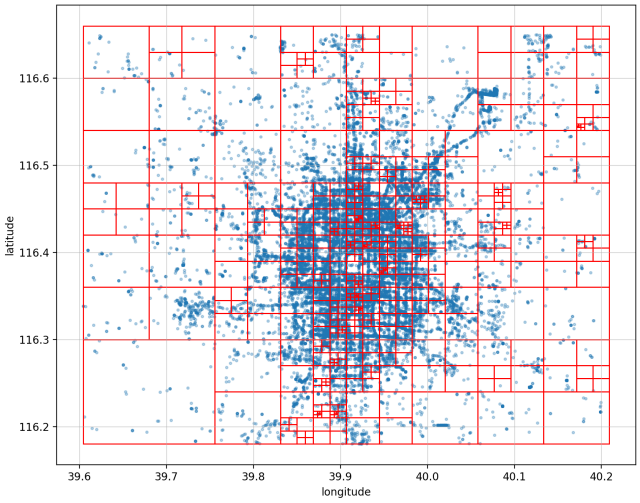

Fig. 10. ϵ-DP Spatial Decomposition of The Beijing Taxi Data Set Using The PrivTree Algorithm with $\epsilon = 1.6$

For large range queries, the relative error of the PrivTree algorithm from this study was consistently one order of magnitude larger than the one from Zhang et al. over all ϵ values tested.

For almost all combinations of ϵ values and query sizes tested, the algorithms from Zhang et al. produced smaller relative errors compared to the algorithms from this study. The only exceptions were for the PrivTree algorithm with $\epsilon = 1.6$ for small queries and $\epsilon \geq 0.8$ for medium queries.

From Fig. 3 it can be seen that the most similar results between this study and Zhang et al. were obtained for medium range query sizes.

Gowalla Social Network Data Set: The results obtained on the Gowalla social network data set were in many ways opposite to those obtained on the Beijing taxi data set. For the Gowalla data set, the relative errors produced by the algorithms

in this study were closer to those of Zhang et al. towards the lower ϵ range. The relative errors of this study increased with respect to those of Zhang et al. for higher ϵ values.

For values of $\epsilon \leq 0.2$, the errors produced in this study were very close to those of Zhang. et al. for all query sizes.

The smallest difference in relative errors produced by this study and Zhang et al., were for large query sizes.

The relative errors produced by this study were lower than those produced by Zhang et al. for both small and medium queries. However, Zhang et al. did manage to produce lower erros for large queries.

General: The UG and PrivTree algorithms from this study produced some mixed accuracy results with respect to those of Zhang et al. This can be because of several factors. The first being that not much detail was given about the geometry of the randomly constructed query domains in Zhang et al. apart from the proportion of the data space Ω they took up. The random queries in this study were always square and never rectangular. More similar results might have been obtained for range query domains of different aspect ratios.

Secondly, and perhaps more importantly, the splitting of domains were dependent on a noisy count of data points. This noisy count was in turn dependent on random noise drawn from a Laplace distribution. This random process could have lead to very different domain splitting which could have had an impact on the relative errors produced by the range queries. An example of a domain that was split incorrectly can be seen in the lower left hand (Southwest) corner of Fig. 7. Because of the skew distribution of the Gowalla data set, this corner contains very little data points and should not have been split. Because of the symmetry of the Laplace distribution around its mean (for this study the means was zero), the noise can take on a negative value. This means that the opposite could have also happened where a very dense region was not split. Both cases can lead to different range query results and therefore different relative errors.

For both data sets and all ϵ values tested, the PrivTree algorithm gives the lowest relative errors for all range query sizes with the single exception of small queries on the Gowalla data set. The reason for the superiority of PrivTree over UG could be that its spatial decomposition adapts to the data set, growing more densely into more dense parts of the data set. One of the drawbacks of UG is that the spatial decomposition does not depend on the data set distribution. Therefore, sparse regions of the data might be over-divided whereas denser regions might be under-divided.

VII. Conclusions

This paper was concerned with the challenge of producing accurate answers to range queries on two-dimensional geospatial data sets while still preserving the privacy of the data set participants. To do this, three differentially private algorithms were selected for spatial decomposition of two different data sets. These spatial decomposition results were then processed to answer 1,000 randomly generated range queries of different sizes. Finally, the relative errors produced in answering these range queries were compared to those obtained from Zhang et al. [1] using the same algorithms on the same data sets.

The accuracy results obtained in this study for different algorithms and ϵ values differed in some cases from those of Zhang et al. In some cases the algorithms from this study produced smaller errors, and in some cases larger errors compared to those of Zhang et al. The ϵ values where performance differed were not consistent between the data sets used. These differences in results can be explained by the fact that the algorithms that produce these results were built on random noise generation processes.

Overall, considering the results of this study and Zhang et al., the PrivTree algorithm outperformed both UG and the simple quadtree algorithms for all cases tested with the single exception of small queries on the Gowalla data set. The reason for the superiority of PrivTree over UG could be that its spatial decomposition adapts to the data set, growing more densely into more dense parts of the data set which UG does not do.

The simple quadtree consistently produced the highest relative errors for all cases tested. The reason for this was due to the simple quadtree algorithm having a tree depth limit. This limit prevented the tree from growing properly into areas where the data was more densely distributed. On the other hand, when the depth limit was increased in an attempt to solve this, more noise was added to the final query outputs resulting in higher errors again.

Acknowledgment

This work was funded by the DSI-NICIS National e-Science Postgraduate Teaching and Training Platform (NEPTTP).

References

[1] J. Zhang, X. Xiaokui, and X. Xing, "Privtree: A differentially private algorithm for hierarchical decompositions," In Proceedings of the 2016 International Conference on Management of Data, 2016, pp. 155-170.

[2] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen and T. Yu, "Differentially Private Spatial Decompositions," 2012 IEEE 28th International Conference on Data Engineering, Washington, DC, 2012, pp. 20-31, doi: 10.1109/ICDE.2012.16.

[3] W. Qardaji, W. Yang and N. Li, "Differentially private grids for geospatial data," 2013 IEEE 29th International Conference on Data Engineering (ICDE), Brisbane, QLD, 2013, pp. 757-768, doi: 10.1109/ICDE.2013.6544872.

[4] "T-Drive trajectory sample dataset," Microsoft, 25 June 2020. [Online]. Available: http://research.microsoft.com/apps/pubs/?id=152883

[5] "Gowalla," Stanford University, 25 June 2020. [Online]. Available: http://snap.stanford.edu/data/loc-gowalla.html