Differentially Private Range Counting in Planar Graphs for Spatial Sensing

Abhirup Ghosh[*]
Dept. of Computing
Imperial College London
abhirup.ghosh@imperial.ac.uk

Jiaxin Ding[*]
John Hopcroft Center for Computer Science
Shanghai Jiao Tong University
jiaxinding@sjtu.edu.cn

Rik Sarkar
School of Informatics
University of Edinburgh
rsarkar@inf.ed.ac.uk

Jie Gao
Computer Science
Rutgers University
jg1555@cs.rutgers.edu

***Abstract*—This paper considers the problem of privately reporting counts of events recorded by devices in different regions of the plane. Unlike previous range query methods, our approach is not limited to rectangular ranges. We devise novel hierarchical data structures to answer queries over arbitrary planar graphs. This construction relies on balanced planar separators to represent shortest paths using $O(\log n)$ number of canonical paths, where n is the number of nodes in the graph. Pre-computed sums along these canonical paths allow efficient computations of $1D$ counting range queries along any shortest path. We make use of differential forms together with the $1D$ mechanism to answer $2D$ queries in which a range is a union of faces in the planar graph. The methods are designed such that the range queries could be answered with differential privacy guarantee on any single event, with only a poly-logarithmic error. They also allow private range queries to be performed in a distributed setup. Theoretical and experimental results confirm that the methods are efficient and accurate on real data and incur less error than competing existing methods.**

***Index Terms*—range query, differential privacy, planar graphs, Internet of Things**

I. Introduction

Large scale spatial data can be collected using densely deployed sensors and IoT devices, including traffic sensors, wireless hotspots, roadside units, and surveillance cameras, that detect and capture activities of interest. Examples of such data include location snapshots of a population, events in traffic, occupancy and motion sensing information. These sensors are increasingly connected through efficient network infrastructure and can communicate freely. The large quantities of data from these sources raise challenges on several fronts. First, the analysis and query mechanisms on such spatial data must be efficient in storage and computation. Distributed mechanisms, where possible, can aid both the speed and scalability of these systems and edge-computing has emerged as a popular setup for this kind of computation. Second, spatial data can contain information that is personally identifiable or sensitive in other aspects. Privacy issues of location reports and trajectories of mobile users have long been recognized as a serious concern [1], [18], [29].

This paper considers the problem of counting range queries in a spatial domain – given a spatial range R in the plane, find the number of events recorded by sensing devices within R. In our setting, we consider an embedded planar graph G in the domain and query ranges in consideration are either along paths in G or as sets of faces in G. This approach makes the proposed method widely applicable across many kinds of geographical data that are given in terms of planar graphs such as maps. Administrative or natural boundaries based on postal zip codes, electoral districts, police precincts, etc., provide natural planar subdivisions of the plane. Statistical queries and analysis based on such subdivisions are common in demographic studies, since administrative decisions and strategies are based on these subdivisions. In addition, the sensing system itself may create subdivisions based on proximity. For example, cellular base stations divide the plane into hexagonal Voronoi cells. Such graphs are convenient as they do not require precise localization and an event or object can be assigned simply to the cell of the nearest base station. In other systems, sensors may collaboratively track objects [2] and mutually agree on a local subdivision graph [12].

There are two types of queries that are of interest:

- *Counting along a $1D$ path*. Certain paths in the graph G are of natural interest for counting range queries. For example, a path may represent the border between two administrative divisions, where the number of people or vehicles crossing can be of interest; or, a path in G may represent a road, a bus route or travel plan and one may want to know the number of emergency events on the route. In this paper, we consider queries along shortest paths in G.
- *Counting in a 2D range*. The common types of $2D$ ranges consist of the connected sets of faces of the planar graph (see Fig. 1(a)). In this paper, we consider queries on ranges with $O(1)$ shortest paths on boundary.

Such query engines have privacy concerns across applications. For example, transport demand can be estimated by aggregating taxi pick-ups and drop-offs along roads, but can also reveal an individual's actions to a well informed adversary. In another application, users often provide their residential addresses to avail certain online services. The service provider or a third party company may use the data to extract knowledge

J. Gao would like to acknowledge supports from NSF CNS-1618391, DMS-1737812, OAC-1939459. A. Ghosh would like to acknowledge supports from the European Unions Horizon 2020 research and innovation programme through the HBP SGA1, GA No 720270 project.

[*] Abhirup and Jiaxin are co-first authors.

such as spatial penetration, but run the risk of exposing the participation and behavior of individual users.

In this paper, we present an efficient data structure and query schemes that support range queries in planar graphs with privacy guarantees. Here we want to achieve a good trade-off between accuracy (the range query answers are approximately accurate) and privacy (that the knowledge of the existence of a single event protected with a probability). Let us first review the state of art for range queries and differential privacy and then present our contributions.

A. Related Work

Spatial Range Queries. Efficient range queries have been extensively studied in computational geometry [8], [27] the *orthogonal range searching*, in which the query ranges are specified by d-dimensional axis-aligned rectangles; and *simplex range searching*, in which a range is a d-dimensional simplex (e.g., a triangle in $2D$).

In orthogonal range searching, the general philosophy is to build a hierarchical space decomposition scheme (quadtree, kd-tree, etc.) such that the query values are pre-computed for carefully selected ranges (called canonical ranges) in the space decomposition. For a given range query, these canonical ranges are used to build up the query answer. The best performance is obtained by balancing two parameters: the number of pre-computed canonical ranges (the storage size of the data structure) and the number of canonical ranges needed to answer a query (query cost/time). For n points in $\mathbb{R}^d$, the best bound for orthogonal range query in $\mathbb{R}^d$ achieves linear space and $O(\log^{d-1} n)$ query time (e.g., using d-dimensional range tree). Simplex range searching is much more challenging and bounds are typically worse. For any linear size data structure simplex range query has to take $\Omega(n^{1-1/d})$ time [4] and there are algorithms with nearly matching upper bounds [21]. Alternatively, there are algorithms with polylogarithmic query time the storage has to be super-linear (essentially in $O(n^d)$) [5].

Differentially Private Range Queries. In differential privacy [9], [10], from the answer to a query, it should not be possible to determine with high confidence if a particular item was present in the input dataset. This is achieved by introducing random noise to the query process such that the probability of the observed output does not change significantly (with a probability no more than a factor of e^{ε} for a small ε) due to presence or absence of a particular element in the input. The study of range query with differential privacy was only done for orthogonal ranges in [3], [11], [15], [28] and further refined in [7], [22], [23]. The general strategy in these methods, as with orthogonal range searching, is to precompute perturbed values to canonical ranges which were then used for queries.

B. Our Contribution

As opposed to the above prior works, this paper considers queries on ranges defined on a planar graph, which has never been studied before. The major contribution of this paper is that it presents, for the first time, an efficient range query mechanisms on planar graphs. With a total storage of $O(n \log^2 n)$, where n is the number of vertices of G, the proposed algorithm can answer both $1D$ queries on shortest paths and $2D$ queries on ranges with $O(1)$ shortest paths as boundaries in time $O(\log n)$. Further, the method achieves ε-differential privacy on any recorded event with error bound by $O(\frac{1}{\varepsilon} \log^{3.5} n \cdot \log\log n \cdot \log \frac{1}{\delta})$ with probability at least $1-\delta$. This is the first non-trivial ε-differential privacy result for non-orthogonal range queries. Achieving these results requires novel algorithms and data structures on shortest paths of planar graphs.

Our approach to the problem consists of pre-computed local aggregate values called *partial sums* or p-sums, such that any query can be answered by adding together a small number of p-sums. An additional requirement for effective differential privacy is that any data item should contribute to only a small number of p-sums. The proposed algorithm achieves both properties.

To answer $1D$ range queries along shortest paths, the planar graph G is recursively decomposed using shortest path separators [26]. Here, a separator is a shortest path in G and it decomposes G into two pieces each of size at most $2/3$ the original size. On a shortest path separator, we construct a binary search tree, and at the nodes of the tree store the aggregate count in the subtrees as Type-I p-sums. Next, another hierarchy of random sampling of the vertices is constructed, such that each vertex is promoted to the higher level with probability $1/2$ independent of others. Within each connected component G' created by the separator hierarchy, we identify Type-II p-sums on shortest paths between two random samples at level i with no interior vertices on level i, and i being roughly logarithmic of the size of G'. The success of the scheme lies in this multi-scale choice of p-sums such that any shortest path can be decomposed into $O(\log n)$ of these p-sums.

This idea is extended to $2D$ range queries using differential forms introduced in [25], which turns the counting range query in a $2D$ range, R, to a query along the boundary of R, thus changing the $2D$ range query to the above $1D$ range query problem, and we can apply the same technique as above.

While the theoretical bounds are only proved for shortest paths (Sections III and IV), our experiments show that our methods are efficient and accurate on trajectories and paths obtained from real datasets (Section VI).

II. Preliminaries

This section formalizes the models for sensing, networking, data, privacy and utility.

A. Discrete Data Model and range queries

We discretize the plane as a cell complex [14]. This decomposition can be visualized as a planar graph, where the faces of the planar graph are the cells. See Figure 1(a). A range in this complex is defined in terms of the elements of the cell complex. A $2D$ range is a set of adjoining faces that together form a topological disk. A $1D$ range is essentially a simple path. Given a range R, the range query becomes:

1) $1D$**:** $c(R) = \sum_{e \in R} c(e)$ [Summation over edges in R.]
2) $2D$**:** $c(R) = \sum_{f \in R} c(f)$ [Summation over faces in R.]

This discretization treats the edges or faces as atomic elements that are added over to achieve the sum. We will see in Subsection II-B that this view fits naturally with a distributed sensing and computing approach. In this work, we do not consider the case of continuously changing counts and work with a static snapshot of the data recorded at a particular time.

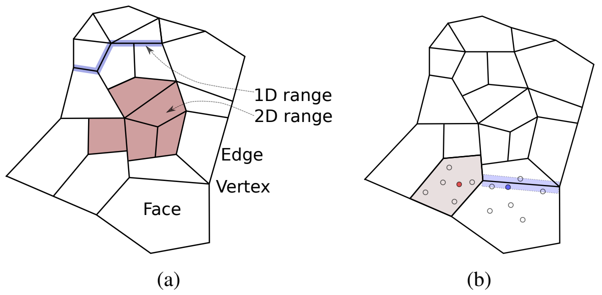

Fig. 1. (a) a planar graph. The shaded region demonstrates $2D$ range. The blue path is a $1D$ range. (b) Sensing devices at the edge, and regions sensed for a $2D$ range and a $1D$ range. The leaders for a face and an edge are shown as shaded.

B. Distributed computation and network model

We assume that the sensors can communicate with the query aggregator and with each other in constant cost using modern communication infrastructure. Further, let us assume that the sensors are aware of the planar subdivision, and they coordinate among themselves for basic tasks. Devices in each cell coordinate to elect a leader for that cell who is responsible for managing data and actions in that cell (Figure 1(b)). An event can be detected and localized by nearby sensors, this information is reported to a suitable leader in the neighborhood.

The rest of the paper takes the simplified view that there is one device – the leader for the cell – corresponding to each cell ($2D$ or $1D$). Moreover, assume that the sensors trust each other to sanitize and transmit data, but they may not trust the aggregator to protect privacy.

C. Differential privacy

The objective is to answer range queries with differential privacy, i.e., from the range query responses, it should not be possible to infer with high confidence if a particular event x has been recorded. Suppose, D and D' are two possible versions of the datasets that differ in the presence of the event x. Such pairs of datasets are called adjacent datasets.

Definition 2.1 (Differential Privacy). *A randomized range query response mechanism M is ε-differentially private if for any two adjacent datasets D and D' and any measurable subset $H \in \mathrm{Range}(M)$, $\Pr[M_D(R) \in H] \leq \exp(\varepsilon) \cdot \Pr[M_{D'}(R) \in H]$, for any range $R \in \mathcal{R}$.*

A differentially private mechanism typically adds noise to the true query answer to obfuscate the precise value. This noise is commonly sampled from a Laplace distribution with zero mean and variance $2b^2$, written as $\mathrm{Lap}(b)$, and its probability density is given as: $P(x|b) = \frac{1}{2b}\exp(-\frac{|x|}{b})$. The mechanism is commonly called as the *Laplace mechanism* [10]. A larger variance (i.e., larger b) implies greater noise in the output which results in greater privacy but reduced utility.

The level of noise is usually determined in terms of *sensitivity*. The sensitivity of a function f, written as Δf, is the largest possible difference in the output of f between any pair of adjacent databases: $\max_{(D,D')}|f(D) - f(D')|$. To achieve ε-differential privacy, a noise of $\mathrm{Lap}(\Delta f/\varepsilon)$ suffices [9].

D. Multiple queries

To support spatial range queries, each cell s can store the count of events, $c(s)$ and while responding to a query add $\mathrm{Lap}(1/\varepsilon)$ noise to make it differentially private. However, with multiple queries, it is possible to infer the true answer to arbitrary precision as the average noise will converge to its mean zero. Therefore, to guard against q queries, differential privacy methods usually increase the noise variance by a factor of q and use $\mathrm{Lap}(q/\varepsilon)$ as the distribution or analogously increase the sensitivity q times.

One way to avoid higher noise is to report the same noisy count without recomputing the noise. Thus it results in $\mathrm{Lap}(1/\varepsilon)$ noise as in the case of a single query.

E. Accuracy loss in range queries

A range consists of multiple cells. Over many cells, the noises can add up, and deviate from the true count. The following result from [3] gives a bound on the deviation:

Lemma 2.2 (Sum of Independent Laplace Distributions). *Given γ_i as independent random variables following Laplace distribution $\mathrm{Lap}(b_i)$. Suppose $Y = \sum_i \gamma_i$. Then, $|Y|$ is at most $O(\sqrt{\sum_i b_i^2}\log(1/\delta))$ with probability at least $1 - \delta$, $0 < \delta < 1$.*

In our context, if a query range consists of m cells, where each cell adds an independent noise from $\mathrm{Lap}(1/\varepsilon)$, then the noise is only bounded by $O(\frac{1}{\varepsilon}\sqrt{m}\log\frac{1}{\delta})$ with probability $1-\delta$. Thus, even if each cell computes a fixed noise, and returns the same answer on each query, the total noise can be excessive for a large range.

F. $1D$ range query and partial sums on $1D$ data

Let us consider a single path P with m edges where each edge contains a value and the queries ask for aggregates over subpaths of P. Now, construct a binary search tree, B, with the edges of P as leaves. Every node in B corresponds to some subpath of P, and it stores the query answer for that subpath as a range, which is called a partial sum, or *p-sum*. Any subpath in P can be represented by at most $\log m$ nodes of B. Thus, a range query now adds only $\log m$ values, each of which is a p-sum. In this case, the tree has $\log m$ levels, and each leaf node contributes to $\log m$ different unique p-sum values, each of which can be queried separately. Thus, each value on edges of P have to answer $\log m$ different queries and need $\mathrm{Lap}(\frac{\log m}{\varepsilon})$ noise.

Thus, in [3], $\mathrm{Lap}(\frac{\log m}{\varepsilon})$ noise is added to each p-sum node of B to answer a range query using B. Any such answer adds together at most $\log m$ p-sums, each of which adds an independent noise from $\mathrm{Lap}(\frac{\log m}{\varepsilon})$. Thus, the error due to noise is bounded by $O(\frac{1}{\varepsilon} \cdot (\log m)^{1.5} \cdot \log \frac{1}{\delta})$ with probability at least $1 - \delta$.

The main insight here is that the noise depends on balancing two different quantities:

- x : The number of p-sum values to which each item contributes;
- y : The number of p-sum values needed to answer a range query R.

Each p-sum needs a noise of distribution $\mathrm{Lap}(x/\varepsilon)$ added to its pre-computed value. This ensures that ε-differential privacy with $O(\frac{x\sqrt{y}}{\varepsilon})$ error is achieved with high probability.

The same idea was independently discovered by several groups of researchers with the same upper bounds [11], [19], [28], and studied in [6], [7], [22]. Further, it is shown [11] that the error has a lower bound of $\Omega(\frac{1}{\varepsilon} \log m)$. Thus the upper bound is nearly tight. Further, the $1D$ counting using p-sum generalizes to multi-dimensional rectangle queries [3], [28].

This paper goes beyond strict rectangular range queries and consider a much more general set of ranges, defined using a planar graph in the domain. The main technical challenge is to carefully re-define p-sums that support efficient range queries.

III. Private Spatial Sensing and Path Queries on Planar Graphs

Let us first discuss the $1D$ range query along shortest paths on a planar graph – given a planar graph $G = (V, E)$, on which each edge (u, v) carries a nonnegative integer value $s(u, v)$, query ranges are shortest paths in G. Say, the set of all shortest paths in G is $\mathcal{R}$ and a range $P(x, y) \in \mathcal{R}$ is the set of edges along the shortest path from x to y. Without loss of generality, we assume that between any two vertices there is a unique shortest path[1]. Thus, there are $\binom{n}{2}$ ranges to consider and one edge can belong to $\Omega(n^2)$ ranges. Each range or shortest path can consist of $\Omega(n)$ edges.

Naïve differential privacy mechanisms incur high error:

- *Simple Mechanism I:* For a query over a shortest path $P(x, y)$, add $\mathrm{Lap}(\frac{n^2}{\varepsilon})$ noise to the actual count for the range. Since an edge could stay on $\Omega(n^2)$ many shortest paths, the noise needs to be proportional to $\Omega(n^2)$ to achieve ε-differential privacy. The range query error in this case is $O(\frac{n^2}{\varepsilon})$.
- *Simple Mechanism II (Local Privacy):* For each edge (u, v), add $\mathrm{Lap}(\frac{1}{\varepsilon})$ noise to its true value $s(u, v)$. To answer a query over $P(x, y)$, sum the noisy counts of the edges in $P(x, y)$. Since a path may have length $\Omega(n)$, this mechanism achieves ε-differential privacy with error $O(\frac{1}{\varepsilon}\sqrt{n})$. This mechanism achieves local differential privacy since noise is added to each sensor data separately.

[1]This could be achieved by adding small perturbation on edge weights.

In both cases, the error is polynomial in n. In the following, we will use the balanced separator hierarchy to reduce the error to be logarithmic in n. The key is to identify a proper set of p-sums. The overall strategy is sketched in Algorithm 1.

Algorithm 1 Create spatial p-sums on a planar graph G

```
1: Recursively partition $G$ into connected components in $O(\log n)$ levels using *balanced shortest path separators* (Section III-A). Construct Type-I $p$-sums (binary tree) along the shortest path separators.
2: In each connected component in the graph decomposition, construct Type-II $p$-sums on *canonical shortest paths*, shortest paths between certain node pairs. (Section III-B).
```

Given a query on a shortest path in G, it can be decomposed into a small number of segments, each of which run along either shortest path separators or canonical shortest paths (Section III-D). Since p-sum hierarchies are constructed for both of them, the range query can be answered using a small number of p-sum values. Additionally, this hierarchic structure also satisfies the property that an edge contributes to a small number of p-sums. Now, let us start with planar separators and recursive partitioning.

A. Planar separators and hierarchic decomposition

A classical result in graph theory is that a planar graph admits a small sized balanced separator. Lipton and Tarjan [20] showed that one can remove $O(\sqrt{n})$ vertices, called a separator Z, such that the graph G of n vertices is decomposed into two subgraphs A and B each with at most $2n/3$ vertices and with no edges connecting A and B.

We use separators that are constructed using shortest paths. Specifically, from an arbitrary node as the root r, a shortest path tree called $H(r)$ is constructed. A *fundamental cycle* is a cycle that contains a single non-tree edge (u, v) and the shortest paths connecting u, v to the root r of the tree $H(r)$. Removing a fundamental cycle decomposes a planar graph into at least two remaining subgraphs. In any graph, there exists a fundamental cycle Z such that each subgraph obtained by removing Z has at most $2n/3$ vertices.

The above observation was described in Lemma 2 of [20], Lemma 2.3 of [26][2] and also evaluated in [16]. This operation can be implemented in linear time. In the above papers, the goal was to get a separator of size $O(\sqrt{n})$. In the current context, we do not require using a small separator. Thus, the balanced separator by one fundamental cycle as described in [16] suffices.

Recursive partitioning using planar separators. We run the balanced separator recursively. That is, start with a planar graph G, remove the separator $Z(G)$, and G is decomposed into two pieces $A(G)$ and $B(G)$, each of which has at most $2|n(G)|/3$ vertices, where $n(G)$ is the number of vertices in

[2]In [26] the authors described removing three shortest paths from r to three vertices of a triangle in G such that all the partitioned pieces have size strictly smaller than $n/2$.

G. Now for each subgraph $A(G)$ and $B(G)$, we find their balanced separators and further decompose them recursively. This will eventually create a hierarchical balanced decomposition of G. This decomposition can be abstracted by a tree Φ with depth of $O(\log n)$. Each node α in this decomposition corresponds to a subgraph denoted as $G(\alpha)$ of G. We denote the number of vertices of $G(\alpha)$ by $n(\alpha)$ in abbreviation of $n(G(\alpha))$ and the same goes with the following denotations. The separator that further partitions $G(\alpha)$ is written as $Z(\alpha)$, and it produces subgraphs $A(\alpha)$, $B(\alpha)$. The subgraph at a node α at j hops from the root of Φ has size $n(\alpha) \leq (2/3)^j n$. See Figure 2 for demonstration.

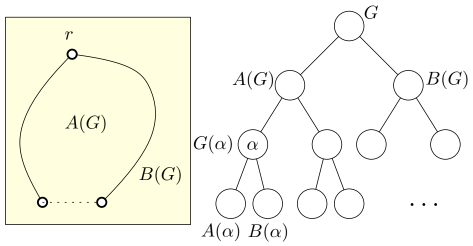

Fig. 2. The balanced separator hierarchy.

Observation 3.1. *Properties of hierarchic decomposition:*

1) *Each edge stays on at most one separator in the hierarchical balanced decomposition.*
2) *Each edge stays in at most $O(\log n)$ subgraphs of the decomposition Φ.*

The balanced separator hierarchy helps us to answer queries on a given shortest path $P(x, y)$.

B. Canonical paths and p-sums

We define two kinds of p-sums:

- Type-I p-sums on separators,
- Type-II p-sums that stay within a subgraph $G(\alpha)$ for a node α in the balanced separator hierarchy.

On the separator $Z(G)$, define a binary interval tree $I(r, u)$ and $I(r, v)$ on $P(r, u)$ and $P(r, v)$ respectively, similar to the $1D$ design in [3]. Each interval of the tree $I(r, u)$ is a p-sum. The tree $I(r, v)$ is similar. For a segment $P(z_1, z_2)$ within the separator $Z(G)$, the number of p-sum needed to calculate the value of the segment is at most $O(\log |P(z_1, z_2)|)$. We do the same on all separators in the hierarchy. The total number of Type-I p-sum is $O(n)$, where n is the number of vertices in the graph G. Since any edge is on at most one separator, an edge belongs to at most $O(\log n)$ Type-I p-sums.

The second family of p-sums is defined within a subgraph $G(\alpha)$ for each node α in the hierarchy Φ. Since a subgraph $G(\alpha)$ could be large, random sampling is used to selectively calculate p-sum on some shortest paths in $G(\alpha)$.

Random Sample Hierarchy. Starting with all the nodes in level 0, i.e., $V_0 = V$, each node in level-i, V_i is promoted to V_{i+1} with probability $1/2$ independently. The hierarchy has the maximum level h and $h = O(\log n)$ with high probability. Each node has a probability of $1/2^i$ to be promoted to V_i.

Now, consider a subgraph $G(\alpha)$ in the separator hierarchy. A shortest path of two nodes u, v in $G(\alpha)$ is a *canonical path* if the following conditions are met.

- $u, v \in V_i$, with $\log n(\alpha) - q \leq i \leq \log n(\alpha)$, for a value $q = \log \log n$.
- On the shortest path $P(u, v)$, no other vertices are in V_i.

It is possible that a shortest path $P(u, v)$ is identified as canonical paths for two indices i and j. In that case, only one copy corresponding to the highest index is kept, which is called the level of the canonical path. A Type-II p-sum is defined for each canonical path.

C. Analysis of Type-II p-sum and edge sensitivity

Lemma 3.2. *Given a subgraph $G(\alpha)$ for a node α in the separator hierarchy Φ, the expected number of canonical paths defined in $G(\alpha)$ is upper bounded by $O(\log^2 n)$.*

Proof: Given two nodes w, y in $G(\alpha)$, a path $P(w, y)$ is a canonical path at level i when both w and y are promoted to V_i and the nodes on the shortest path between w, y are not promoted to V_i. Suppose $P(w, y)$ has length ℓ. Denote by $\Pr_i[w, y]$ the probability for this to happen; then:

$$\Pr_i[w, y] = \frac{1}{2^i} \cdot \frac{1}{2^i} \cdot (1 - \frac{1}{2^i})^{\ell - 1} \leq \frac{1}{2^{2i}}.$$

Summing up for all possible $\log n(\alpha) - q \leq i \leq \log n(\alpha)$:

$$\sum_{i=\log n(\alpha)-q}^{\log n(\alpha)} \Pr_i[w, y] \leq \frac{2^{2q}}{n(\alpha)^2} \sum_{j=0}^{q} \frac{1}{2^{2j}} = O(\frac{2^{2q}}{n(\alpha)^2})$$

Since there are $n(\alpha)^2$ pairs of nodes in $G(\alpha)$, the total number of canonical paths in $G(\alpha)$ is $O(2^{2q}) = O(\log^2 n)$. □

Theorem 3.3. *The total number of Type-II p-sums is $O(n \log^2 n)$ in G, where n is the number of vertices of the planar graph G.*

The proof comes from Lemma 3.2 and the fact that there are $O(n)$ nodes in the hierarchy Φ.

Next, we argue that each edge only contributes to $O(\text{polylog}\, n)$ p-sums as defined above.

Lemma 3.4. *Each edge (u, v) stays on at most $O(\log^3 n \log \log n)$ p-sums on average.*

Proof: Each edge (u, v) stays on only one separator. Thus the number of Type-I p-sum that contains (u, v) is bounded by $O(\log n)$.

Now, we bound the average number of p-sums that contain (u, v) within one subgraph $G(\alpha)$. For that, we first show that the total length of the p-sums defined in $G(\alpha)$ is at most $O(n(\alpha) \log^2 n \log \log n)$. Thus one edge stays in $O(\log^2 n \log \log n)$ p-sums, on average. Since an edge only stays in $O(\log n)$ nodes of Φ, the claim follows.

For a path $P(w, y)$ denote by $L_i[w, y]$ the expected contribution towards the expected total length of p-sum paths, for index i. Recall ℓ is the length of path $P(w, y)$.

$$L_i[w, y] = \Pr_i[w, y] \cdot \ell = \frac{1}{2^i} \cdot \frac{1}{2^i} \cdot (1 - \frac{1}{2^i})^{\ell - 1} \cdot \ell$$

We analyse two cases: If $\ell > 2^i$, we have

$$L_i[w,y] \leq \frac{1}{2^i}\cdot\frac{1}{2^i}\cdot(1-\frac{1}{2^i})^{2^i}\cdot\ell \leq \frac{1}{2^{2i}}\cdot\frac{1}{e}\cdot\ell$$

Observe that $\ell \leq n(\alpha)$ and $2^i \geq n(\alpha)/2^q$.

$$L_i[w,y] \leq \frac{2^{2q}}{n(\alpha)^2}\frac{1}{e}\cdot n(\alpha) = \frac{2^{2q}}{e}\cdot\frac{1}{n(\alpha)}.$$

If $\ell \leq 2^i$,

$$L_i[w,y] \leq \frac{1}{2^{2i}}\cdot\ell \leq \frac{1}{2^i}.$$

Therefore, the total length of p-sum paths is

$$\sum_w\sum_i\sum_y L_i(w,y)$$
$$=\sum_w\sum_i\Big(\sum_{y,\ell\leq 2^i} L_i(w,y) + \sum_{y,\ell>2^i} L_i(w,y)\Big)$$

Define $\mathbb{I}[\ell \leq 2^i] = 1$ if $\ell \leq 2^i$ and 0 otherwise. Similarly, define $\mathbb{I}[\ell > 2^i] = 1$ if $\ell > 2^i$ and 0 otherwise.

The first term gives

$$\sum_w\sum_i\sum_y \mathbb{I}[\ell \leq 2^i]\cdot\frac{1}{2^i} \leq \sum_w\sum_i n(\alpha)\frac{1}{2^i} \leq 2n(\alpha)2^q$$

The second term gives

$$\sum_w\sum_i\sum_y \mathbb{I}[\ell > 2^i]\frac{2^{2q}}{e}\cdot\frac{1}{n(\alpha)} \leq \sum_w\sum_i\frac{2^{2q}}{e} = n(\alpha)\frac{q2^{2q}}{e}$$

Therefore, the total sum of the length of all Type-II p-sums is in the order of $O(n(\alpha)\log^2 n\log\log n)$, thus each edge stays on $O(\log^2 n\log\log n)$ Type-II p-sums in one subgraph $G(\alpha)$. Recall that each edge stays in at most $O(\log n)$ subgraphs corresponding to the nodes in the decomposition Φ. This gives a bound of $O(\log^3 n\log\log n)$ on the number of p-sums that an edge contributes to, on average. □

The above bound is for the number of contributed p-sums for each edge, on average. The worst-case bound depends a lot on the graph structure. For example, if an edge is a bridge between two large components (i.e., with high betweenness centrality), it could contribute to a large number of p-sums. Although, in reality, such edges are likely to be selected as parts of separators and thus not contribute to many p-sums.

D. Answering Queries

Now we show that the pre-computed p-sums can be used to answer queries.

Lemma 3.5. *Each shortest path query can be answered by using at most $O(\log n)$ p-sums.*

Proof: This constructive proof identifies how to find $O(\log n)$ p-sums to answer a query $P(x,y)$.

Consider the separator $Z(G)$, consisting of two shortest paths $P(r,u)$, $P(r,v)$ and one edge (u,v). We first argue that the path $P(x,y)$ cannot visit a shortest path $P(r,u)$ multiple times. Assume otherwise, suppose the path $P(x,y)$ leaves $P(r,u)$ at a vertex w and later visits $P(r,u)$ at a vertex w'.

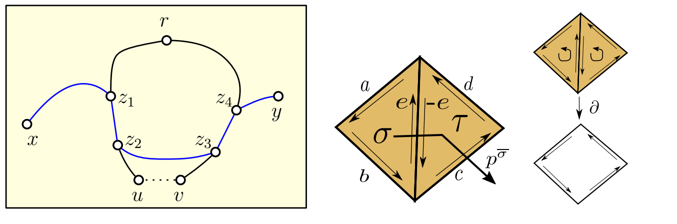

Fig. 3. **(a)** The shortest path $P(x,y)$ visits the separator $Z(G)$ at most twice. **(b)** Here, w events occur at σ and zero event at τ. As $p^{\overline{\sigma}}$ crosses e and c, $\theta(e)$ and $\theta(c)$ are updated to $\theta(e)-w$ and $\theta(c)-w$ respectively. Therefore, with appropriate range orientation $d\theta(-\sigma)=w$ and $d\theta(\tau)=0$. **(c)** The Boundary operation on the counter-clockwise oriented faces produces the edges enclosing the region with consistent orientation.

The shortest path from w to w' shall precisely coincide with $P(r,u)$, since the shortest path from w to w' is unique.

Therefore, the shortest path $P(x,y)$ can be decomposed into at most 5 segments (see Figure 3(a)) at vertices z_1, z_2, z_3, z_4 such that each segment stays either entirely in $A(G)$ or $B(G)$, or entirely on the separator $Z(G)$ (on $P(r,u)$ or $P(r,v)$).

For the two segments $P(z_1,z_2)$, $P(z_3,z_4)$ on $Z(G)$, we use Type-I p-sum to calculate their contribution to the query. This uses at most $O(\log n)$ Type-I p-sums.

Now we handle a segment entirely in $A(G)$ or $B(G)$. We only explain how to handle $P(x,z_1)$. The other two cases are similar.

Suppose $P(x,z_1)$ stays in $A(G)$ with n_1 vertices. $n_1 \leq 2n/3$. On the shortest path $P(x,z_1)$, denote by the highest level j such that there are nodes of $P(x,z_1)$ in V_j. If $\log n_1 - q \leq j \leq \log n_1$, we identify a chain of vertices $w_1^{(j)}, w_2^{(j)}, \cdots, w_f^{(j)} \in V_j$ on $P(x,z_1)$ such that the segments between consecutive pairs are Type-II p-sums of level j within $A(G)$. We call this chain a level j chain. Notice that this chain may have only one vertex.

Apart from the level j chain, we are left with two shortest paths $P(x,w_1^{(j)})$ and $P(w_f^{(j)},z_1)$ at the two ends. We consider level $j-1$ and V_{j-1} in the same manner. Recall that nodes $w_1^{(j)}, w_f^{(j)}$ in V_j stays in V_{j-1} as well, by our random sampling hierarchy. We could identify two level $j-1$ chains: one that ends at $w_1^{(j)}$, and one that starts from $w_f^{(j)}$. Continue with $j-2$ until level $\log n - q$. This way, we have a chain of p-sum with level starting from $\log n_1 - q$, going up monotonically to j and then going monotonically down to $\log n_1 - q$. Before this chain, there is a 'head' path $P(x,b_1)$ with the last endpoint b_1 on level $\log n_1 - q$. After this chain, there is a 'tail' path $P(b_2,z_1)$ with b_2 on level $\log n_1 - q$. On the head or tail path, there cannot be interior vertices of level $\log n_1 - q$ by definition. Now we claim:

- The chain from b_1 to b_2 using p-sums of level $\log n_1 - q$ to $\log n_1$ has at most $\log n$ p-sums, in expectation.
- The head and tail chains can be recursively handled with p-sums defined in a subgraph $G(\alpha)$ with α being a descendant of $A(G)$ in Φ.

Now we prove the two claims. First, we want to show that the total number of nodes on $P(x,z_1)$ with levels between

$\log n_1 - q$ and $\log n_1$ is at most $\log n$. This is an upper bound on the number of p-sums from b_1 to b_2.

Take a node w on $P(x, z_1)$. The chance that w is promoted to level $\log n_1 - q$ is $1/2^{\log n_1 - q} = 2^q/n_1$. Recall that $P(x, z_1)$ is inside $A(G)$ with n_1 vertices. Thus $|P(x, z_1)| \leq n_1$. The expected total number of nodes on levels between $\log n_1 - q$ and $\log n_1$ is

$$|P(x, z_1)| \cdot \frac{1}{2^{\log n_1 - q}} \leq n_1 \cdot \frac{2^q}{n_1} = 2^q = \log n.$$

For the second claim, we only prove for a head chain $P(x, b_1)$. The same applies for a tail chain. Recall that a head chain $P(x, b_1)$ does not have any nodes of level $k = \log n_1 - q$ (or higher). We argue that $P(x, b_1)$ cannot be too long.

$$\Pr[|P(x, b_1)| \geq \beta n] \leq (1 - \frac{1}{2^k})^{\beta n} \leq (\frac{1}{e})^{3 \cdot 2^{q-1} \beta}$$

Recall that we take $q = \log\log n$. Thus the right hand side gives $o(1/n)$. This says that there are at most 6 such head/tails chains on the original path $P(x, y)$ that will be recursively handled, and they have length at most βn for a small β – say just take $\beta < 1/100$. Therefore, the recursive function on the number of p-sums for $P(x, y)$ is

$$M(n) \leq 6M(\beta n) + 5 \log n.$$

Solving this recurrence gives us $O(\log n)$ p-sums. This finishes the argument. □

Now, we can summarize the main results.

Theorem 3.6. *On a planar graph G of n vertices in which each edge carries a sensing reading. We can preprocess the graph G with total storage $O(n \log^2 n)$ such that we can answer range query on any shortest path by querying $O(\log n)$ p-sums.*

We add a noise of $\text{Lap}(\log n/\varepsilon)$ to each p-sum. And by the differential privacy properties, we have the following theorem.

Theorem 3.7. *The range query along any shortest path can be answered with error $O(\frac{1}{\varepsilon} \log^{3.5} n \log\log n \cdot \log \frac{1}{\delta})$ with probability at least $1 - \delta$, with ε-differential privacy on any event.*

IV. Range Query Algorithms Using p-sum

In many practical scenarios, a $1D$ query path of interest can be represented as a concatenation of several shortest paths in the underlying planar subdivision. In the worst case, a path with m edges contains m shortest paths, but often the number is much smaller. For example, realistic mobility routes may follow shortest paths between waypoints in the road network and could be subject to ranges of queries. In such queries, we can apply the algorithm in Lemma 3.5 separately for each shortest path segment to get the following.

Lemma 4.1. *A range query on a $1D$ path that is concatenation of k shortest paths can be answered using at most $O(k \log n)$ p-sums.*

As each p-sum is subject to $\text{Lap}(\frac{\log n}{\varepsilon})$ noise, we have the following error bound.

Theorem 4.2. *A range query on a $1D$ path that is concatenation of k shortest paths can be answered with error $O(\frac{1}{\varepsilon}\sqrt{k} \log^{3.5} n \log\log n \log \frac{1}{\delta})$ with probability at least $1 - \delta$ with ε-differential privacy on any event.*

$2D$ Range Queries Using Discrete Differential Forms. In many natural settings, the events occur at the faces of the administrative, or city block-wise planar subdivisions of the domain, G and the query ranges of interest are the sets of faces $U \subseteq F$. In practical applications, usually the range boundaries can be represented as a concatenation of several shortest paths in G.

Here, we summarize a technique proposed in [25] to reduce the query on a $2D$ range to a query on its $1D$ range boundary using concepts from discrete differential forms. Following this reduction, all the algorithms and results discussed above applies to $2D$ range queries.

We assume each face in G is consistently orientated in counter-clockwise direction [17] and the edges inherit direction from the faces. The boundary ∂U of a set of faces $U \subseteq F$ is the set of edges that separates U from the rest of the faces $F \backslash U$ (Figure 3(c)).

Let's define a function $d\theta$ on a face, $\sigma \in F$ such that $d\theta(\sigma) = w$, if w events occurred in σ. Therefore, the number of events inside the query range U is $\sum_{\sigma \in U} d\theta(\sigma)$. Consider another function, θ called differential 1-form, on the edges of G with the property that $\theta(-e) = -\theta(e)$ for an edge e . This can be extended to $d\theta$ using the relation $d\theta(\sigma) = \theta(\partial\sigma)$ and further to $d\theta(U) = \theta(\partial U)$.

The implication of this discrete differential form is that the count at any face is incorporated into the count of any cycle with the face in its interior. Thus, given a query for $d\theta$ in a query range U, it suffices to compute θ along its boundary. In [25], θ is computed by constructing a spanning tree on the dual graph of G, and for each face, adding its count to the edges on the path to the exterior face. Other ideas of differential form constructions for geospatial data analysis are discussed in [13].

To support ε differential privacy, the differential form construction can be amended with a Laplace noise addition to the edges of the planar graph with p-sum construction on shortest paths. Following which, the $1D$ range query can be applied to the range boundaries and analogous results will apply as Theorem 4.2. A caveat here is that the noise requirement can be sensitive to the constriction of the differential form. A face, whose path to the exterior face is long imposes greater noise requirement.

V. Distributed Operation

Let's consider that as a part of the infrastructure, p-sum structure is initialized and stored in a distributed way. A node in the interval tree, $I(u, v)$ covering $P(u, v)$ stores the noisy event count for the range $P(u, v)$. A leader is selected among the nodes in $P(u, v)$ using a leader election algorithm to store the information for $I(u, v)$. A node stores this information for all the p-sums it is part of.

When the aggregator knows the canonical paths and the p-sum structure, it can directly ask the suitable leaders for the counts.

When the aggregator does not know the canonical paths and p-sum hierarchy, it requires a distributed cooperation from the local devices. The aggregator sends a query $P(x, y)$ to all the nodes in $P(x, y)$ and asks for Type I p-sums. As no two Type I p-sums overlap, each node $u \in P(x, y)$ can uniquely decide whether it contains a p-sum that overlaps with $P(x, y)$. Suppose, the overlap is S. If the p-sum range covered by the node u is contained in S, but the range of its parent is not contained in S, then u responds to the aggregator. This can be decided either by inter-sensor communication or storing the range of the parent at each node without any inter-sensor communication.

Next, the aggregator queries remaining query segments for Type II p-sums. Along a shortest path, no two canonical paths can overlap unless one of them is a subpath of the other. Therefore, each node again can uniquely decide the maximal Type II p-sum it is part of that overlaps with the query and apply the above protocol to distributedly find the nodes that contain relevant data to answer a query.

VI. Experiments

This section evaluates the privacy mechanisms with empirical data to show that they preserve utility for practical purposes. Highlights of the results are the following.

- Both $1D$ and $2D$ range query schemes outperform competing mechanisms in terms of query accuracy.
- Although the p-sum construction aims to optimize performance for shortest paths, $1D$ queries preserve accuracy even when the query paths are not shortest paths.
- The p-sum structure is both time and space efficient.

A. Experimental Setup

Dataset. Our experiments use real world road network data (Figure 4(a)) in Porto from openstreetmap[3]. The trajectories (Figure 4(i)) are collected also from Porto[4]. They are partitioned according to the trips and collected over 6 months; we ignore the temporal information in the trajectories for our purposes.

Utility measure using range queries. Range queries are fundamental to mining and learning methods and thus used here as the measure of utility. Each experiment uses 1000 uniformly chosen query ranges. If the count of events in a query range with η events is estimated as $\tilde{\eta}$, then the error is $\xi = \frac{|\eta - \tilde{\eta}|}{\eta}$. The relative error is symmetric with respect to addition, i.e., estimating $\eta + \Delta$ and $\eta - \Delta$ incur same error.

Creating the graph G. For the experiments on $1D$ queries, the graph, G, is considered to be the road network as given in Openstreetmap. For $2D$ range queries, the primal graph (G) is created using Delaunay triangulation of 4000 random points in the domain. The range boundary of the $2D$ range is defined using the edges in G.

Creating hierarchical decomposition. Whereas, Section III describes a general algorithm for decomposition, here we implement a simpler mechanism as the domain (Figure 4(a)) is well shaped. We alternatively choose a separator in horizontal and vertical direction that goes through the middle of the current partition of the graph. Then build separators using shortest paths on the graph.

Creating p-sum and answering query. Noisy counts are pre-computed and stored at the p-sum nodes. A node, u, can be part of multiple p-sum paths and they all contribute to the sensitivity of u. A p-sum path of length l contributes to the sensitivity of u by $\log l$, this is due to the p-sum structure – changing the count at u affects $\log l$ partial sums in the tree. Therefore, the sensitivity of u is $\log l_1 + \log l_2 + \cdots + \log l_z$, if u is part of z p-sums with lengths l_1, l_2, etc. An internal node in a p-sum tree has sensitivity as the maximum of its children.

Given a query path $q = u, \cdots, v$ a greedy approach is implemented to find the component p-sums in the following way. Starting from an end point u or v, the largest p-sum covering q is chosen and then applied recursively on the remaining part to be covered in q. The total noise in the output is the sum of the pre-computed noisy counts at the canonical p-sum nodes.

B. Range query along $1D$ paths

Taxi pickups are considered as events on the road segments and they are mapped to the nearest edge in G. Given a path P, a query asks for the count of the pickups along P.

1) Query on $1D$ shortest paths: When P is a shortest path in G, the range queries are accurate (Figure 4(b, c)). With increasing ε the privacy decreases and error in the rage query also decreases. With increasing q, the number of canonical paths increases and so, the error decreases.

2) Query on $1D$ paths: In many applications, the query path may not be a shortest path. Figure 4(f, g), show that the range queries are accurate even when the P comes from the trajectories in Figure 4(i). Note that these trajectories do not always follow the shortest path as shown in Figure 4(e). Therefore, our hierarchical constructions are practically more general than answering only shortest path queries.

3) Comparing with baseline: Local privacy algorithm adds noise to the value at each edge. Figure 4(d, h) show that the p-sum mechanism preserves better utility than local privacy mechanism for $1D$ query ranges as shortest paths and trajectories respectively.

C. $2D$ range query

Here, the events are again taxi pickups. They are considered to occur at the faces of the triangulation graph and the query is to count the number of pickups at a given $2D$ range.

[3]https://www.openstreetmap.org/

[4]https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i

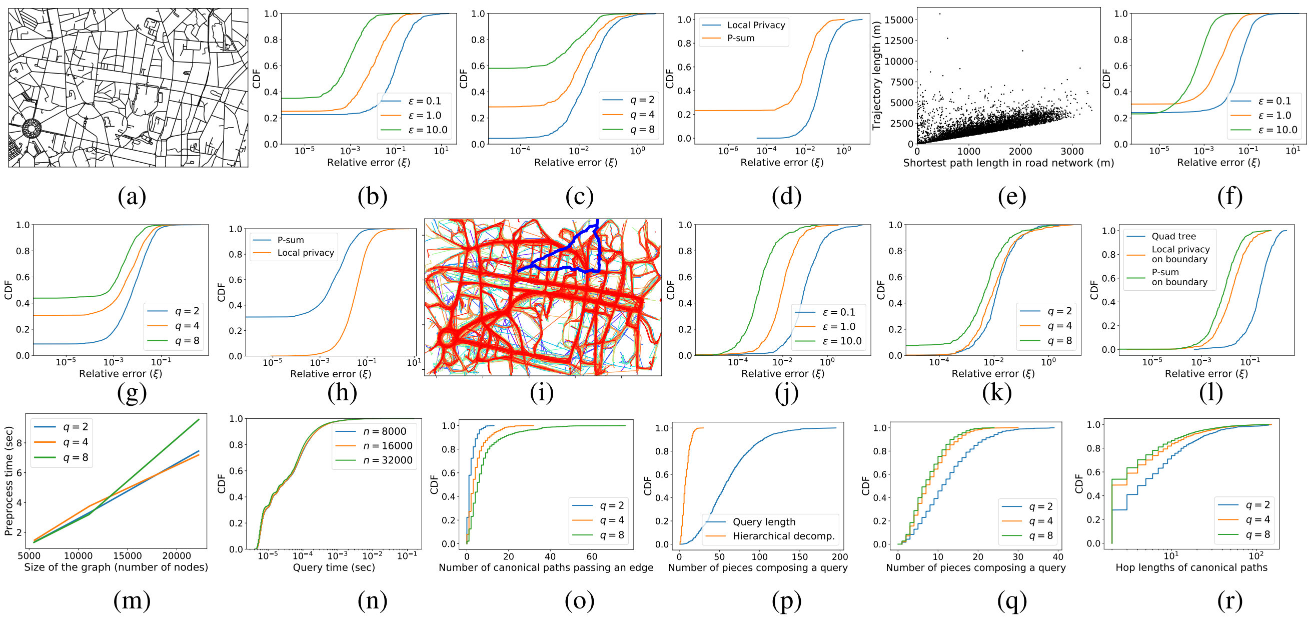

Fig. 4. **(a)** A part of road-map (latitude$-[41.1541°N, 41.1698°N]$ and longitude$-[8.6321°W, 8.6014°W]$) of size roughly $3km \times 3km$ from Porto. It contains 3848 nodes and 4295 edges. There are 5919 taxi trip trajectories in the same region **(i)**. We fix $\varepsilon = 1$ and $q = 4$ suitably. $1D$ **range queries:** **(b, c, d)** show results for queries along shortest paths and **(f, g, h)** show results along the trajectories in (i). **(b, f)** With increasing ε the error decreases. **(c, g)** With increasing q the error decreases. Although with increasing q the number of p-sum increases (o), but the number of composing p-sum for a query decreases(q), and thus overall error decreases. **(d, h)** Our p-sum mechanism preserves better utility than Local privacy. Although the p-sum construction is for shortest path queries, they are produce accurate range query results along real trajectories in **(i)**. These trajectories often follow longer paths than the shortest paths in road network as shown in **(e)**. An example $2D$ range is shown in **(i)**. **(j, k)** Privacy-utility tradeoffs are consistent with $1D$ queries. **(l)** The p-sum mechanism achieves better utility for $2D$ queries than quad tree based mechanism and local privacy mechanism on the range boundary. **(m)** The pre-processing time naturally increases with graph size, but remains small for large graphs. The pre-processing time for $q = 8$ is larger as more canonical paths are created. However, for smaller values of q, the time stays similar. **(n)** Time to query shortest path with p-sum is the same for graphs of different sizes which shows that once p-sums are created, the query time does not depend on the graph size. **(o)** For all values of q, more than 90% of the cases an edge is part of less than 20 canonical paths. So, each edge needs small amount of storage overhead to store reference to canonical paths. With increasing q the overhead naturally increases as more canonical paths are created. **(p)** The number of canonical pieces to compose a query path is smaller than the number of edges in the query path. Here, $q = 2$. **(q)** For larger values of q, more canonical paths are created and therefore, it needs fewer pieces to compose a query path. **(r)** The hop lengths of canonical paths decrease with increasing q because with increasing q more canonical paths are created of smaller lengths (at lower levels).

1) Choosing query ranges: These experiments choose triangular ranges, where the triangle edges are defined using shortest paths in G and the vertices are chosen randomly from the nodes in G. Figure 4(i) shows an example of such a triangle. Varying ε and q, p-sum mechanism produces expected privacy-utility trade-off (Figure 4(j, k)).

2) Comparing mechanisms: The quadtree mechanism initially builds a quad-tree on the faces of G by representing each face as the mean of its vertices and then samples noise from $\mathrm{Lap}(\log n/\varepsilon)$ at each node. Given a query range, it adds the noise at the canonical range nodes to get the total noise in the output. The local privacy mechanism applies local privacy mechanism on the boundary of the range. Figure 4(l) shows that p-sum preserves better privacy than both of them.

D. Efficiency of hierarchical mechanism

1) Time complexity: Figure 4(m, n) show that both the time to pre-process and query are small for large graphs. With increasing q, the pre-processing time increases as the number of canonical paths increases; however, the query time remains similar for different graph sizes.

2) Space complexity: An edge needs to store the canonical paths it is part of. Figure 4(o) shows that this number is low for all the edges. Again the number of canonical pieces composing a shortest path is also low (Figure 4(p, q)), and the lengths of the canonical paths is low (Figure 4(r)) as well, therefore the size of the partial sum structures are not large.

VII. Conclusion

Using carefully constructed p-sum structures, this paper proposes an efficient and differentially private mechanism to answer queries for $1D$ and $2D$ ranges on planar subdivisions of spatial domain. We can easily extend this mechanism to the spatio-temporal dimension by considering a stream of events at each sensor and apply the continuous release mechanism in [3] to each stream. An example adaptation would be to maintain a temporal partial sum at each p-sum we construct.

Our work leaves an open question of range query on mobile objects. This poses a harder challenge as the continuity property of mobility makes the events at nearby edges or faces correlated. Protecting privacy of data with correlation is a much harder problem as the correlation could be exploited by an attacker [24]. This remains future work.

REFERENCES

[1] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security*, CCS '13, pages 901–914, New York, NY, USA, 2013. ACM.

[2] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 150–161. ACM, 2003.

[3] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, Nov. 2011.

[4] B. Chazelle. Lower bounds on the complexity of polytope range searching. *J. Amer. Math. Soc.*, 2(4):637–666, 1989.

[5] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, Feb. 1993.

[6] G. Cormode, T. Kulkarni, and D. Srivastava. Answering range queries under local differential privacy. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pages 1832–1834, New York, NY, USA, 2019. ACM.

[7] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 20–31, Washington, DC, USA, 2012. IEEE Computer Society.

[8] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer Science & Business Media, Mar. 2008.

[9] C. Dwork. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12. Springer-Verlag, 2006.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[11] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724. ACM, 2010.

[12] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanners for routing in mobile networks. *IEEE Journal on Selected Areas in Communications Special issue on Wireless Ad Hoc Networks*, 23(1):174–185, 2005.

[13] A. Ghosh, B. Rozemberczki, S. Ramamoorthy, and R. Sarkar. Topological signatures for fast mobility analysis. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 159–168. ACM, 2018.

[14] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[15] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings VLDB Endowment*, 3(1-2):1021–1032, Sept. 2010.

[16] M. Holzer, F. Schulz, D. Wagner, G. Prasinos, and C. Zaroliagis. Engineering planar separator algorithms. *J. Exp. Algorithmics*, 14:5:1.5–5:1.31, Jan. 2010.

[17] L. C. Kinsey. *Topology of surfaces*. Springer Science & Business Media, 1997.

[18] J. Krumm. A survey of computational location privacy. *Pers. Ubiquit. Comput.*, 13(6):391–399, Aug. 2009.

[19] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, pages 123–134, New York, NY, USA, 2010. ACM.

[20] R. Lipton and R. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, Apr. 1979.

[21] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8(3):315–334, Sept. 1992.

[22] W. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 757–768, Apr. 2013.

[23] W. Qardaji, W. Yang, and N. Li. Understanding hierarchical methods for differentially private histograms. *Proceedings VLDB Endowment*, 6(14):1954–1965, Sept. 2013.

[24] A. Rezaei and J. Gao. On privacy of socially contagious attributes. In *Proceedings of the 19th IEEE International Conference on Data Mining (ICDM'19)*, November 2019.

[25] R. Sarkar and J. Gao. Differential forms for target tracking and aggregate queries in distributed networks. *Networking, IEEE/ACM Transactions on*, 21(4):1159–1172, Aug 2013.

[26] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, Nov. 2004.

[27] C. D. Toth, J. O'Rourke, and J. E. Goodman. Handbook of discrete and computational geometry. 2017.

[28] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on knowledge and data engineering*, 23(8):1200–1214, 2011.

[29] J. Zeng, G. Telang, M. P. Johnson, R. Sarkar, J. Gao, E. Arkin, and J. Mitchell. Mobile r-gather: Distributed geographic clustering for location anonymity. In *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'17)*, pages 7–1. ACM, 2017.