Efficient and Accurate Range Counting on Privacy-preserving Spatial Data Federation

Anonymous Author(s)

Abstract. A spatial data federation is a collection of data owners (*e.g.*, a consortium of taxi companies), and collectively it could provide better location-based services (LBS). For example, car-hailing services over a spatial data federation allow end users to easily pick the best offers. We focus on the range counting queries, which are primitive operations in spatial databases but received little attention in related research, especially considering the privacy requirements from data owners, who are reluctant to disclose their proprietary data. We propose a grouping-based technical framework named FedGroup, which groups data owners without compromising privacy, and achieves superior query accuracy as compared to directly applying existing privacy mechanisms achieving Differential Privacy (DP). Our experimental results also demonstrate that FedGroup runs orders-of-magnitude faster than traditional Secure Multiparty Computation (MPC) based method, and FedGroup even scales to millions of data owners, which is a common setting in the era of ubiquitous mobile devices.

1 Introduction

A federated database system is a collection of multiple cooperating but autonomous database systems [14]. It reflects the real-world situation that the entire database of customer records is usually divided among different companies, and each company owns a distinct share of the entire database. Federated database systems have drawn many research interests [15,4,3] and been deployed in real-world applications. For example, multiple hospitals participate in an alliance to collectively contribute their data for research purposes such as discovering new drugs [3].

In the specific domain of location-based services (LBS) systems, *spatial data federation* considers data federations of *spatial data*, such as locations or trajectories. Similar to general-purposed federated databases, spatial data federation has also seen a wide range of real-world applications. For example, Alibaba's AMap [17] in China provides car-hailing services over a federation of different taxi companies, which enables users to have more flexibility and easily pick the best offers.

To support the aforementioned LBS applications, query processing techniques are needed for a wide range of spatial queries. Among them, *range counting queries* are one of the most important primitive operations. Range counting queries return the count of the spatial objects located within a given range. A

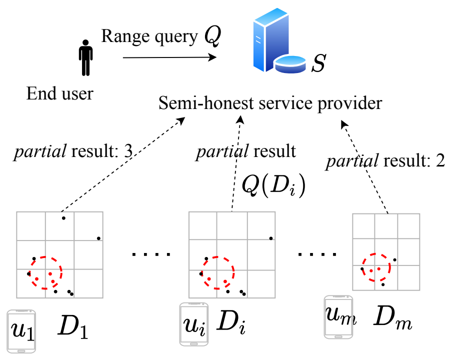


Fig. 1: FPRC problem. There are m data owners, each owner u_i owning a spatial database D_i. An end user issues a range counting query Q over the spatial data federation: $D_1 \cup \ldots D_i \ldots \cup D_m$.

car-hailing federated system may frequently issue the following range counting query: *"how many cars are within 500 meters of the location of a customer?"*

In this paper, we target at the *Federated Privacy-preserving Range Counting (FPRC)* problem (Fig. 1), with a special focus on the large-scale spatial data federations, where the number of data owners is large (*e.g.*, more than 10K). Such large-scale settings have recently drawn more attentions due to the wide adoption of mobile devices. For example, in a decentralized crowdsourcing platform, each data owner is a mobile device, and each user holds his/her own data and is not willing to disclose any unprotected private data to other parties. In such applications, the number of data owners could easily scale up to the order of millions.

It is challenging to support range counting queries over a spatial data federation, because each data owner is often reluctant to disclose its proprietary data. Directly applying traditional Secure Multiparty Computation (MPC) techniques or Differential Privacy (DP) solutions would either be too computationally expensive, or bring too much noise to outweigh any meaningful results for the LBS applications. As demonstrated by our experiments in Sec. 4, the MPC baseline is orders-of-magnitude slower than our solution, requiring more than 13 hours when the input size is large, while the DP baseline loses more than 50% accuracy. Due to the interactive nature of the LBS applications, an ideal solution should offer excellent accuracy, practical efficiency, and proven privacy guarantee.

To tackle the challenges of FPRC problem, we propose a grouping-based technical framework named FedGroup. To reduce the large amount of noise injected by directly applying DP for each data owner, we assign similar data owners (mobile users in our setting) into groups. We argue that injecting DP noise in each group (as opposed to injecting noise for each data owner) suffices to provide privacy guarantee. The reason is due to the fact that certain users may have social relationships with other data owners (*e.g.*, they belong to the same family), and they share similar trajectories and whereabouts.

To achieve effective grouping, we adopt a commonly accepted concept named (k, r)-core in graph analysis [23] to ensure that the group has strong connections inside (here the nodes in the graph are data owners and the edges between nodes indicate that they have strong spatial similarities, *i.e.*, their trajectories are similar). We formulate an optimization problem to find the best way to put users into groups in order to minimize the error introduced by DP noise in the FPRC problem. We show that the problem is NP-hard and provide an efficient greedy-based algorithm with a performance guarantee.

Last but not least, the grouping of users relies on how to measure the spatial similarity between users (how similar their trajectories are). It is a non-trivial task to construct the similarity graph between each pair of users, considering that the trajectories are private information of each data owner. We devise a novel hybrid solution to combine DP and MPC to achieve accurate similarity graph construction.

To summarize, we make the following contributions:

- We develop a novel technical framework FedGroup, to utilize an offline grouping step to largely reduce the amount of noise needed in the Federated Privacy-preserving Range Counting (FPRC) problem. We introduce the FPRC problem in Sec. 2.
- In order to achieve effective grouping in FedGroup, we show that it is an NP-hard problem to achieve optimal grouping to minimize the amount of noise needed in FPRC problem. Then, we propose an efficient greedy-based algorithm with a performance guarantee to achieve effective grouping.
- We also devise effective techniques to construct the spatial similarity graph between users by combining DP and MPC techniques, considering data owners' trajectories as private information. The details are presented in Sec. 3.
- We conduct extensive experiments to validate the effectiveness and efficiency of our proposed solution. The results are shown in Sec. 4.

In addition, we review related works in Sec. 5 and conclude in Sec. 6.

2 Problem Definition

In this section, we introduce some basic concepts, the adversary model, and the formal definition for the Federated Privacy-preserving Range Counting (FPRC) problem.

2.1 Basic Concepts

Definition 1 (Location). *A location $l = (x, y)$ represents a 2-dimensional spatial point with the coordinates (x, y) on an Euclidean space.*

Definition 2 (Data owner). *There are m data owners $u_1, \ldots, u_m$. Each data owner u_i owns a spatial database D_i. Each spatial database D_i consists of multiple data records (locations as in Definition 1)* , i.e., *$D_i = \{l_1, \ldots, l_{|D_i|}\}$.*

We also refer to each data owner's database D_i as a *data silo*. We let $D = \cup_{i=1}^{m} D_i$ denotes the collection of all data silos, *i.e.*, the union of all data records from each data silo.

Definition 3 (Range counting query). *A range counting query Q asks for how many data records are within distance r to the query location q_0,* i.e.,

$$Q(r, q_0) = \sum_{l \in D} \mathbb{I}(d(l, q_0) < r) \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function which equals to 1 if the predicate $d(l, q_0) < r$ is true, and 0 otherwise. $d(\cdot)$ is the Euclidean distance function. $l \in D$ is any record from the union of all data records from each data silo.

A toy example is provided in the full report [1] for easier understanding.

2.2 Privacy and Adversary Model

There are mainly three parties (roles) in our spatial data federation (see Fig. 1): (1) the **end user** who issues the query; (2) the **service provider** who receives the query and coordinates the execution of the query; and (3) each **data owner** u_i who owns the private spatial database D_i.

We assume that all parties are *semi-honest*, meaning that they are *curious but not malicious*. They are *curious* about other parties' private information, but they honestly follow and execute system protocols. The setting has been widely adopted in recent privacy-preserving LBS related applications [18,22,16].

2.3 Federated Privacy-Preserving Range Counting problem

Based on the previous concepts and the adversary model, we now define the Federated Privacy-preserving Range Counting (FPRC) problem as follows.

Definition 4 (Federated Privacy-preserving Range Counting problem). *Given a federation of m spatial databases $D_1, \ldots, D_m$, a range counting query $Q(q_0, r)$, and a privacy parameter (*a.k.a.*, privacy budget) ϵ, the FPRC problem asks for a query answer $\tilde{Q}$, with the following privacy requirements:*

- *R1. The computed final result $\tilde{Q}$ satisfies ϵ-Differential Privacy (DP), where the definition of neighboring databases refers to changing one single record in any data silo D_i.*
- *R2. The intermediate result disclosed by any data silo D_i satisfies ϵ-DP.*
- *R3. The private inputs for each data silo D_i are confidential if there is any multiparty computation involved.*

Two baselines, each respectively based on MPC and DP, are proposed. The details are deferred to the full report [1].

As mentioned in Sec. 1, the **key challenges** to solve the FPRC problem are threefold: 1) privacy: each data owner is reluctant to disclose its sensitive data; 2) efficiency: directly applying MPC results an impractical solution; 3) accuracy: an effective solution should offer reasonably high accuracy, showing significant improvements over the DP baseline.

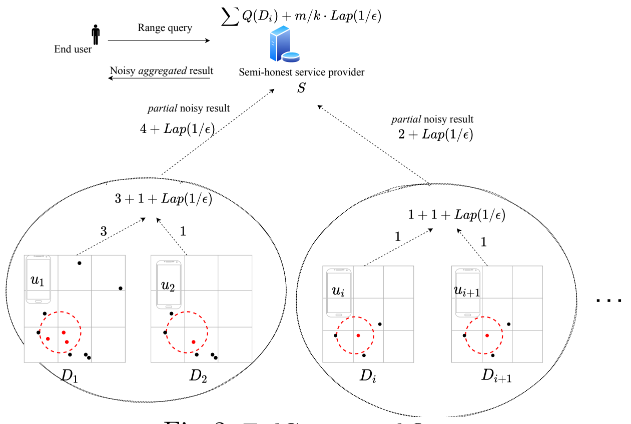


Fig. 2: FedGroup workflow.

3 Our Solution FedGroup

In this section, we introduce the technical details of FedGroup, which addresses the challenges of the FPRC problem. We highlights its overall workflow first (Fig. 2 is an illustrative figure), and then introduce the details in each of the steps: 1) constructing the spatial similarity graph; 2) finding groups given the similarity graph; and 3) partial answers and aggregation.

3.1 Key Idea

Intuition In our setting of the Federated Privacy-preserving Range Counting (FPRC) problem (as in Sec. 2), the number of data owners (m) is potentially very large (*e.g.*, millions), as each data owner could be a mobile user. The key idea of FedGroup is based on the following observation: certain mobile users have strong connections (due to social ties or other collaboration relationships) with each other. If we consider such pairs of data owners to belong to the same group, then the privacy protection inside the group could be relaxed. For example, there is no need to consider privacy protection of users' trajectories within a family, as family members are very likely to know about other members' whereabouts during the day. Thus, DP noise is only needed cross groups, as opposed to the case in the DP straw-man baseline, where an instance of Laplace noise is injected for every mobile user. As a result, the overall noise injected for the query result for the FPRC problem could be greatly reduced.

3.2 Spatial Similarity Graph Construction

In this step, the goal is to construct an undirected graph $G_s = (V, E)$ between data owners. Each node in V of the graph represents a data owner u_i, and an edge in E between two nodes u_i and u_j indicates the similarity of the spatial databases D_i and D_j owned by u_i and u_j, respectively. The weight of an edge measures the strength of the connection between u_i and u_j, and we use cosine similarity as the weight of the edge.

We first introduce the details of similarity graph, including how to measure the spatial similarity between data owners. Then, we consider the privacy-preserving setting, where each data owner's spatial database is considered private, and present our solutions of computing the similarity function between data owners in a differentially private manner.

Spatial similarity graph The key to construct the spatial similarity graph $G_s = (V, E)$ is measuring the weight of each undirected edge $e = (u_i, u_j)$ between two data owners $u_i, u_j \in V$. In the following section, we introduce how to compute the similarity function between the two data silos D_i and D_j.

For each data silo D_i, we use a grid structure T to decompose the spatial domain into a number of grids, where $|T|$ denotes the number of grids. Then, within each grid, a count is computed to denote the number of spatial points (records) of D_i falling inside the grid. Thus, we obtain a $|T|$-dimensional count vector for the data silo D_i.

Similarly, for data silo D_j, we could obtain another count vector $v_j = [c_{j,1}, c_{j,2}, \ldots, c_{j,|T|}]$. Since the two vectors v_i and v_j are with the same length $|T|$, we could use *cosine similarity* to measure their similarity as:

$$\text{sim}(D_i, D_j) = \cos(v_i, v_j) = v_i \cdot v_j / \|v_i\| \|v_j\| \tag{2}$$

An illustrative example is provided in the full report [1].

The cosines similarity ranges from 0 to 1, and the larger the value is, the more similar the two count vectors are. In our application, the higher the cosine similarity between the two count vectors is, the more similar are the two associated spatial databases, owned by two different data owners.

Constructing the graph To construct the spatial similarity graph $G_s = (V, E)$, first we insert all the data owners as the node set V. Then, we iterate over all pairs of data owners $u_i, u_j \in V$, and measure the spatial similarity of their spatial databases D_i and D_j, according to Equation (2). The weight of the corresponding edge $e = (u_i, u_j)$ is set as $\text{sim}(D_i, D_j)$, *i.e.*, $w(e) = \text{sim}(D_i, D_j)$.

An illustrative example is provided in the full report [1].

Without considering the privacy of each data silo D_i, constructing the spatial similarity graph G_s is straightforward. The method is running on the service provider S. First, we create the node set V of G_s by inserting all data owners $u_1, \ldots, u_m$ into V. Then, the method simply iterates over the pairs of data owners u_i, u_j for $i, j \in [m]$ and $i < j$, and calculates $\text{sim}(D_i, D_j)$. If the similarity is larger than the given threshold r, then an edge (u_i, u_j) will be inserted into the edge set E of G_s, with the edge weight set as $w(u_i, u_j) = \text{sim}(D_i, D_j)$.

The time complexity is $O(m^2)$, since we iterate over all possible pairs of data owners. The space complexity required is also $O(m^2)$, because we need to store G_s on S, including all the nodes and the edges (and their weights).

Privacy-preserving computation Although it is straightforward to construct G_s without considering the privacy of each data silo D_i, it is crucial to provide a proven privacy guarantee when we measure the spatial similarity between each pairs of data silos. The first and foremost motivation of this work is to consider privacy-preservation in the spatial federation setting, where each data owner's data silo D_i is considered sensitive.

Obviously, the non-private way of constructing the graph fails to meet the privacy requirement R2 in our problem definition in Definition 4. In R2, it requires that the intermediate results shared by each data owner satisfy ϵ-DP. However, in the non-private version, the provided D_i (or its count vectors v_j) are directly published by each data owner to the service provider S. Clearly it is a privacy failure.

Furthermore, R1 in Definition 4 may fail. It is not clear whether the constructed similarity graph G_s provides any formal privacy guarantee *w.r.t.* changing of one record in any data silo D_i.

Thus, in this section, we provide two options for privacy-preserving way of constructing G_s, and offer a hybrid solution. The solution utilizes both Secure Multiparty Computation (MPC) and the standard Laplace mechanism in DP.

Hybrid solution. The hybrid solution combines the advantages of both worlds – the high accuracy of the MPC option (because the computations are *exact*), and the high efficiency of the DP option (because each data silo performs the Laplace computation independently, the injecting Laplace noise itself is an efficient operation).

The key idea of the hybrid solution is that: we use the efficient Laplace mechanism in DP to quickly compute a noisy weight $\tilde{w}$. Instead of directly using this noisy weight as a surrogate weight for the edges on the ground-truth graph, we only use it as a filtering mechanism. If the noisy weight $\tilde{w}$ is too small as compared to the given threshold r, we could quick conclude that the edge should *not* be inserted to the graph. Or, if the noisy weight $\tilde{w}$ appears to be high, we quickly conclude that the edge should be inserted to the graph. If the noisy weight lies on the borderline, which shows uncertainty and may introduce errors, we invoke the computationally heavy MPC method to calculate the exact edge weight given the two private inputs from two data owners.

Algo. 1 shows the detailed steps for the hybrid solution. The inputs $\tilde{v_1}, \ldots, \tilde{v_m}$ are the noisy counts vectors, obtained by injecting Laplace noise to the counts vectors $v_1, \ldots, v_m$.

Privacy analysis. We show that our hybrid solution satisfies the privacy requirements as in Definition 4.

Theorem 1. *The hybrid solution in Algo. 1 satisfies all privacy requirements (R1-R3) in Definition 4.*

Proof. Since the only released intermediate results are the noisy counts $\tilde{v_i}$, R2 is satisfied. R1 is satisfied by the post-processing and parallel composition theorem of DP, because the output graph $\tilde{G_s}$ only depends on the noisy counts. R3 is satisfied because the multi-party computation only happens at Line 12, and

Algorithm 1: `Construct` G_s `- Hybrid solution`

```
Input: $u_1, \dots, u_m, \tilde{v_1}, \dots, \tilde{v_m}, r, r_l, r_u$.
Output: $\tilde{G_s} = (V, \tilde{E})$.
1  $V := \{u_1, \dots, u_m\}$
2  $\tilde{E} := \{\}$
3  foreach $u_i \in V$ do
4      foreach $u_j \in V$ and $j > i$ do
5          $e := (u_i, u_j)$
6          $\tilde{w} := \cos(\tilde{v_i}, \tilde{v_j})$
7          if $e.weight < r_l$ then
8              Continue
9          else if $\tilde{w} > r_u$ then
10             $e$.weight $:= \tilde{w}$
11             Insert $e$ to $\tilde{E}$
12         else
13             $e$.weight $:=$ Secure_Sim$(D_i, D_j)$
14             if $e.weight > r$ then
15                 Insert $e$ to $\tilde{E}$
16 return $\tilde{G_s} = (V, \tilde{E})$
```

the confidentiality of the data is provided by the MPC protocol. We defer the detailed analysis to the full report [1].

3.3 Finding Groups

(k, r)-core We adopt a commonly accepted concept (k, r)-core [23] from graph mining literature to define the groups. The concept of (k, r)-core considers two important criteria of determining closely related groups (*a.k.a.*, community detection): *engagement* and *similarity*, such that the group members not only have strong social connections with each other (strong engagement), but also share high similarities (*i.e.*, their trajectories are similar).

In our setting, as we focus on the spatial database of data owners, rather than the social graph, we require that the groups we form demonstrate strong similarities between data owners. Thus, we require each group to be a r-clique, meaning each group is a clique, and the edge weight (*i.e.*, the spatial similarity, introduced in Sec. 3.2) between the data owners is larger or equal to r.

Definition 5 (r-group). *A r-group g is a clique in the spatial similarity graph G_s, such that there exists an edge between any two data owners belonging to the group, and the edge weight is larger than the threshold r,* i.e.*, for any two data owners $u_i, u_j \in g$,*

$$\exists e = (u_i, u_j) \in E \ , w(e) > r \tag{3}$$

Optimizing the grouping The concept of r-group gives meaning to grouping data owners together, because they are similar. However, there could be various ways of putting data owners into different groups. For example, a data owner could belong to multiple r-group, and which group should we choose to put the data owner in?

In this section, we formulate an optimization problem to connect the utility goal of the ultimate problem of this paper – the FPRC problem, with the grouping strategy. Then, we show that this optimization is an NP-hard problem, and we propose a greedy algorithm, which is effective and efficient.

We first define the Data Owner Grouping (DOG) problem.

Definition 6 (Data Owner Grouping (DOG) problem). *Given the inputs to the FPRC problem in Definition 4, the DOG problem asks for a way to assign data owners into λ disjoint groups: $g_1, g_2, \ldots, g_\lambda$, where each group g_i is a r-group (according to Definition 5), such that the aggregated amount of noise injected by FedGroup for the FPRC problem is minimized.*

Next, we show that to minimize the error (the total amount of noise injected) for the FPRC problem, it is equivalent to minimize the number of groups λ.

Lemma 1. *Minimizing the error in the FPRC problem is equivalent to minimizing λ, which is the total number of groups in the DOG problem.*

Proof. We review the error of our FedGroup solution in the FPRC problem. FedGroup assigns data owners into disjoint groups, and each group injects one single instance of Laplace noise to the query result. Then, the partial noisy result is collected from each group and aggregated as the final answer. Thus, the total error of our final answer $\tilde{Q}$, which is measured by its variance, is:

$$\mathrm{Var}(\tilde{Q}) = \mathrm{Var}(\sum_{i=1}^{\lambda} \mathrm{Lap}(1/\epsilon)) = \lambda \cdot \mathrm{Var}(\mathrm{Lap}(1/\epsilon)) = 2\lambda/\epsilon^2. \quad (4)$$

Thus, to minimize the total error in Equation (4), we need to minimize λ.

Now, our DOG problem becomes finding an optimal way of assigning data owners into disjoint groups, where each group is a r-clique, and the number of groups is minimized. In fact, the problem could be reduced from the Minimum Clique Partition (MCP) problem and the hardness result is shown as follows.

Theorem 2. *The DOG problem is NP-hard.*

Proof. We reduce the Minimum Clique Partition (MCP) problem to the DOG problem. Since MCP problem is an NP-hard problem, DOG shares the same hardness. Due to space limit, the detailed proof is deferred to the full report [1].

Greedy algorithm. Since the DOG problem is an NP-hard problem, we devise an efficient greedy solution to obtain effective grouping to provide a bounded noise scale in the returned solution. The solution is inspired by the greedy solution to the Minimum Graph Coloring problem, which is shown as an equivalent problem to the MCP problem.

Algorithm 2: `Greedy Find Groups`

```
   Input: $\tilde{G_s} = (V, \tilde{E})$.
   Output: $g_1, g_2, \ldots, g_\lambda$.
1  $G_c$ := ComplementGraph($\tilde{G_s}$)
2  assigned := InitializeIntegerArray(size=$|V|$, initValue=0)
3  $\lambda := 1$ ; assigned[$u_1$] = 1 ; $g_1$.insert($u_1$)
4  $u$ := NextUnassignedVertex($V$, assigned)
5  while $u$ exists do
6      $gid$ := NextGroupId($u$.neighbors(), assigned)
7      assigned[$u$] = $gid$ ; $g_{gid}$.insert($u$)
8      if $gid > \lambda$ then
9          $\lambda = gid$
10     $u$ := NextUnassignedVertex($V$, assigned)
11 return $g_1, g_2, \ldots, g_\lambda$
```

Due to space limit, the correctness proof, together with an illustrative running example, are deferred to the full report [1].

Performance guarnatee. The greedy solution we present has a bounded number of groups, and it offers a bounded noise for the FPRC problem, as the next theorem shows.

Theorem 3. *If d is the largest degree of a node in the complement graph G_c, then Algo. 2 returns at most $d+1$ groups.*

Proof. We focus on a data owner u with degree d (the maximum degree in the graph) as Algo. 2 proceeds. There are at most d neighbors of u that we should avoid using the same group id. Since we are using the lowest-numbered group ids that have not been used by any of the neighbors, among group id $1, 2, \ldots, d+1$, there is at least one id that could be used by u (*e.g.*, the first d group ids are used, and we now use $d+1$). Thus, we conclude that at most $d+1$ groups are returned by Algo. 2.

Time complexity. Each time when Algo. 2 processes a vertex, it iterates over the neighbors of the vertex to find the lowest-numbered group id that is available. Thus, overall, it takes $O(|E|)$ time to run the algorithm. The graph complement step at the initialization also takes $O(|E|)$ time.

3.4 Partial Answers and Aggregation

Sec. 3.3 describes the most critical step of FedGroup, which is finding the groups for data owners. After the grouping, data owners belonging to the same group could avoid injecting separate instances of Laplace to their individual query answers. Instead, each group aggregates the partial answers from data owners and injects one instance of Laplace noise to the partial aggregated answer.

As the last step, the service provider S collects the partial noisy answers from all groups $g_1, \ldots, g_\lambda$. The noisy answers are aggregated (taking a summation) and then returned to the end user.

3.5 Extension to Range Aggregation Queries

In this paper, we focus on the most fundamental primitives for LBS systems, the range counting queries. However, the techniques we present could be easily extended to other aggregation queries, including range `SUM()` and `AVG()`.

To extend FedGroup to `SUM()`, the important extension is the scale of noise injected for each group (the sensitivity). Since now each data record in a spatial database does not only affect the final query result by +/-1, we need to inject the worst-case scale to satisfy differential privacy. To avoid injecting unbounded Laplace noise, a truncation could be performed, *i.e.*, we truncate all data records to make a certain attribute smaller or equal to a truncation parameter θ.

The extension to `AVG()` is straightforward as the average could be calculated by `SUM()/COUNT()`.

4 Experimental Study

In this section, we first introduce the experimental setup including the datasets, the control variables, baselines, and the metrics in Sec. 4.1. Then, we present the detailed experimental results in Sec. 4.2.

4.1 Experimental Setup

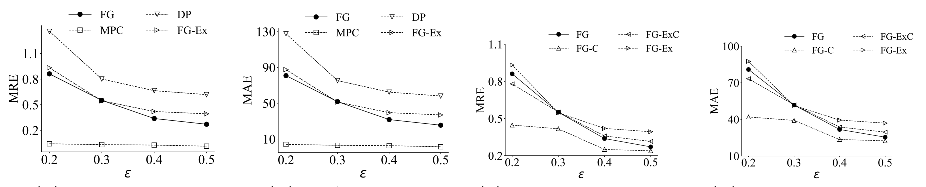
Datasets. We use a real-world geo-social network datasets Gowalla [7] and a randomly generated synthetic dataset in our experiments.
Baselines. We compare our proposed FedGroup (short as **FG**) method with the straw-man DP baseline (short as **DP**) and the straw-man MPC baseline (short as **MPC**).

For FedGroup, we also implement a variant based on cliques sizes (a heuristic based solution). It lists the cliques in the graph in descending sizes, and keeps adding the largest clique into a new group until all nodes are assigned with a group. We name this variant as FedGroup-Exhaustive (short as **FG-Ex**). Note that finding the maximum clique in a graph is in general an NP-hard problem, so we expect that this variant is only tractable on small graphs.
Metrics. We focus on the following end-to-end metrics:

- Mean relative error (MRE): the ratio of error as compared to the true result, *i.e.*, |returned result - true result|/true result, averaged over repeated queries.
- Mean absolute error (MAE): the absolute error of the returned result as compared to the true results, *i.e.*, |returned result - true result|, averaged over repeated queries.
- Query evaluation time: the time to execute the spatial count queries in seconds, averaged over repeated queries.

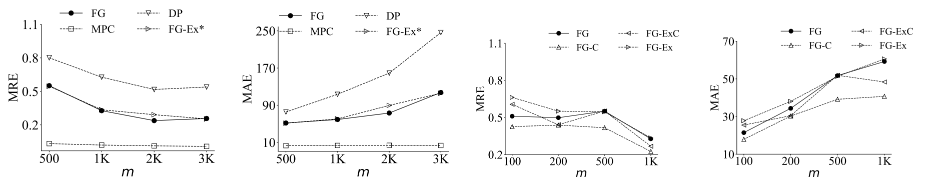
Control variables.

- The number of data owners: $m \in [\mathbf{500}, 1K, 2K, 3K]$. For the scalability test: $m \in [100K, 250K, 500K, 1M]$.



(a) MRE vs. ϵ. (b) MAE vs. ϵ. (c) Noisy vs. Clear. (d) Noisy vs. Clear.

Fig. 3: End-to-end query accuracy of different methods vs. the privacy budget ϵ. (Gowalla dataset, $m = 500$.)



(a) MRE vs. m. (b) MAE vs. m. (c) Noisy vs. Clear. (d) Noisy vs. Clear.

Fig. 4: End-to-end query accuracy of different methods vs. the number of federations m. (Gowalla dataset, $\epsilon = 0.3$.)

- Privacy budget: $\epsilon \in [0.2, \mathbf{0.3}, 0.4, 0.5]$. The privacy budget considered is relatively small, because it is a personal budget for each data owner, and repeated queries may result linear composition of the budget.

4.2 Experimental Results

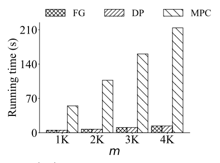
First, we present the results on query accuracy to first verify the motivation of our work – our proposed solution FedGroup offers superior query accuracy as compared to the straw-man DP baseline.

Then, we compare the running time of query execution to show that the straw-man MPC baseline is not tractable even in moderate data sizes, while our solution FedGroup offers excellent scalability.

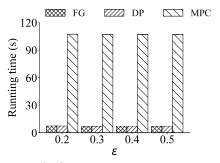
Despite the best query accuracy offered by the MPC solution (due to the single instance of Laplace noise injected), it suffers from prohibitive computational overhead. Our FedGroup solution offers the best utility/efficiency tradeoff. FedGroup provides up to 50% query accuracy gain over the other DP baseline while maintaining the same efficiency, which allows it to scale to datasets with millions of data owners.

Query accuracy. We test the query accuracy (using MRE and MAE) of different methods. The results are shown in Fig. 3 and Fig. 4. Over different privacy budget, FedGroup consistently outperforms the DP baseline by a large margin (30-50% improvement). Across different input sizes (the number of federations m), FedGroup also provides a significant accuracy improvement.

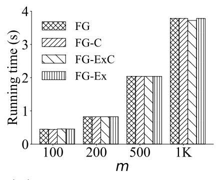
Varying ϵ. Fig. 3 shows the query accuracy results when we test the compared methods over different privacy budget ϵ for a spatial federation of $m = 500$ data



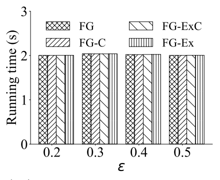
(a) Varying m.



(b) Varying ϵ.

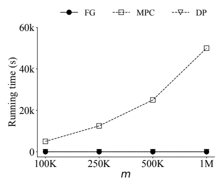


(c) Noisy vs. Clear.

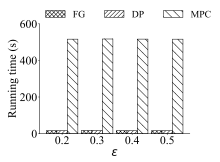


(d) Noisy vs. Clear.

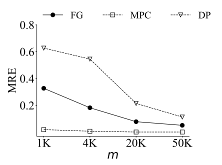
Fig. 5: Efficiency (running time) for different control variables. (Gowalla dataset)



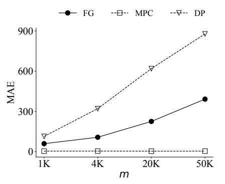
(a) Time vs. m.



(b) Time vs. ϵ.



(c) MRE vs. m.



(d) MAE vs. m.

Fig. 6: Scalability tests. (Synthetic dataset)

owners using the Gowalla dataset. FG offers a clear improvement over the strawman DP baseline. Specifically, as shown in Fig. 3a, the MRE for the DP baseline at $\epsilon = 0.3$ is 0.801 (80.1%), while FG reduces the error to 0.551 (55.1%), which provides a $(0.801 - 0.551)/0.801 * 100\% = 31.2\%$ improvement.

When ϵ gets larger, indicating a more relaxed privacy requirement, the query accuracy generally improves for all methods, reflected by a smaller MRE and MAE. At $\epsilon = 0.5$ (as shown in Fig. 3a and Fig. 3b), the MRE and MAE for the DP baseline is 0.618 and 58.069, respectively. In comparison, the MRE and MAE of FG improves to 0.271 and 25.448, respectively. The query accuracy improvement of FG over DP is about 56.1%. The accuracy improvement is more significant than the one of $\epsilon = 0.3$ (31.2%), and the explanation is that when ϵ gets larger, more accurate spatial similarity graphs are constructed (leading to more correctly identified edges), and FG could assign data owners into smaller number of groups, which eventually leads to a smaller end-to-end query error.

It is worth noting that the computationally heavy MPC baseline offers the best query accuracy (due to the $O(1)$ noise) across all ϵ. Despite its impracticability in real-world data sizes, we include its results here for completeness.

Varying m. For a fixed privacy budget ϵ, we test the query accuracy of different methods across different numbers of federations m. The results (Fig. 4) verify that FG consistently outperforms the DP baseline by a significant margin. As m grows from 500 to 3K, the MAE of the DP baseline grows from 75.3 to 246.5. In comparison, our FG provides a lower MAE, which grows from 51.8 to 117.4. From Fig. 4b, we could see that the MAE of FG grows more slowly than the DP baseline.

When m increases, the true query count also increases. This results a decreasing trend for MRE for all methods, as shown in Fig. 4a. When $m = 2K$, the MRE of the DP baseline is 0.517, while our FG's MRE is 0.238, which provides a 53.97% improvement. For different m, the MRE of DP is about $2\times$ as large (worse) as the one of our FG.

Query efficiency. We then conduct experiments to test the query efficiency for different methods. The results are shown in Fig. 5. The results on the scalability tests on the synthetic dataset are shown in Fig. 6.

Gowalla dataset. For smaller m in the range of 1K-4K, we show the query execution time of different methods in Fig. 5, using the Gowalla dataset. The query time using the MPC method grows linearly as m grows (Fig. 5a), and stays at the same level across different ϵ (Fig. 5b), but it is orders-of-magnitude larger (slower) than FG and DP.

Fig. 5c and Fig. 5d show that the query execution time is not impacted much by whether we use the clear (ground-truth) graph or the noisy graph. The time grows roughly linearly when m increases, but all FG variants finish within 4 seconds when $m = 1K$, which is not comparable to the computationally heavy MPC baseline.

Scalability tests. We further verify the scalability of our FG method by testing on the synthetically generated dataset. Fig. 6a shows the query execution time when we scale m to a million. Despite the linear growth of running time of MPC, it requires more than 13 hours to execute a query when $m = 1M$. This is not acceptable in real-world LBS applications. In comparison, both the DP and our FG answers the query within 20 seconds at the largest input size.

For a given m, the running time of different methods do not vary much over different ϵ (Fig. 6b). In addition to the efficiency results, we also include some query accuracy results when we test the methods on extremely large inputs from the synthetic dataset (shown in Fig. 6c-Fig. 6d). These results further verify that our FG method is consistently outperforming the equally efficient DP method.

Summary. We conduct extensive experiments to examine both the effectiveness (in terms of query accuracy) and the efficiency of our proposed method FedGroup. In terms of query accuracy, FedGroup provides up to 50% improvement over the DP baseline. Though the computationally heavy MPC baseline delivers the best accuracy, its inefficiency prevents it from real-world application. The FedGroup runs orders-of-magnitude faster than the MPC baseline and executes a query within 20 secondes for a million data owners.

5 Related Work

In this section, we first review the works on general-purposed data federations, and then discuss specific works on spatial data federation, as well as other spatial data management techniques considering data privacy.

General-purposed data federations. The concept of data federation dated back to the 90s when the need arises to manage autonomous but cooperating database systems [14]. In recent years, there are a growing number of works on optimizing and adopting secure multiparty computations to build privacy-preserving data federations [3,4,5]. Bater et al. first implemented the MPC-based data federation system called SMCQL [3], and then improved the system to Shrinkwrap [4] by using differential privacy to reduce the amount of dummy records inserted during query execution in order to make the computation *oblivious*. Its later effort, SAQE [5], further improves the utility/efficiency/privacy tradeoff by considering approximate query processing, as a by-product of injecting necessary noise for achieving differential privacy. Conclave [20] is another system which adopts secure query processing, *i.e.*, the private data are *confidential* during the execution of the queries.

These security-based systems offer strong privacy guarantee, however the efficiency issue is a bottle-neck. Most of the works are limited to only two parties in the secure computation, and experimental results [19] show that the current system is still far from being scalable in real-world data sizes. For example, joining a private table L (shared by multiple parties) and a public table R takes more than 25 minutes, even if R contains only 100 objects.

Specialized spatial data federations. In an effort to develop specialized and more optimized spatial data federation, Shi et al. [15] first studied how to efficiently perform approximate range aggregation queries without considering privacy. Later, a MPC-based system HuFu [19] was built, with specialized optimization made for spatial operators. Though significant improvements are made over other existing works (*e.g.*, Conclave), the running time is still a major concern. In our experiments, we demonstrate even the most simplified MPC baseline runs orders-of-magnitude slower than our proposed solution.

Privacy-preserving spatial data management. In parallel, there are a fruitful amount works of spatial data management using DP or local DP (LDP). The DP model masks the existence of a single location record [9,24] or an entire trajectory [12]. There are also other efforts on protecting location data [21] or using LDP or its variants on offering theoretical guarantee to protect the location data [6,2,11,13,8,10]. The DP and LDP model both differ from our privacy model, where a number of data owners exist.

6 Conclusion

In this paper, we target at the Federated Privacy-preserving Range Counting (FPRC) problem, where the number of federations (data owners) is large. By utilizing the social ties between data owners (such as family members), we propose a grouping-based framework called FedGroup. It reduces the amount of noise required by only injecting one instance of Laplace noise per group, while the straw-man DP baseline requires one instance of noise for each data owner, resulting a unacceptably large amount of noise, overweighing the true query answer. In addition, FedGroup is vastly more efficient than the MPC-based solution, especially when we scale to a million data owners. Extensive experimental results verify the accuracy improvement and efficiency advantage of our proposed solution.

References

1. Anonymous full paper. https://anonymous.4open.science/r/dpfed_code-852B/full.pdf (2022)
2. Andres, M.E., Bordenabe, N.E., Cjhatzikokolakis, K., Palamidessi, C.: Geo-indistinguishability: differential privacy for location-based systems. In: CCS (2013)
3. Bater, J., Elliott, G., Eggen, C., Goel, S., Kho, A.N., Rogers, J.: SMCQL: Secure query processing for private data networks. VLDB (2017)
4. Bater, J., He, X., Ehrich, W., Machanavajjhala, A., Rogers, J.: Shrinkwrap: Efficient SQL query processing in differentially private data federations. VLDB (2018)
5. Bater, J., Park, Y., He, X., Wang, X., Rogers, J.: SAQE: practical privacy-preserving approximate query processing for data federations. VLDB (2020)
6. Chen, R., Li, H., Qin, A.K., Kasiviswanathan, S.P., Jin, H.: Private spatial data aggregation in the local setting. ICDE (2016)
7. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. KDD (2011)
8. Cormode, G., Kulkarni, T., Srivastava, D.: Answering range queries under local differential privacy. VLDB (2019)
9. Cormode, G., Procopiuc, C., Srivastava, D., Shen, E., Yu, T.: Differentially private spatial decompositions. ICDE (2012)
10. Cunningham, T., Cormode, G., Ferhatosmanoglu, H., Srivastava, D.: Real-world trajectory sharing with local differential privacy. VLDB (2021)
11. Gu, X., Li, M., Cao, Y., Xiong, L.: Supporting both range queries and frequency estimation with local differential privacy. In: CNS (2019)
12. He, X., Cormode, G., Machanavajjhala, A., Procopiuc, C.M., Srivastava, D.: DPT: differentially private trajectory synthesis using hierarchical reference systems. VLDB (2015)
13. Hong, D., Jung, W., Shim, K.: Collecting geospatial data with local differential privacy for personalized services. In: ICDE (2021)
14. Sheth, A.P., Larson, J.A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. CSUR (1990)
15. Shi, Y., Tong, Y., Zeng, Y., Zhou, Z., Ding, B., Chen, L.: Efficient approximate range aggregation over large-scale spatial data federation. TKDE (2021)
16. Tao, Q., Tong, Y., Zhou, Z., Shi, Y., Chen, L., Xu, K.: Differentially private online task assignment in spatial crowdsourcing: A tree-based approach. In: ICDE (2020)
17. Technode: Alibaba's amap launches taxi ride-hailing platform in beijing (2021)
18. To, H., Shahabi, C., Xiong, L.: Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. In: ICDE (2018)
19. Tong, Y., Pan, X., Zeng, Y., Shi, Y., Xue, C., Zhou, Z., Zhang, X., Chen, L., Xu, Y., Xu, K., et al.: Hu-fu: efficient and secure spatial queries over data federation. VLDB (2022)
20. Volgushev, N., Schwarzkopf, M., Getchell, B., Varia, M., Lapets, A., Bestavros, A.: Conclave: secure multi-party computation on big data. In: EuroSys (2019)
21. Xiao, Y., Xiong, L.: Protecting locations with differential privacy under temporal correlations. In: CCS (2015)
22. Zhai, D., Sun, Y., Liu, A., Li, Z., Liu, G., Zhao, L., Zheng, K.: Towards secure and truthful task assignment in spatial crowdsourcing. WWW (2019)
23. Zhang, F., Zhang, Y., Qin, L., Zhang, W., Lin, X.: When engagement meets similarity: Efficient (k, r)-core computation on social networks. VLDB (2017)
24. Zhang, J., Xiao, X., Xie, X.: Privtree: A differentially private algorithm for hierarchical decompositions. In: SIGMOD (2016)