IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 10, 2020, accepted April 22, 2020, date of publication April 27, 2020, date of current version May 18, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2990706*

# Maximum Influential Location Selection With Differentially Private User Locations

**SEHWA PARK, JUNKYU LEE, AND SEOG PARK, (Member, IEEE)**
Database Laboratory, Department of Computer Science and Engineering, Sogang University, Seoul 04107, South Korea

Corresponding author: Seog Park (spark@sogang.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) under Grant NRF-2017R1D1A1B03036252 and Grant NRF-2019R1A2C1088126

**ABSTRACT** The widespread use of mobile devices and social network services has made optimal location queries an important research topic. Previous studies have focused on the problem of maximum influential (Max-inf) location selection, that is, finding a location that can attract as many clients as possible. The location information of each client should be collected to process such a query. However, client location is considered sensitive information. Therefore, a privacy protection technique should be applied to Max-inf problems. Motivated by this, we propose a Max-inf problem query-processing technique with differentially private client location information. Furthermore, we present a Voronoi region-based technique to guarantee query accuracy and a Voronoi envelope-based pruning heuristic to improve query performance.

**INDEX TERMS** Differential privacy, maximum influential location selection problem, optimal location selection query.

## I. INTRODUCTION

During the past decades, a vast amount of geo-spatial data has been collected by various location-based services owing to the widespread use of mobile devices. The increasing amount of location data can provide exciting opportunities to support market analysis, such as decision making problems of competitive location selection and establishment of public facilities. Especially, previous studies of geo-spatial data have focused on the maximum influential (Max-inf) location selection problem, that is, finding a location that can attract as many clients as possible. These applications generally assume that customers trust data analysts and agree to the collection of their location information without any restrictions. However, location data are typically collected by telecommunication operators and social network service providers rather than data analysts. In addition, places that people visit disclose extremely sensitive information, such as their behavior, home and work locations, preferences, and habits. Therefore, people dislike disclosing their exact location, and location privacy has become an emerging issue in the spatial database community. For this reason, location-based service providers usually exploit privacy preserving data analyzing techniques when offering customer location data to data analysts. Motivated by this, we propose a novel privacy preserving query processing technique that finds the best location to establish a new facility while satisfying privacy requirements.

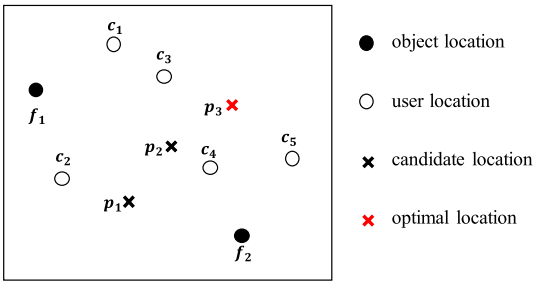



**FIGURE 1.** Example of max-inf problem.

Optimal location selection [1]–[7] is a common problem that finds the best location to add a new facility that optimizes an objective function. Specifically, a Max-inf problem [1]–[3] is a traditional problem that identifies the most influential object in a given database which consists of potential objects $P$, existing facilities $F$, and client locations $C$, as depicted in Fig. 1. A Max-inf problem finds a location that maximizes influence on clients under the assumption that each client utilizes the nearest facility. We occasionally use the words "user," "client," and "customer" interchangeably.

The associate editor coordinating the review of this manuscript and approving it for publication was Jerry Chun-Wei Lin.

A Max-inf problem is extremely useful for market analysis and beneficial in several real-life applications. For example, Starbucks may want to open a new branch to compete with other coffee shop franchises, or a telecommunication service provider may want to add a new base station in a densely populated area to improve their service quality. Fig. 1 shows two existing facilities, $\{f_1, f_2\}$, and five clients, $\{c_1, c_2, c_3, c_4, c_5\}$. We want to determine the best location between three different potential locations, $\{p_1, p_2, p_3\}$, to establish a new facility. We assume that $f_1$ is the nearest facility of $\{c_1, c_2, c_3\}$, and $f_2$ is the nearest facility of $\{c_4, c_5\}$. If a new facility is established at $p_1$, then it becomes the nearest facility of $\{c_2\}$. If we select location $p_2$, then we can attract clients $\{c_3, c_4\}$. However, if we select location $p_3$, then it becomes the nearest facility for every clients, that is, $\{c_3, c_4, c_5\}$. Therefore, $p_3$ is the most attractive location. A conventional approach to support such decision making is to utilize computational geometry techniques with the assumption that the exact locations of customers are known. However, customers provide information only to trusted service providers rather than data analysts.

To remedy this problem, we present novel Max-inf problem query-processing techniques while applying differential privacy to client location data. Differential privacy [8] is a de facto standard privacy protection technique that applies a randomized mechanism to add controlled noise into statistical query results. A naïve approach to apply differential privacy to a Max-inf problem is to add a Laplace noise to its objective function, which is called a Laplace mechanism. We invoke reverse nearest neighbor (RNN) queries for each potential location $p_i \in P$ and identify clients whose nearest neighbor is $p_i$. Then, we add a Laplace noise to the number of clients and select the most influential location. However, if the potential locations are close to each other, then client location information is disclosed recursively. For instance, the RNN clients of $p_2$ and $p_3$ are $\{c_3, c_4\}$ and $\{c_3, c_4, c_5\}$, respectively, as shown in the previous example. The intersection of the two RNN clients is $\{c_3, c_4\}$, which indicates that their location is leaked twice by $p_2$ and $p_3$. This problem is called sequential composition [8] in differential privacy and requires dividing the privacy budget $\epsilon$ by the number of potential locations. However, sequential composition degrades query accuracy exponentially and suffers from the number of potential locations. Thus, we propose a Voronoi region-partitioning method (VPM) that partitions the Voronoi region of a potential location to exploit the parallel composition [9]. Although the VPM mitigates the degradation of performance accuracy, it suffers from expensive computational cost. To reduce computational cost, we exploit 2 r-trees for potential locations and client locations with aggregate r-tree for existing facilities, which is a variation of the r-tree [19]. We present a pruning technique called the Voronoi envelope filtering method (VEM), which precomputes the upper-bound of the noisy count of influece regions to reduce search space.

In summary, our contribution is threefold. First, we present a Max-inf problem with a differential private user location. To the best of our knowledge, this study is the first attempt to apply differential privacy directly to the query processing of optimal location selection. Second, we present two algorithms to improve query performance, namely, the VPM and the VEM. Third, we study the properties of the proposed methods empirically on an actual dataset. The remainder of the paper is organized as follows. Section 2 reviews studies relevant to the Max-inf problem and differential privacy, and Section 3 formalizes problem definitions and presents system models. Sections 4 and 5 propose two query processing techniques, the VPM and the VEM. Section 6 provides the experimental results from an actual dataset. Finally, Section 7 concludes the paper and recommends directions for future work.

## II. RELATED WORKS AND BACKGROUNDS

In this section, we review existing studies relevant to the Max-inf problem and differential privacy. Location optimization problems are characterized by optimization functions and can be classified into three categories, namely, Max-inf, Min-sum, and Min-dist problems. These problems are closely related to RNN queries. Therefore, we first briefly review RNN studies.

### A. RNN QUERIES

The RNN has received significant attention in research [10]–[13] since its introduction by Korn and Muthukrishnan [10]. These authors were the first to study RNN queries and present a general approach to solve such queries. The authors precalculated the nearest neighbor distance for each data object and found its surrounding circle that its radius is the nearest neighbor distance. Then, for any query $q$, each point is the RNN for $q$ that contains $q$ in its circle. Yiu *et al.* [11] first studied the problem of RNN queries on road networks. They proposed the Eager algorithm, which is a filter and refinement method based on the network expansion approach. The Eager algorithm traverses the network around the query point $q$ with ascending order of the shortest distance from $q$ to each node of the network. For each node $n$ retrieved, the Eager algorithm performs a range-NN query in the range $d(n, q)$. If the data object $p$ is retrieved, then all the nodes with the shortest path to $q$ that pass through $n$ can be pruned. Vlachou *et al.* [12] extended the RNN to the reverse top-k query (RkNN), which retrieves an object in a weighted feature space. An RkNN query is used to assess the impact of a potential product in the market. This option is based on the number of clients that identify a top-k product according to their preference. The authors introduced a threshold algorithm-based method, that is, the RTA, to solve the RkNN problem. Lu *et al.* [13] investigated the reverse spatial and textual kNN (RSTkNN) search, which considers textual similarities in RkNN retrieval. An RSTkNN query is used to find objects that take a specified query object as one of its k-most spatial–textual similar objects. The authors proposed a hybrid index structure, namely, the intersection union r-tree (IUR-tree) to answer the RSTkNN query. The IUR-tree consists of an r-tree with inverted files for each node. The leaf

nodes contain entries with location and keyword information, whereas each intermediate node has union and intersection vectors for the keywords of its child. The authors designed a branch-and-bound algorithm on the basis of the IUR-tree to solve the RSTkNN problem.

### B. MAX-INF PROBLEMS

The Max-inf problem was first introduced by Cabello *et al.* [1]. It maximizes the influence of a facility, where influence indicates the number of clients who are the RNNs of a facility. The authors found regions for a new facility through the nearest location circle (NLC). The NLC of a client $c$ is a circle centered at $c$, and its radius is the distance between $c$ and the nearest facility of $c$. Only a facility established within the NLC of $c$ can be the new nearest facility of $c$. Therefore, a solution can be obtained by finding regions that are enclosed by the largest number of NLCs. Wong *et al.* [2] also studied the Max-inf problem using a first polynomial time complexity algorithm called the MaxOverlap. The authors likewise exploited NLCs to avoid evaluating intersection points that are guaranteed not to be optimal. Yan *et al.* [3] presented an approximate method for the Max-inf problem. The authors designed an efficient influential location miner called FILM, which returns a small grid cell where all locations have an influence guarantee. Contrary to existing approaches that return a precisely optimal location at the expense of long running time, the authors' approach returns near optimal locations in considerably less time. Meanwhile, Zhang *et al.* proposed the Min-dist location selection problem [4]. This method finds points within $Q$ given a client set $C$, an existing facility set $F$, and a region $Q$. Thus, if a new facility is established at any one of these points, then the average distance of the clients to their respective nearest facilities is minimized. To solve the problem, the authors proposed a method that initially identifies a set $L$ of candidate locations from $Q$ and then divides $L$ progressively until an answer set is found. Qi *et al.* [5] also resolved the Min-dist problem and proposed the maximum NFC distance (MND) method. The MND is a variation of the minimum bounding rectangle (MBR), which is combined with the NLC. Moreover, Xiao *et al.* [6] and Chen *et al.* [7] presented an optimal location selection query in a road network environment.

### C. DIFFERENTIAL PRIVACY

Differential privacy was first introduced by Dwork and Roth [8]. The aim of differential privacy is to mask the differences in queries among neighboring datasets. Its definition and properties are as follows.

*Definition 1: Differential privacy. Given two neighboring databases $D_1$,$D_2$, such that $\|D_1 - D_2\|_1 \leq 1$, a randomized mechanism $M$ is $\epsilon$-differential private if the following condition holds for all $S \subseteq Range(M)$.*

$$Pr[M(D_1) \in S] \leq exp(\epsilon)Pr[M(D_2) \in S] \quad (1)$$

where $\epsilon$ is the privacy budget, and $\|\cdot\|_1$ is a norm of a vector. One method to achieve $\epsilon$-differential privacy is to use a Laplace mechanism, as explained in the previous section. It simply adds noise sampled from a Laplace distribution to the query results, where the noise is proportional to the sensitivity of mechanism $M$.

*Definition 2: Sensitivity. For two neighboring data sets $D_1$,$D_2$, the sensitivity $\Delta$ of $M$ captures the magnitude by which a single individual's data can change the output of $M$ in the worst case, as follows*:

$$\Delta M = max_{D_1,D_2}\|M(D_1) - M(D_2)\| \quad (2)$$

Differential privacy satisfies simple composition properties, which are called sequential composition and parallel composition as follows.

*Definition 3: Sequential Composition. Let $M_1$ and $M_2$ be two differential private mechanisms and their privacy budgets be $\epsilon_1$ and $\epsilon_2$, respectively. Then, their combination, $M_{1,2}$, is $(\epsilon_1 + \epsilon_2)$-differentially private. Therefore, the composition of multiple differentially private mechanisms leads to a linear increase in the privacy budget or an increase in noise to maintain a fixed $\epsilon$ total privacy budget.*

*Definition 4: Parallel Composition. Let $M_i$ provide $\epsilon_i$–differential privacy and $D_i$ be an arbitrary disjoint subset of database $D$. Then, the sequence of $M_i(X \cap D_i)$ provides $max_i(\epsilon_i)$-differential privacy.*

Existing studies on differential privacy that is related to our work have used private spatial decomposition techniques. Location-based services involve several privacy concerns. For example, a location-based server aims to hide the number of people in a region, and this range query can be solved by differential privacy. Cormode *et al.* [14] applied spatial decomposition methods, which are a type of dataset-partitioning mechanisms, to decrease noise. The authors instantiated a hierarchical tree structure to decompose a geometric space from large to small areas with data points partitioned among the leaves. In addition, the authors added noise to the count for each node. Qardaji *et al.* [15] identified the selection of partition granularity to balance errors from two sources as the key challenge in differentially private synopsis methods. The authors proposed a methodology for selecting grid size for the uniform grid method on the basis of the analysis of the dependence of errors on grid size. Li *et al.* [16] proposed a matrix mechanism that can answer sets of linear counting queries. The set of queries, which is defined as a workload, is transformed into matrix $A$, where each row contains the coefficients of a linear query. The essential element of the matrix mechanism is to select $A$ to represent a set of queries. The matrix mechanism can be extended to various approaches based on the selection of $A$. For example, if $A$ is an identity matrix, then this mechanism can be a normal Laplace mechanism for batch queries. Zhang *et al.* [17] created a quadtree for spatial datasets. The authors defined a threshold to determine the minimum of a subdomain and another threshold to

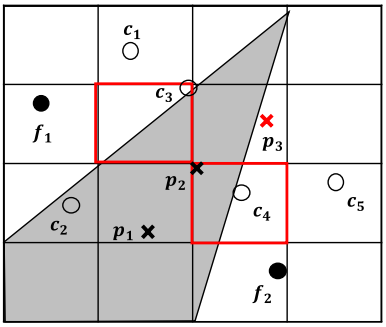



**FIGURE 2.** The limitation of spatial decomposition method.

limit the height of a quadtree. The amount of noise added to the quadtree can be limited to a constant by using the two thresholds. Several researchers have focused intensively on the private spatial decomposition method in spatial differential privacy. However, private spatial decomposition is inadequate in optimal location queries owing to its dataset skewness. When applying private spatial decomposition to the Max-inf problem, accuracy would be poor if the users are not evenly distributed. However, users are usually skewed in real-world applications, and Max-inf problems are often out of alignment with grid cells. Fig. 2 shows this alignment problem. The gray area shows the influence region of $p_1$. Then, $c_3$ and $c_4$ are outside the region, but their grid cells overlap with the region. Since we don't know the exact locations for $c_3$ and $c_4$, it is difficult to decide whether the influence region of $p_1$ include them or not. Therefore, we utilize the Voronoi region-based approach rather than the grid cell-based approach to process the Max-inf problem.

## III. PROBLEM DEFINITION

We formally define the problems. Table 1 summarizes the notations frequently used in the study. All data objects are represented by points in the Euclidean space. Let $d(,)$ denote the Euclidean distance between two points. Then, the Max-inf location selection problem is defined as follows.

*Definition 5: Max-inf location selection query. The Max-inf finds an optimal location that maximizes the influence on clients under the assumption that each client utilizes the nearest facility given existing facilities $F$, potential location $P$, and client location $C$.*

$$MaxINF(P) = \arg\max_{p \in P} \sum_{c \in IS(p)} w(c) \tag{3}$$

$$IS(p) = \{c \in C | \forall f \in F, d(c,p) \leq d(c,f)\} \tag{4}$$

In (3), every client $c \in C$ is associated with a positive weight $w(c)$ that captures the importance of the client. Generally, every client has the same importance. Thus, we set $w(c)$ as 1 for all the clients in this study. As prescribed in Section 1, a random noise drawn from a Laplace distribution can be added to the objective function to find an optimal location with a differentially private approach. The magnitude of the noise depends on sensitivity, and the sensitivity of the counting problem is 1. Thus, the Max-inf problem with a differentially private user location is defined as follows.

**TABLE 1.** Frequently used symbols.

| Notation | Description |
|---|---|
| $o$ | A point in the data space |
| $C, F, P$ | Set of clients, existing facilities, and potential locations, respectively |
| $c, f, p$ | Client in $C$, an existing facility in $F$, and a potential location in $P$, respectively |
| $n, m, l$ | Cardinality of $C$, $F$, and $P$, respectively |
| $d(,)$ | Distance between two points |
| $\epsilon$ | Privacy budget |
| $IS(p)$ | Influence set of $p$, which is a subset of $C$ |
| $V(p)$ | Influence region of a potential location $p$ |
| $OP(p)$ | Overlapped potential location set of $p$, which is a subset of $P$ |
| $VOP(p)$ | Voronoi regions of the overlapped potential location set of $p$ |
| $ncount(p)$ | Noisy count of $p$ |
| $Lap()$ | Noise from Laplace distribution |
| $PV(p)$ | Partitioned influence region of $p$ |
| $VN(p)$ and $VN(f)$ | Voronoi neighbors of potential location $p$ and Voronoi neighbors of existing facility $f$ |
| $CVN(p)$ | Candidate Voronoi neighbors of $p$ |
| $v_i$ | Voronoi vertex |
| $VC(v_i)$ | A circle that passes corresponding Voronoi neighbors, and its center is a Voronoi vertex $v_i$ |
| $VE(p)$ | Voronoi envelope of $p$ |
| $UBncount(f)$ | Upper-bound noisy count of $f$ |
| $R_c, R_f, R_p$ | R-tree of $C$, $F$, and $P$, respectively |
| $N_c, N_f, N_p$ | Node of $R_c$, $R_f$, and $R_p$, respectively |

*Definition 6: Max-inf with a differentially private user location. Given the same object datasets as the Max-inf problem, the DP-Max-inf finds a location that maximizes influence with a differentially private client location.*

$$DPMaxINF(P) = \arg\max_{p \in P} \sum_{c \in IS(p)} w(c) + Lap(1/\epsilon') \tag{5}$$

The DP-Max-inf simply changes the objective function from (3) to (5) by adding a Laplace noise. As described in the previous sections, the influence regions of potential locations overlap each other. Therefore, the DP-Max-inf divides $\epsilon$ by the number of potential locations to achieve total $\epsilon$-differential privacy. Hence, we set the privacy budget as $\epsilon' = \epsilon/|P|$ in the naïve approach. However, if we apply sequential composition to the DP-Max-inf problem in the naïve approach, then it will suffer from highly poor performance accuracy. Therefore, we present an enhanced approach, $SC_{enhanced}$. $SC_{enhanced}$ computes influence regions for each potential location $p_i$. The influence region is a subspace of the Euclidean space in which the customers are the RNN of $p_i$. Then, $SC_{enhanced}$ checks whether they overlap and calculates how much noise should be added to satisfy $\epsilon$-differential privacy. Finally, $SC_{enhanced}$ improves accuracy by providing tighter noise bounds than the naïve approach because not all the influential regions of the potential location overlap in general. The proposed methods exploit the Voronoi diagram [18] of existing facilities and the potential location. The Voronoi diagram is defined as follows.

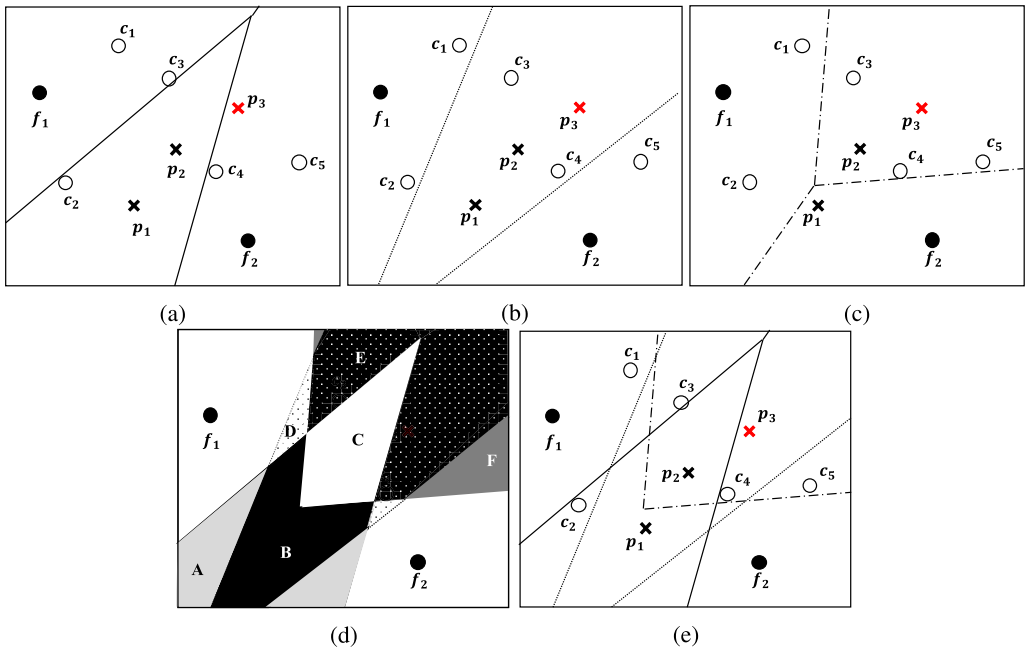



**FIGURE 3.** Example of Max-inf problem.

*Definition 7: Voronoi diagram. The Voronoi diagram of existing facilities $F = \{f_1, \ldots, f_m\}$ partitions the Euclidean space into m regions. Each region includes all points in the Euclidean space with a common closest point in F according to $d(,)$.*

The proposed methods determine the influence regions of each potential location after the Voronoi diagram is constructed. Then, the influence region is the Voronoi cell of the corresponding potential location. The influence region is defined as follows.

*Definition 8: Influence region. For any object o in the Euclidean space, the influence region of the potential location $p \in P$ on the set $F \cup \{p\}$ is a region $V(p)$ that satisfies that for any point $p' \in F \cup \{p\}, p \neq p'$ and for any point $o \in V(p), d(p, o) \leq d(p', o)$.*

The influence region of p encloses all and only the clients in $IS(p)$. We use the influence region to quickly identify $IS(p)$. Thus, we redefine (3) as $IS(p) = \{c | c \in C \wedge c \in V(p)\}$. Next, we determine the overlapped potential location set, in which the influence regions overlap with the influence region of each potential location.

*Definition 9: Overlapped potential location set. For any potential location p and its influence region $V(p)$, an overlapped potential location set is a subset of potential locations that hold the following condition.*

$$OP(p) = \{p' \in P | p' \neq p, V(p) \cap V(p') \neq \varnothing\} \tag{6}$$

Finally, we redefine the objective function of DP-Max-inf with the overlapped potential location set as follows:

$$ncount(p) = |IS(p)| + Lap(|OP(p)|/\epsilon) \tag{7}$$

$$DPMaxINF(P) = \arg\max_{p \in P} ncount(p) \tag{8}$$

## IV. VORONOI REGION PARTITIONING METHOD

Even if we use $SC_{enhanced}$ to solve the DP-Max-inf problem, accuracy degrades exponentially with the maximum cardinality of the overlapped potential location set. Therefore, we propose the VPM to further improve accuracy.

### A. BASELINE APPROACH OF VPM

The VPM changes a sequential composition to a parallel composition; thus, it mitigates the degradation of accuracy. The VPM constructs an influence region of each potential location $p_i \in P$ with existing facilities $F$. Then, it finds the overlapped potential location set $OP(p_i)$, and the VPM enumerates the combinations of overlapped regions. Thereafter, the VPM counts the number of clients who are located in each partitioned region and adds a Laplace noise. Finally, the VPM sums up the noisy count of every partitioned region.

*Definition 10: Partitioned influence region. The meet operation is defined as $Meet(S) = \cap_{u \in S} u$, which is extracted from the basic theorem on Galois lattice [23]. We also define Voronoi regions of potential locations in $OP(p_i)$ as $VOP(p_i) = \{V(p_j) | p_j \in OP(p_i)\}$. Then, given potential location $p_i$, influence region $V(p_i)$ and its overlapped potential location set, $OP(p_i)$, the partitioned influence region, $PV(p_i)$, is a set of regions that holds the following condition.*

$$PV(p_i) = \{V(p_i) \cap Meet(S) \cap Meet(\bar{S}) | S \subseteq VOP(p_i), \bar{S} = VOP(p_i) \setminus S, S \neq \varnothing, \bar{S} \neq \varnothing\} \tag{9}$$

For example, Fig. 3 shows the Voronoi regions of the facilities and the partitioned influence regions of each

potential location presented in Fig. 1. Fig. 3 (a) shows the Voronoi regions, which are composed of $\{f_1, f_2, p_1\}$, and Fig. 3 (b) and (c) are constructed by $\{f_1, f_2, p_2\}$ and $\{f_1, f_2, p_3\}$, respectively. Then, Fig. 3 (d) and (e) show the partitioned influence regions of $\{V(p_1), V(p_2), V(p_3)\}$. The partitioned regions are constructed to compute the noisy count of $p_1$, as shown in Fig. 3 (d). The Voronoi region of $p_1$ is composed of the following partitioned influence regions.

$$A = V(p_1) \setminus (V(p_2) \cup V(p_3))$$
$$B = (V(p_1) \cap V(p_2)) \setminus V(p_3)$$
$$C = V(p_1) \cap V(p_2) \cap V(p_3)$$

As shown in Fig.3 (e), region $A$ contains $c_2$, whereas the other regions of $PV(p_1)$ contain no clients. Let $n_X$ is the noise of the partitioned region $X$. Then, the noisy count of $V(p_1)$ is calculated as follows:

$$\begin{aligned} ncount(p_1) &= (1 + n_A) + (0 + n_B) + (0 + n_C) \\ &= 1 + n_A + n_B + n_C \end{aligned}$$

Similarly, the Voronoi region of $p_2$ is partitioned with regions $\{B, C, D, E\}$, and its noisy count is $2 + n_B + n_C + n_D + n_E$. The noisy counts of regions $B$ and $C$ are reused because they have been computed during the process of $V(p_1)$. If we compute the noisy count of potential location $p$, then the total noise of $ncount(p)$ is proportional to $O(\sqrt{|PV(p)|})$ owing to parallel composition. Therefore, query accuracy is better than $SC_{enhanced}$ if $|PV(p)|$ is less than $|OP(p)|^2$. However, the computation cost of the VPM is extremely high, because the time complexity of finding $PV(p)$ is $O(2^{|PV(p)|})$ in the worst case. Thus, the baseline algorithm of the VPM is based on the divide-and-conquer framework. We utilize the following remark to compute the partitioned influence region.

*Remark 1: Monotonicity of partitioned influence region.*

*Let a set of partitioned influence regions be as* $PV = \{PV_1, PV_2, \ldots, PV_n\}$. *Then, we define* $PV_{-i}$ *as a subset of* $PV$ *except for the* $i$*th region,* $PV_i$. *Then,* $Meet(PV) = \varnothing$ *if any* $PV_{-i}$ *exists, such that* $\cap PV_{-i} = \varnothing$

The overall procedures are described in Algorithms 1 and 2. Algorithm 1 computes the partitioned influence region of each potential location, whereas Algorithm 2 shows the overall query-processing steps of the VPM. The VPM constructs Voronoi regions based on potential locations and partitions them by overlapped regions. Then, the VPM adds a Laplace noise once to each partitioned region, which is a parallel composition. Therefore, the VPM is $\epsilon$-differentially private owing to parallel composition.

**Algorithm 1** Get Partitioned Influence Region (GetPV)

**Input:** $pv$ - Partitioned influence region, $j$ - index of overlapped potential location set, $OP$ - overlapped potential location set, $V$ - Voronoi regions of potential locations
**Output:** $PV(p)$ - A set of partitioned influence regions

```
L ← [ ]
if j is greater than OP.size then
    insert pv into L
else
    p_j ← OP[j]
    v_j ← V[p_j]
    if pv intersects with v_j then
        pv1 ← pv.intersection(v_j)
        pv2 ← pv.difference(v_j)
        L1 ← GetPV(pv1, j + 1, OP, V)
        L2 ← GetPV(pv2, j + 1, OP, V)
        L ← L1 + L2
    else
        L ← GetPV(pv, j + 1, OP, V)
Return L
```

**Algorithm 2** Voronoi Partition Method (VPM)

**Input:** $V$ - Voronoi regions of potential locations, $P$ - potential locations, $C$ - user locations, $\epsilon$ - privacy budget
**Output:** $p_r \in P$ - near-optimal location

```
PQ ← priority queue sorted by user count in decreasing order
for p_i ∈ P do
    OP ← [ ]
    v_i ← V[p_i]
    for v_j ∈ V do
        if v_i interects with v_j then
            add v_j to OP
    L ← GetPV(v_i, 0, OP, V)
    cnt_i ← 0
    for pv_j ∈ L do
        cnt ← count the number of users that pv_j contains
        cnt_i ← cnt_i + cnt + Laplace(1/ε)
    PQ.enqueue(cnt_i, p_i)
Return PQ.top
```

### B. R-TREE-BASED APPROACH

Although the VPM is based on the divide-and-conquer approach, it still suffers from high computational cost in generating the influence region and finding the overlapped potential location set. Therefore, the VPM exploits three r-tree [19] indices, that is, $R_f$ for Voronoi regions of existing facilities, $R_p$ for potential locations, and $R_c$ for client locations. Then, we can use the intersection query to determine the overlapped potential location set. We can also use the range query to compute the partitioned influence region of the potential location and to count the number of users in each partitioned influence region with $R_c$. In $R_f$, the leaf node is composed of MBRs for each Voronoi cell of existing facilities. As explained in the previous section, a Voronoi diagram of $F \cup \{p_i\}$ should be constructed to generate the influence region of each potential location $p_i \in P$. We precompute the candidate Voronoi neighbors of each influence region to reduce the computational cost of generating the influence region. Voronoi neighbors are subsets of existing facilities that are adjacent to a given Voronoi cell. We refer to the edge of a Voronoi cell as a Voronoi edge and each end

point as a Voronoi vertex. A Voronoi edge is a perpendicular bisector of a line segment between two existing facilities. For each Voronoi edge of the existing facility $f$, we refer to the corresponding facility $f' \in F$ as Voronoi neighbors of $f$ denoted $VN(f)$. In addition, $VN(p)$ represents Voronoi neighbors of the influence region of potential location $p$. Since $|VN(p)| \ll F$, if we know $VN(p)$ in advance, then it is possible to reduce the construction time of the influence region. However, it is impossible to pre-compute the Voronoi neighbors; thus, we use candidate Voronoi neighbors instead. Candidate Voronoi neighbors are a superset of Voronoi neighbors. We need to compute the Delaunay triangulation of existing facilities to determine the candidate Voronoi neighbors of the influence region. The Delaunay triangulation [20] of a discrete point set $O$ in the general position corresponds to the dual graph of the Voronoi diagram for $O$. The circumcenters of Delaunay triangles are the vertices of the Voronoi diagram. In the Euclidean space, Voronoi vertices are connected via edges. They can be derived from the adjacency relationships of the Delaunay triangles. If two triangles share an edge in the Delaunay triangulation, then their circumcenters relate to an edge in the Voronoi cell. Thus, we can define the candidate Voronoi neighbors as follows.

*Definition 11: Candidate Voronoi neighbors. Assume that location p is located inside the Voronoi cell $V(f)$. Let DT be a set of Delaunay triangles that consists of existing facilities F and Disk() be a circumcircle covering each delaunay triangle. Then, candidate Voronoi neighbors, $CVN(p)$, are the subset of existing facilities and defined as follows*:

$$CVN(p) = \{f' \in t | t \in DT \wedge Disk(t) \cap V(f) \neq \varnothing\} \quad (10)$$

*Lemma 12:* $VN(p) \subseteq CVN(p)$

*Proof:* The proof can be found in the appendix. □

We can construct the influence region of the potential location with its candidate Voronoi neighbors but not all existing facilities. In addition, we can compute the overlapped potential location set through the Voronoi neighbors of the influence region. Let a potential location $p$ is fixed, and its influence region $V(p)$ and Voronoi neighbors $VN(p)$ are given. Then, each Voronoi vertex corresponds to a pair of Voronoi neighbors. Thus, let the corresponding Voronoi neighbors of Voronoi vertex $v_i \in V(p).vertex$ be $vn_i$ and $vn_i'$. In addition, the circle whose center is each Voronoi vertex $v_i$ and the radius of $d(v_i, vn_i) = d(v_i, vn_i')$ are denoted as $VC(v_i)$. Then, we can easily find the overlapped potential location set by following lemma.

*Lemma 13: Potential location $p'$ is an overlapped potential location of p if and only if $p'$ is inside the union regions of $VC(v_i)$, for all $v_i \in V(p).vertex$.*

*Proof:* The proof can be found in the appendix. □

Therefore, we can easily compute the overlapped potential location set through the range query of $R_p$ based on the above lemma. In conclusion, lines 5 to 7 in Algorithm 2 are changed to invoke the range query of $R_p$ with the influence region of the given potential location. Although we utilize these properties, computational cost is still extremely high if potential locations are skewed. In the next section, we propose the last query-processing algorithm that reduces computational time by slightly sacrificing accuracy to overcome this drawback.

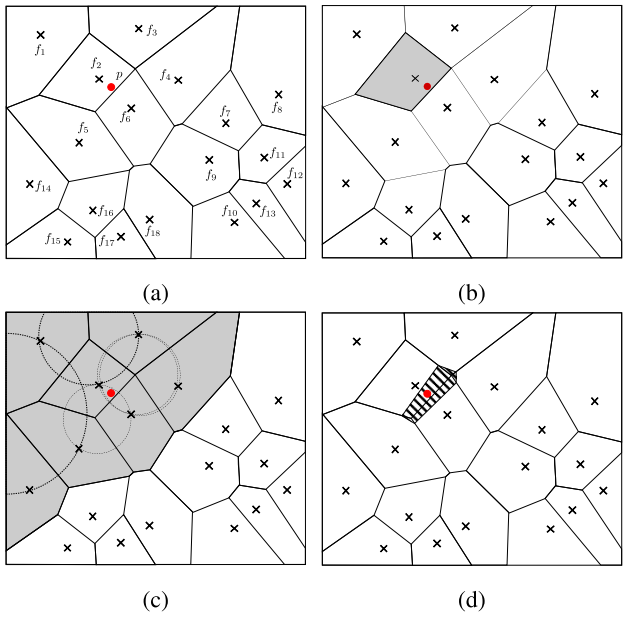



**FIGURE 4.** Voronoi envelope.

## V. VORONOI ENVELOPE FILTERING METHOD (VEM)

### A. BASIC IDEA OF VEM

In this section, we propose a VEM. The VPM suffers from worst case query-processing time. In the worst case, the overlapped potential location set constructs every combination of intersection regions. Thus, the time complexity of the VPM increases exponentially proportional to the cardinality of the overlapped potential location set. Although it is extremely time-consuming, computing the partitioned influence region for query accuracy is inevitable. However, we observe that customers and facilities are generally skewed in the Euclidean space. Most potential locations are less influential among customers than the optimal location. Therefore, we can reduce query-processing time if we know the upper-bound noisy count of the potential location. Motivated by this observation, we filter unnecessary potential locations whose upper-bound noisy count is less than the noisy count of the optimal location. To compute the upper-bound noisy count, we initially determine the candidate influence region of each potential location, which is called the Voronoi envelope. The Voronoi envelope of a potential location p, which is denoted by $VE(p)$, is the union of the Voronoi cells of $CVN(p)$. Fig. 4 shows an example of the Voronoi envelope. We have 15 existing facilities and one potential location p. Fig. 4 (a) shows the Voronoi diagram of existing facilities, and p is located inside the Voronoi cell of $f_2$, as depicted in Fig. 4 (b). Then, the Voronoi envelope of $f_2$ consists of its candidate Voronoi neighbors $\{f_1, f_2, f_3, f_4, f_5, f_6, f_{14}\}$, as shown in Fig. 4 (c). Fig. 4 (d) describes the actual influence region of $p$. As shown in this example, $VE(p)$ is an upper-bound region of $V(p)$.

**Algorithm 3** Voronoi Envelope Filtering method(VEM)

**Input:** $R_f$ - R-tree of existing facilities, $R_p$ - R-tree of potential locations, $\epsilon_2$ - privacy budget
**Output:** $p_r \in P$ - near-optimal location

$p_r \leftarrow \varnothing$
$CB \leftarrow 0$; $R \leftarrow \varnothing$; $PQ \leftarrow$ priority queue
$PQ.enqueue(R_p.root, R_f.root, UBncount(R_f.root))$
**while** $PQ$ is not empty and $CB < PQ.top$ **do**
  $E_p, E_f, UBncount \leftarrow PQ.pop()$
  **if** Both of $E_p$ and $E_f$ are R-tree node **then**
    **for** $N_p \in E_p.children$ and $N_f \in E_f.children$ **do**
      **if** $N_p$ interects with $N_f$ **then**
        $PQ.enqueue(N_p, N_f, UBncount(N_f))$
  **else if** $E_p$ is R-tree node and $E_f$ is not **then**
    **for** $N_p \in E_p.children$ **do**
      **if** $N_p$ interects with $E_f$ **then**
        $PQ.enqueue(N_p, E_f, UBncount(E_f))$
  **else if** $E_f$ is R-tree node and $E_p$ is not **then**
    **for** $N_f \in E_f.children$ **do**
      **if** $N_f$ contains $E_p$ **then**
        $PQ.enqueue(E_p, N_f, UBncount(N_f))$
  **else if** Both of $E_p$ and $E_f$ are not R-tree node **then**
    Generate the Voronoi region of $E_p$ with $E_f$
    Find out the overlapped potential location set of $E_p$
    Compute the Voronoi regions of each of $OP$
    $L \leftarrow GetPV(E_p, 0, OP, V)$
    $cnt_i \leftarrow 0$
    **for** $pv_j \in L$ **do**
      $cnt \leftarrow$ count the users that $pv_j$ contains
      $cnt_i \leftarrow cnt_i + cnt + Laplace(\frac{1}{\epsilon_2})$
    **if** $E_p.noisy\_count > CB$ **then**
      $CB \leftarrow E_p.noisy\_count$
      $p_r \leftarrow E_p$
**Return** $p_r$

Thus, we can compute the upper-bound noisy count of the influence region. As shown in Fig. 4 (c) and (d), the Voronoi envelope is the upper-bound region of any potential location inside a corresponding Voronoi cell. Then, the noisy count of the Voronoi envelope is the upper-bound noisy count of each influence region. Thus, we divide $\epsilon$ into $\epsilon_1$ and $\epsilon_2$, where $\epsilon_1$ is used to compute the noisy count of each Voronoi cell of existing facilities, and $\epsilon_2$ is used for the VPM. The noisy count of each Voronoi cell and its upper-bound count are computed as follows:

$$ncount(f) = |IS(f)| + Lap(1/\epsilon_1) \quad (11)$$

$$\sum_{f' \in CVN(f)} ncount(f') \quad (12)$$

Equation (11) has the same form as (7), except the term for the overlapped potential location set is removed. The Voronoi diagram partitions the entire region, so it is possible to apply parallel composition. Therefore, the sensitivity of (11) is also 1 as same as (7). As we will describe later,

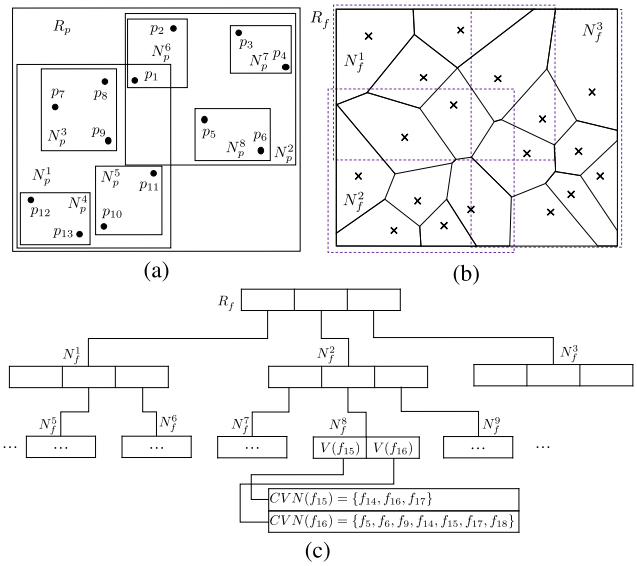



**FIGURE 5.** Potential locations and r-tree of existing facilities.

**TABLE 2.** Upperbound noisy count of Voronoi regions.

| VID | UBncount | VID | UBncount |
|---|---|---|---|
| $V(f_1)$ | 1 | $V(f_{10})$ | 10 |
| $V(f_2)$ | 2 | $V(f_{11})$ | 11 |
| $V(f_3)$ | 3 | $V(f_{12})$ | 12 |
| $V(f_4)$ | 4 | $V(f_{13})$ | 13 |
| $V(f_5)$ | 5 | $V(f_{14})$ | 14 |
| $V(f_6)$ | 6 | $V(f_{15})$ | 15 |
| $V(f_7)$ | 7 | $V(f_{16})$ | 16 |
| $V(f_8)$ | 8 | $V(f_{17})$ | 17 |
| $V(f_9)$ | 9 | $V(f_{18})$ | 18 |

the intermediate nodes of the r-tree only use the noisy count of voronoi regions, so it does not need additional privacy budget due to the post-processing property of differential privacy [8]. The VEM can filter out potential locations whose upper-bound noisy count is less than the current best location during query processing. For this reason, we change $R_f$ to an aggregate R-tree (aR-tree) [24], which is a variation of the R-tree and maintains aggregate information in the intermediate nodes. The leaf nodes of $R_f$ is composed of MBRs for each Voronoi cell, which is the same as the VPM, and their aggregate count is the upper-bound noisy count for each Voronoi cell. In $R_f$, the intermediate node stores the maximum upper-bound noisy count of its children nodes. Let $N_f$ be a node of $R_f$ and $N_f.children$ be its children nodes. Then, the upper-bound of the noisy count of $N_f$ is calculated as follows:

(i) $N_f$ is an intermediate node including the root

$$UBncount(N_f) = \max_{f \in N_f.children} ncount(f) \quad (13)$$

(ii) Otherwise

$$UBncount(N_f) = \max_{N'_f \in N_f.children} UBncount(f) \quad (14)$$

**TABLE 3.** R-tree structure of existing facilities.

| Node ID | Children list | UBncount | Node ID | Children list | UBncount |
|---|---|---|---|---|---|
| $N_f^0$ | $\{N_f^1, N_f^2, N_f^3\}$ | 18 | $N_f^7$ | $\{V(f_5), V(f_{14})\}$ | 14 |
| $N_f^1$ | $\{N_f^4, N_f^5, N_f^6\}$ | 6 | $N_f^8$ | $\{V(f_{15}), V(f_{16})\}$ | 16 |
| $N_f^2$ | $\{N_f^7, N_f^8, N_f^9\}$ | 18 | $N_f^9$ | $\{V(f_{17}), V(f_{18})\}$ | 18 |
| $N_f^3$ | $\{N_f^{10}, N_f^{11}, N_f^{12}\}$ | 13 | $N_f^{10}$ | $\{V(f_7), V(f_8)\}$ | 8 |
| $N_f^4$ | $\{V(f_1), V(f_2)\}$ | 2 | $N_f^{11}$ | $\{V(f_{11}), V(f_{12}), V(f_{13})\}$ | 13 |
| $N_f^5$ | $\{V(f_3), V(f_4)\}$ | 4 | $N_f^{12}$ | $\{V(f_9), V(f_{10})\}$ | 10 |
| $N_f^6$ | $\{V(f_6)\}$ | 6 | | | |

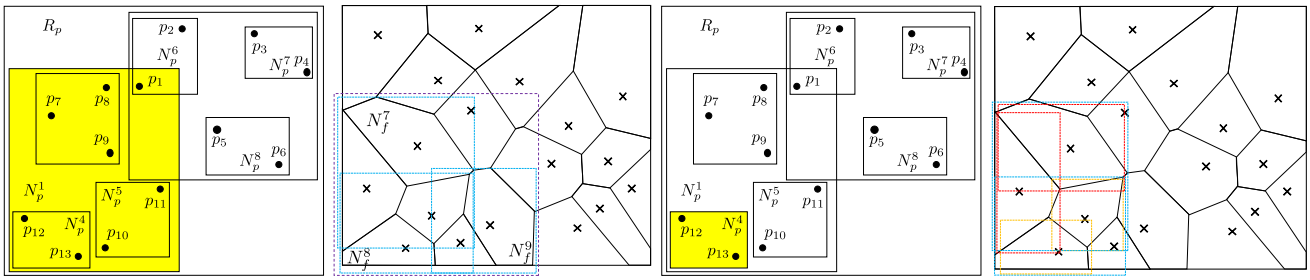



**FIGURE 6.** Query processing steps of VEM.

**TABLE 4.** Experimental settings.

| Parameters | Values |
|---|---|
| # of users | **10K**, 30K, 50K, 70K, 90K |
| # of potential locations | 100, 300, **500**, 700, 900 |
| # of existing facilities | 500, 750, **1000**, 1250, 1500 |
| $\alpha$ (Zipfian distribution) | 0.7, 0.9, **1.1**, 1.3, 1.5 |
| $N$ (Zipfian distribution) | **100** |
| $\epsilon$ ratio | 0.025, 0.05, **0.1**, 0.2, 0.4 |
| Total $\epsilon$ | 0.25, 0.5, **1.0**, 2.0, 4.0 |

If $N_f$ is a leaf node of $R_f$, then the upper-bound noisy count is the sum of the noisy count of its candidate Voronoi neighbors. By contrast, the upper-bound noisy count of the intermediate node is the sum of the upper-bound noisy count of its children nodes. Then, the noisy count of the potential location is less than $UBncount(N_f)$ if its nearest existing facility is a descendent of $N_f$.

### B. QUERY PROCESSING OF VEM

The VEM also exploits three r-trees, namely, $R_p$, $R_f$, and $R_c$, during query processing. It traverses $R_p$ in the best-first search approach [25] while finding the corresponding nodes of $R_f$. The VEM algorithm searches the Voronoi cell of the existing facility, which contains potential locations concurrently and prunes out unnecessary potential locations during the traversing $R_p$. Let $N_p$ be a node of $R_p$ and $E_p(E_f)$ be an entry of $R_p(R_f)$. Algorithm 3 shows the query-processing steps of the VEM. The VEM maintains the triplet $(N_p, N_f, UBncnt(N_f))$ in a maxheap sorted by $UBncnt(N_f)$. Then, the VEM dequeues the triplet $(E_p, E_f, UBcnt)$ of the maxheap at each step. Four cases exist, depending on the types of $E_p$ and $E_f$ as follows:

(i) $E_p$ and $E_f$ are nodes of the R-tree.
(ii) $E_p$ is a potential location point, and $E_f$ is a node of $R_f$.
(iii) $E_p$ is a node of $R_p$, and $E_f$ is a Voronoi cell of the existing facility.
(iv) $E_p$ is a potential location point, and $E_f$ is a Voronoi cell of the existing facility.

In Case (i), the VEM extracts the children of $N_p$ and $N_f$. Next, it finds the pairs of the children $(N_p^c, N_f^c)$ whose MBRs overlap. Then, the VEM computes the upper-bound noisy count of each $N_f^c$ and enqueues the new triplet $(N_p^c, N_f^c, UBncount(N_f^c))$. In Case (ii), the VEM extracts the children of $N_f$ and finds each child node $N_f^c$, which contains $E_p$. Then, the VEM computes the upper-bound noisy count of each $N_f^c$ and enqueues the new triplet $(E_p, N_f^c, UBncount(N_f^c))$. In Case (iii), the VEM extracts the children of $N_p$ and finds each child node, $N_p^c$, which overlaps with $E_f$. The last step is the same as that of the above cases. In Case (iv), the VEM computes the influence region of $E_p$ with $E_f$ and invokes the *GetPV* operation of the VPM algorithm to calculate its noisy count. If its noisy count is greater than the current best, then the VEM updates the current best. The VEM repeats these steps until the current best is greater than the top of the maxheap. The potential locations in Fig. 5 (a) are considered with existing facilities in Fig. 4 (a). Then, the Voronoi diagram and $R_f$ are constructed, as shown in Fig. 5 (b) and (c). Assume that the upper-bound noisy counts of Voronoi regions are given as described in Table 2, and nodes of $r_f$ are constructed as Table 3.

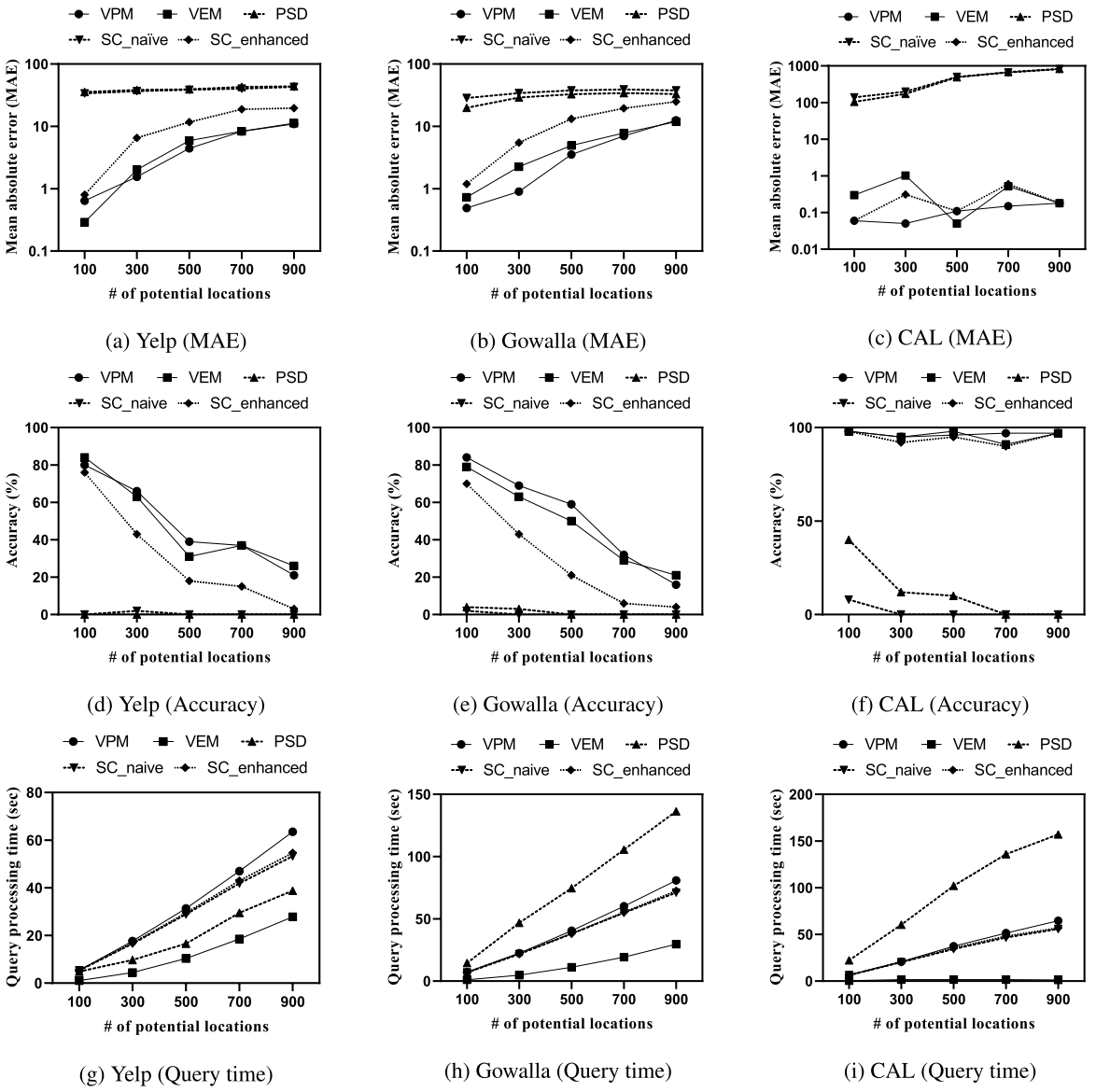



**FIGURE 7.** The experimental results at varying potential locations.

In this example, $p_{12}$ is in $V(f_{14})$, and $p_{13}$ is in $V(f_{15})$. Fig. 6 shows the steps to find corresponding existing facilities of $p_{12}$ and $p_{13}$ with $R_p$ and $R_f$. Initially, the VEM enqueues the root nodes of $R_p$ and $R_f$. Then, the VEM finds that $N_p^1$ intersects with $N_f^1$, $N_f^2$, and $N_f^3$ at the next iteration. The VEM composes the tiplet, $(N_p^1, N_f^1, 6)$ and enqueues it into the priority queue. Further, $(N_p^1, N_f^2, 18)$ and $(N_p^1, N_f^3, 13)$ are inserted into the maxheap. In addition, $(N_p^2, N_f^1, 6)$, $(N_p^2, N_f^2, 18)$ and $(N_p^2, N_f^3, 13)$ are also inserted, because $N_p^2$ also intersects with $N_f^1$, $N_f^2$, and $N_f^3$. Assume that ties are broken, $(N_p^1, N_f^2, 18)$ is popped out at next iteration. Then, the VEM computes the intersection children pairs of $N_p^1$ and $N_f^2$. $N_p^4$, the child node of $N_p^1$ intersects with $N_f^7$ and $N_f^8$, as depicted in Fig. 6 (a) and (b), respectively. Then, $(N_p^4, N_f^7, 14)$ and $(N_p^4, N_f^8, 16)$ are inserted in the maxheap. Further, $(N_p^5, N_f^7, 14)$, $(N_p^5, N_f^8, 16)$, $(N_p^5, N_f^9, 18)$ and $(N_p^6, N_f^7, 14)$ are inserted in the same manner. Continuously, the VEM dequeues $(N_p^5, N_f^9, 18)$ and finds out that $p_{10}$ is in $V(f_{17})$ and $p_{11}$ is in $V(f_{18})$. Then, the VEM generates the influence region of $p_{11}$ and finds out its overlapped potential location set. Finally, the VEM invokes the VPM with $p_{11}$ and computes its noisy count. If the noisy count of $p_{11}$ is greater than 18, the algorithm is terminated, because $(N_p^2, N_f^2, 18)$ is the top of priority queue. Otherwise, the algorithm repeats until it satisfies the terminating condition. After several iterations later, $(p_{12}, V(f_{14}), 14)$ and $(p_{13}, V(f_{15}), 15)$ are popped out, if the terminating condition is not meet. Then, the influence regions of $p_{12}$ and $p_{13}$ are constructed, and the VEM invokes the VPM with $p_{12}$ and $p_{13}$. The overall algorithm is described in the appendix due to the lack of space.

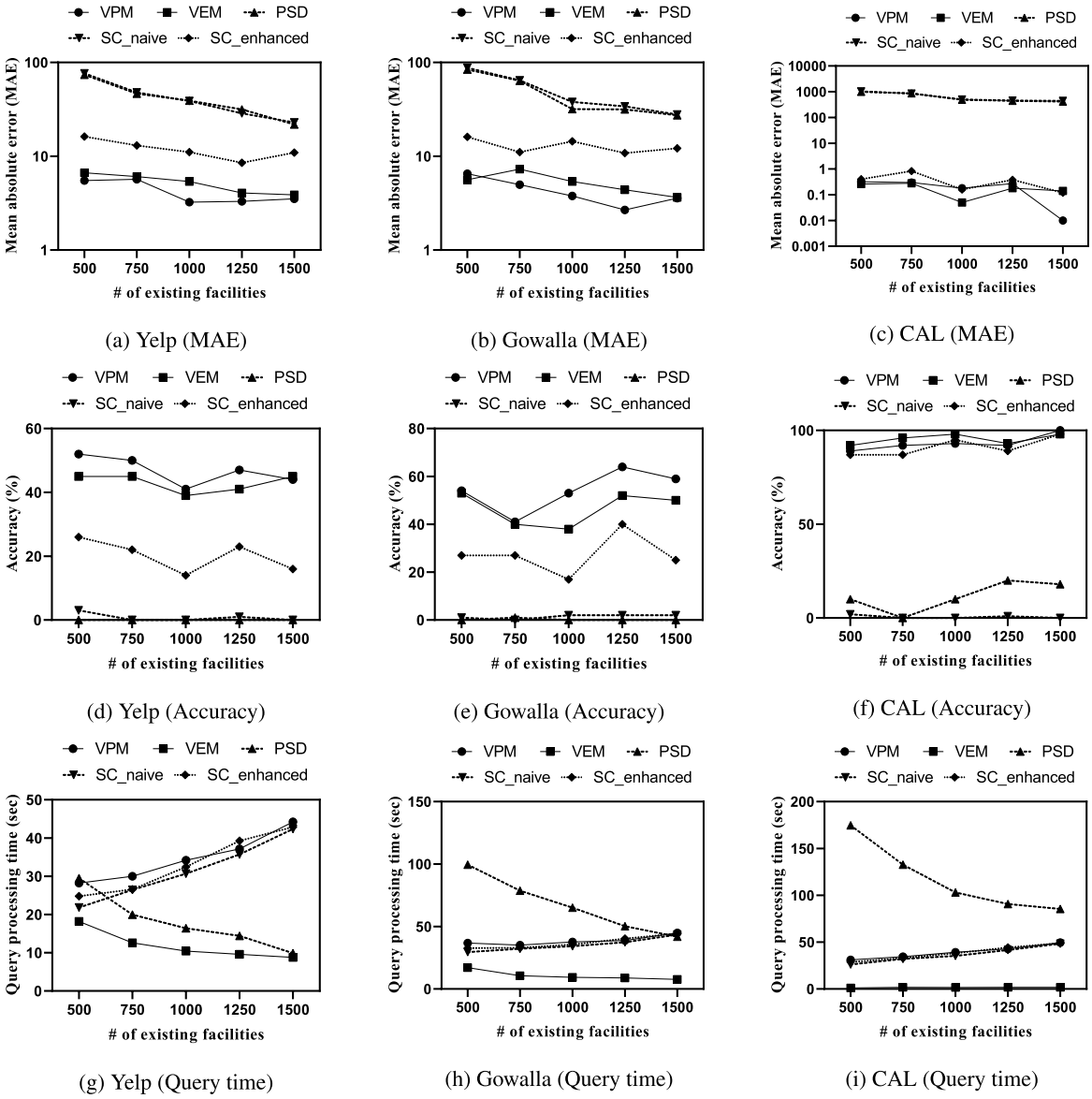


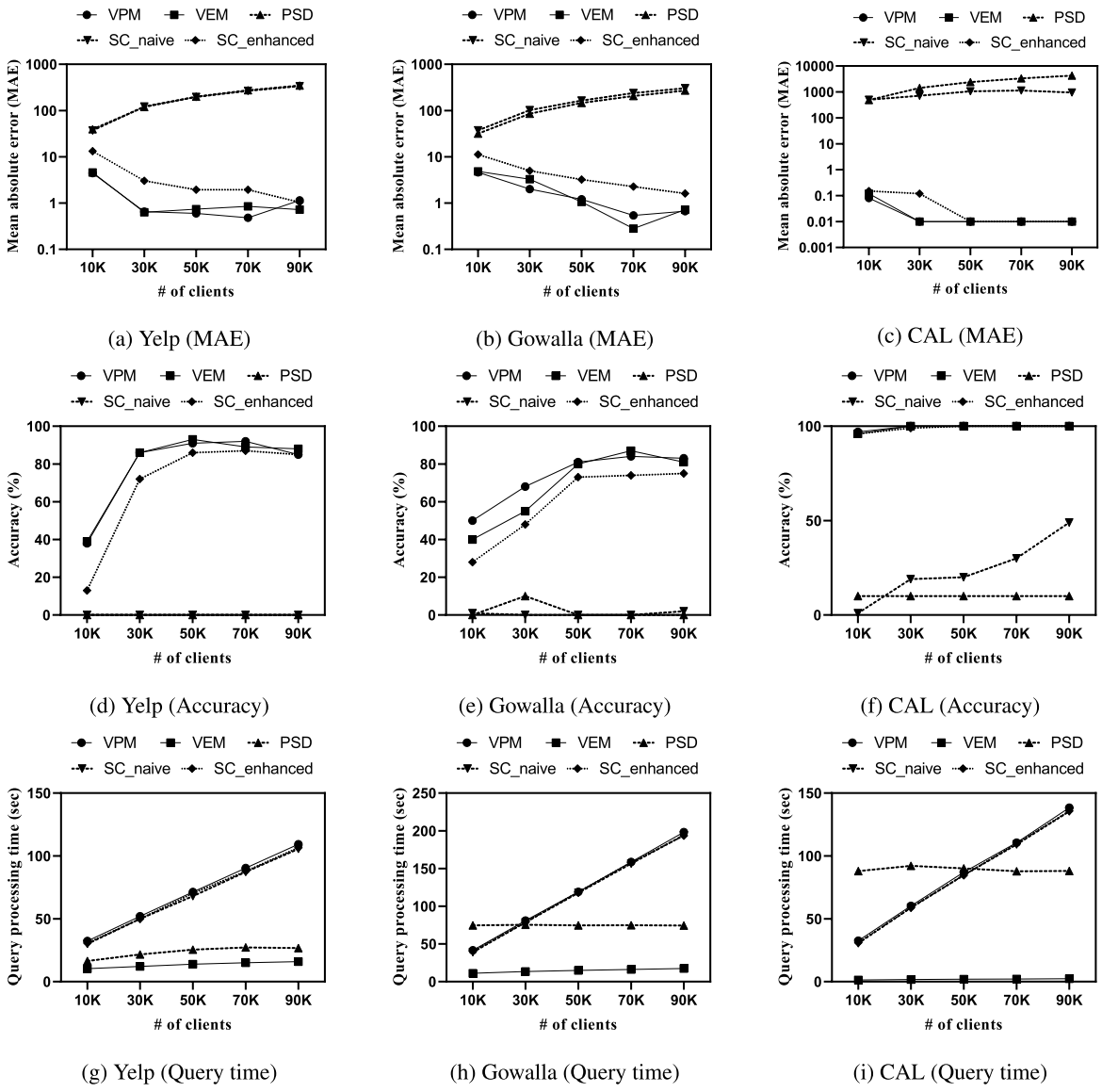

**FIGURE 8.** The experimental results at varying existing facilities.

## VI. EXPERIMENTAL RESULTS

In this section, we report the experimental results of the proposed method. Section 6.2 studies the behavior of the methods with varying dataset sizes. Section 6.3 evaluates the performance of the methods with varying parameters.

### A. EXPERIMENTAL ENVIRONMENTS

We perform experiments to evaluate our proposed VPM and VEM methods by using three datasets, namely, Yelp [21], CAL [22] and Gowalla [27]. The Yelp dataset consists of location and check-in data and is a location-based social network service. The user datasets, which comprise a set of 1,326,097 users, are generated by check-in data. A total of 174,566 points of interest (POIs) exists in the business data. Therefore, we divide the POIs into two datasets. One dataset is for potential locations, and the other is for existing facilities. Then, we randomly select 100 to 900 points from potential locations. The CAL dataset is an actual road network dataset from California that contains 21,048 vertices, 21,693 edges, and 85,070 POIs. In this case, we divide the POIs into two sections to form potential locations and existing facilities. Next, we randomly generate client locations based on Zipfian distribution varying the skewness of the data. We randomly select the POIs in CAL dataset and find the nearest neighbor facilities of each POI. Then, we compute the nearest neighbor distance, and we generate distance of each user from zipf distribution. Finally, we randomly generate 2D positional values with



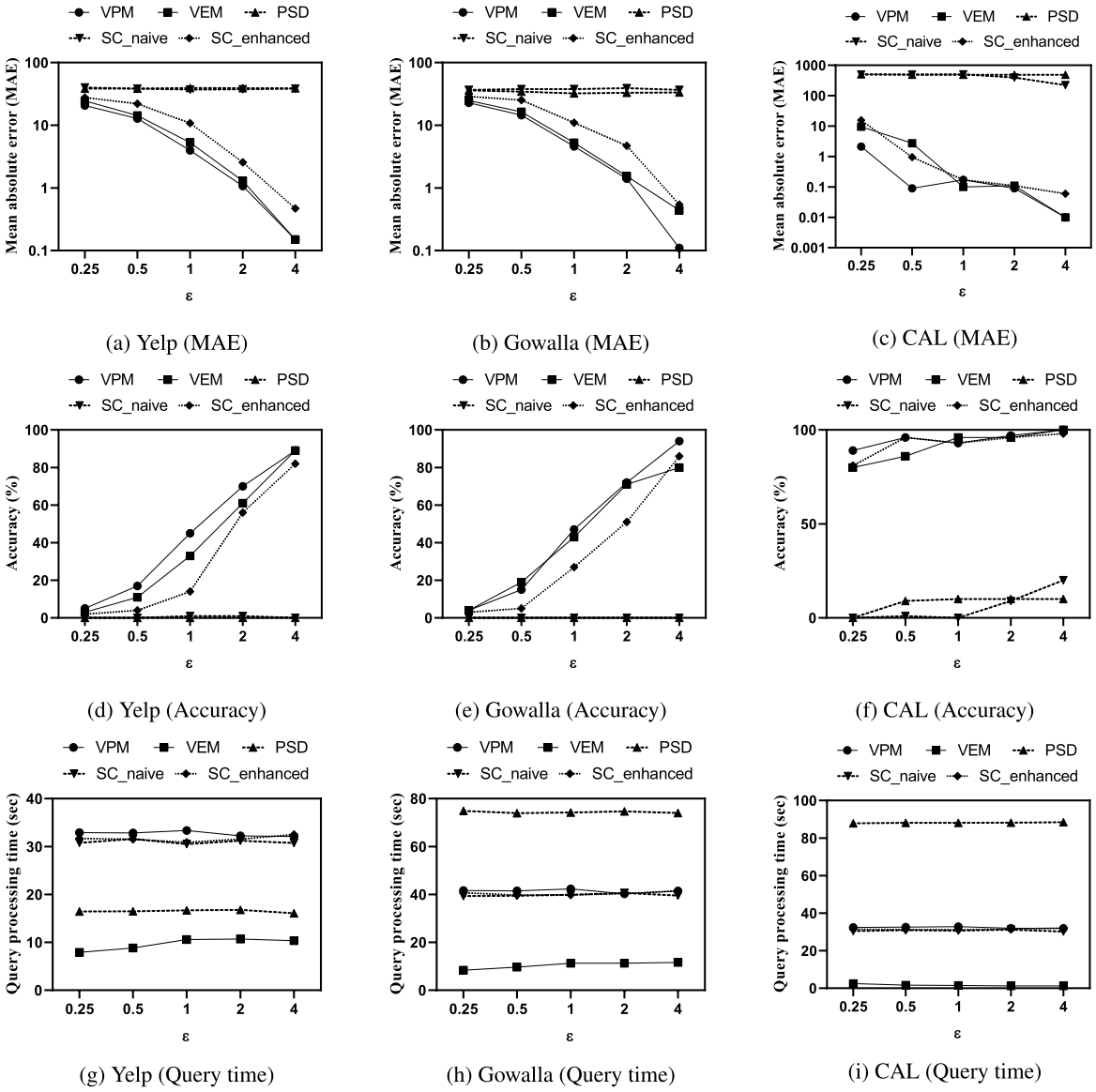

(a) Yelp (MAE) (b) Gowalla (MAE) (c) CAL (MAE)

(d) Yelp (Accuracy) (e) Gowalla (Accuracy) (f) CAL (Accuracy)

(g) Yelp (Query time) (h) Gowalla (Query time) (i) CAL (Query time)

**FIGURE 9.** **The experimental results at varying number of users.**

this distance. The Gowalla dataset has only check-in data of users. There are 2,548,428 user histories, and the users visit total 706,947 POIs. Thus, we select 50,000 potential locations and existing facilities, respectively. Then, we make the other 606,947 POIs as client locations to match the ratio of POIs to client location with the Yelp dataset. To measure accuracy with generality, all data sets were divided into 10 subdata sets, and the experiments are conducted 10 times with varying parameter values with each subdata sets. Table 4 shows a summary of the experimental setup in which the bold texts are default settings. The experiments are conducted with an Ubuntu 14.04 operation system with an Intel(R) Xeon(R) CPU E5-2620 v2@2.10 GHz and 64 GB RAM. We compare SC_naive (based on sequential composition), SC_enhanced (the enhanced version of sequential composition), the PSD (private spatial decomposition), the VPM, and the VEM. The PSD is based on DAWA[16] which is one of the best private spatial decomposition methods in 2D data [26]. We set the system parameter of DAWA as same as [16]. To find out optimal location with PSD index, we construct each influence region of potential locations. Then, we compute the overlapping areas of the regions and PSD index. Next, we compute the noisy count of each potential location with overlapping ratio. Finally, we find out the potential location with the highest noisy count. We measure mean absolute error (MAE), query accuracy, and query processing time of optimal location selection. Let $p_o(i)$ be an actual optimal location and $p_m(i)$ be the query output of each method in $i_{th}$ iteration. Then, the query error means the difference between the number of clients attracted by $p_o(i)$ and $p_m(i)$. Meanwhile, the query accuracy is the number of times that $p_m(i)$ is same with $p_o(i)$. With the number of test case $n_t$, the MAE and query accuracy




(a) Yelp (MAE) (b) Gowalla (MAE) (c) CAL (MAE)

(d) Yelp (Accuracy) (e) Gowalla (Accuracy) (f) CAL (Accuracy)

(g) Yelp (Query time) (h) Gowalla (Query time) (i) CAL (Query time)

**FIGURE 10.** The experimental results at varying privacy budget.

are calculated as follows:

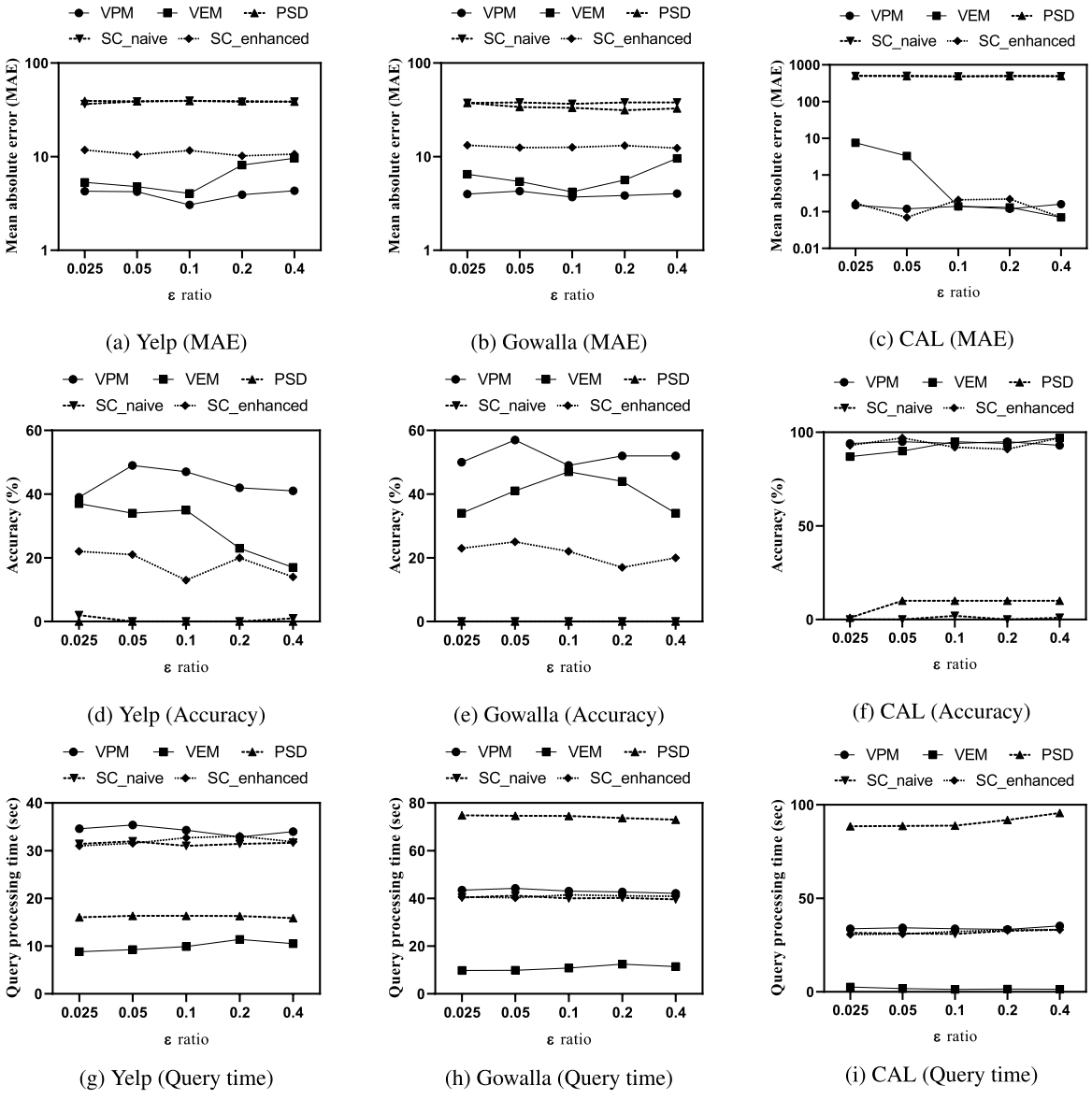
$$MAE = \sum_{i=1}^{n_t} \frac{|IS(p_o(i))| - |IS(p_m(i))|}{n_t} \tag{15}$$

$$accuracy = \sum_{i=1}^{n_t} \frac{I(p_m(i))}{n_t} \tag{16}$$

$$I(x) = \begin{cases} 0 & x \neq p_o(i) \\ 1 & x = p_o(i) \end{cases} \tag{17}$$

### B. SCALABILITY TEST

Fig. 7 shows the results of potential location scalability. We evaluate accuracy, MAE, and query time by varying the number of potential locations from 100 to 900. As depicted in the figures, all schemes degrade performance in terms of MAE and accuracy as the cardinality of the potential location set increases. However, the VPM and the VEM outperform the other methods in all the experiments. In the Gowalla dataset, synthetic client data is highly skewed, so the proposed techniques seem to be significantly better than the other methods. As Fig. 7 (g), (h) and (i) show, the VPM suffers from poor scalability of query-processing time. Meanwhile, the VEM outperforms the other methods in terms of query processing time, and its query accuracy is similar to that of the VPM. The scalability of existing facilities is depicted in Fig. 8. It shows that the accuracy of all the methods increases as the number of existing facilities increases except accuracy in the Yelp dataset. Fig. 8(d) shows that all the methods slightly degrade in accuracy,

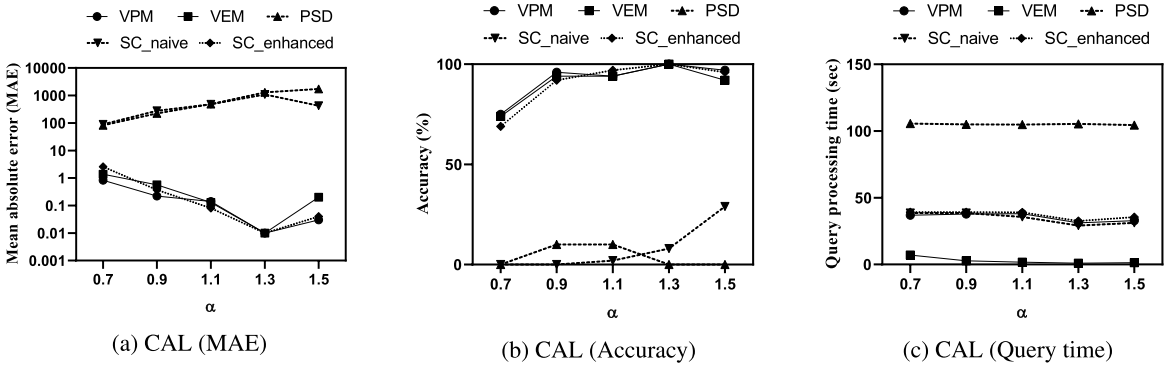



(a) Yelp (MAE) (b) Gowalla (MAE) (c) CAL (MAE)

(d) Yelp (Accuracy) (e) Gowalla (Accuracy) (f) CAL (Accuracy)

(g) Yelp (Query time) (h) Gowalla (Query time) (i) CAL (Query time)

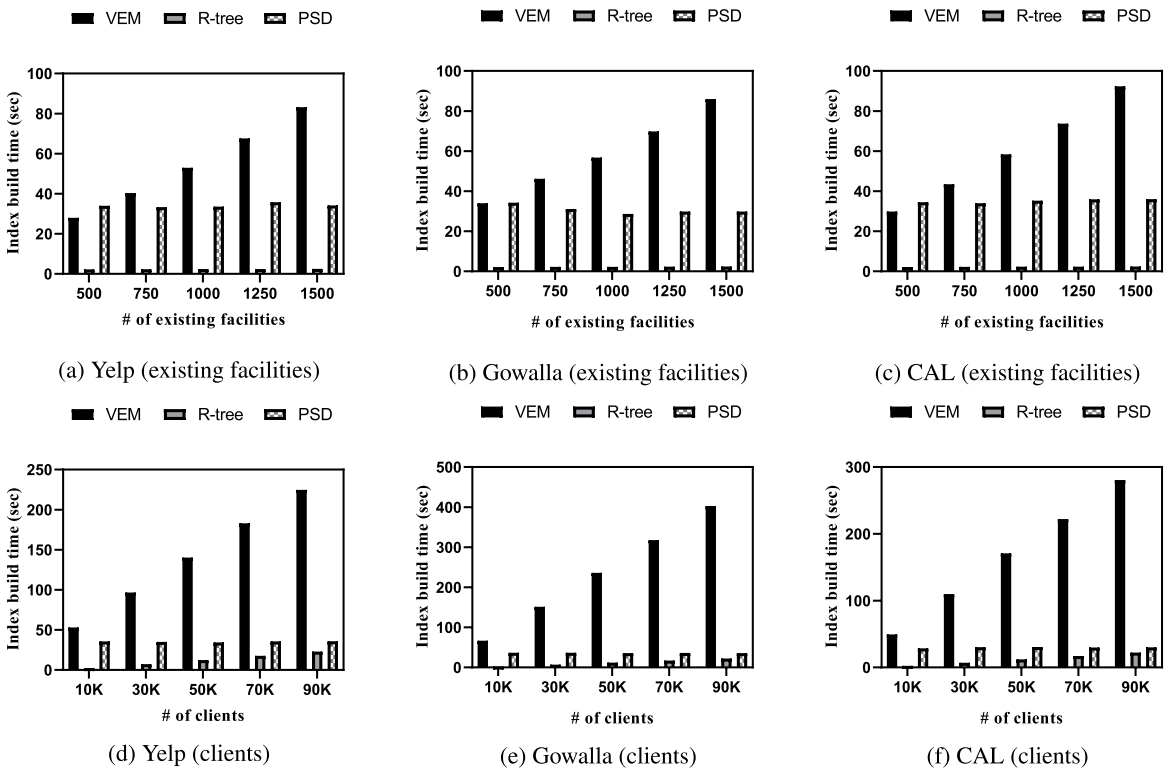
**FIGURE 11.** The experimental results at varying $\epsilon$ ratio.




(a) CAL (MAE) (b) CAL (Accuracy) (c) CAL (Query time)

**FIGURE 12.** The experimental results at varying $\alpha$.

but there are not significant change. Moreover, the VPM and the VEM still outperforms the other methods. The areas of the potential locations depend on the existing facilities. As the number of existing facilities increases, the influence region of each potential location would be smaller. Therefore, the influence of each potential location decreases,




(a) Yelp (existing facilities) (b) Gowalla (existing facilities) (c) CAL (existing facilities)

(d) Yelp (clients) (e) Gowalla (clients) (f) CAL (clients)

**FIGURE 13.** Index build time.

as the cardinality of existing facility increases. As a result, the overall query accuracy decreases because it is highly affected by the inserted noise. The results of query time are depicted in Fig. 8 (g), (h) and (i). The query processing time increases in overall methods except the PSD and the VEM, as the number of existing facilities increases. It is because the computing time of overlapping area decreases in the PSD and the probability of pruning increases in the VEM due to the small size of influence region. In conclusion, the VPM and the VEM outperform the other methods in terms of accuracy and MAE, but the VPM still suffers from query-processing time, as described in Fig. 8. The results of user scalability are depicted in Fig. 9. It shows that the accuracy of all the methods increases as the number of users increases. By contrast, the MAE decreases as the number of users increases. As the number of users increases, the influence of individual potential locations increases. As a result, the accuracy of the query is less affected by the inserted noise. The query processing time increases in all the methods, as the number of users increases. However, the PSD and the VEM are less affected than the others. The PSD just computes the overlapping areas of each influence region and the DAWA index, so the query processing time is independent of the number of users. The VEM prunes out unnecessary potential locations, so the query processing time slightly increases. In all datasets, the VPM and the VEM outperform the other methods in terms of accuracy and MAE, but the VPM still suffers from query-processing time.

### C. PARAMETER TEST AND INDEX BUILD TIME

The second set of experiments demonstrates performance evaluation with varying parameter values. We evaluate the effectiveness of the total privacy budget $\epsilon$ and $\epsilon$ ratio. As explained in Section 5.1, we divide the total privacy $\epsilon$ into $\epsilon_1$ and $\epsilon_2$. The $\epsilon$ ratio indicates the budget amount used to construct the index. When the $\epsilon$ ratio is $r$, the total privacy budget is computed as follows:

$$\epsilon = \epsilon_1 + \epsilon_2 \tag{18}$$

$$\epsilon_1 = \epsilon \times r \tag{19}$$

$$\epsilon_2 = \epsilon \times (1 - r) \tag{20}$$

We evaluate accuracy and MAE by varying the total privacy budget $\epsilon$ from 0.25 to 4. The experimental results are shown in Fig. 10. As expected, the accuracy of the overall methods increases substantially with the increase in $\epsilon$ except the SC_naive and PSD. In general, the VPM outperforms the other methods, and the VEM performs second best. However, the VEM is the best in terms of query-processing time. The query processing time is not affected by the privacy budget. The results of the experiments that vary the $\epsilon$ ratio are shown in Fig. 11. These figures demonstrate that the VEM exhibits similar accuracy and MAE, but it has the best when $\epsilon$ ratio is 0.1. Therefore, we select the default value of $\epsilon$ ratio as 0.1. The $\epsilon$ ratio does not affect the query processing time as same as the privacy budget. Fig. 12 shows the results of experiments at varying clients distribution skewness ($\alpha$). As $\alpha$ grows, the clients get to be more skewed.

The query accuracy of the PSD decreases in proportion to $\alpha$ as depicted in Fig. 12 (a) and (b). The query processing time is not affected by the client skewness, as shown in Fig. 12 (c). Fig. 13 describes the results of the last experiments for the index build time. We compare the index build time of the VEM with the R-tree and the DAWA index structures. Since the scalability of potential locations does not affect the build time of the VEM, we conducts the experiments for the existing facilities and clients. The VEM has the poor performance in all datasets. However, index build time is not on-demand service time, but precomputation time.

## VII. CONCLUSION AND FUTURE WORKS

We propose a differentially private method to protect user locations in the query processing of Max-inf location selection. To the best of our knowledge, this study is the first to point out the influence region overlapping problem when applying differential privacy to Max-inf problems. It is possible to find an adequate location for various purposes, such as analysis of trade areas and establishment of public facilities, while preserving the individual location of users. Also, we present query processing methods VPM and VEM to improve query accuracy and to reduce query processing time. In this work, the VEM is the best method in terms of accuracy and query-processing time. In the future, we will perform additional experiments to obtain other appropriate parameters to improve performance on various datasets.

## APPENDIX

*Proof of Lemma 12:* Suppose that $VN(p) \nsubseteq CVN(p)$. Then, $\exists f \in VN(p)$ such that $f \notin CVN(p)$ and $\exists o \in V(f)$, such that $d(o,f) > d(o,p)$. Without loss of generality, assume that $o$ is inside $\Delta f v_i v_j$, such that $v_i$ and $v_j$ are the adjacent Voronoi vertices of $f$. Let the projection point of $o$ onto $\overline{v_i v_j}$ be as $o'$, $f$ onto $\overline{v_i v_j}$ be as $m_f$, and $p$ onto $\overline{v_i v_j}$ be as $m_p$.Then, $d(f,o) < d(f,o')$. There are 3 cases as follows:

(i) $o'$ is on $\overline{v_i m_f}$:

$$\begin{aligned} d(f,o')^2 &= d(f,m_f)^2 + d(o',m_f)^2 \\ &= d(f,m_f)^2 + (d(v_j,o') - d(v_j,m_f))^2 \\ &= d(v_j,f)^2 - 2d(v_j,m_f)d(v_j,o') + d(v_j,o')^2 \\ &< d(v_j,p)^2 - 2d(v_j,m_f)d(v_j,o') + d(v_j,o')^2 \\ &< d(v_j,p)^2 - 2d(v_j,m_p)d(v_j,o') + d(v_j,o')^2 \\ &= d(p,m_p)^2 + (d(v_j,o') - d(v_j,m_p))^2 \\ &= d(p,m_p)^2 + d(o',m_p)^2 \\ &= d(p,o')^2 \end{aligned}$$

First inequality is hold by assumption, $d(v_j,f) < d(v_j,p)$ and second inequality follows from that $d(v_j,m_f) > d(v_j,m_p)$.

(ii) $o'$ is on $\overline{m_f m_p}$:

$$\begin{aligned} d(f,o')^2 &= d(f,m_f)^2 + d(o',m_f)^2 \\ &= d(f,m_f)^2 + (d(v_j,m_f) - d(v_j,o'))^2 \\ &= d(v_j,f)^2 - 2d(v_j,m_f)d(v_j,o') + d(v_j,o')^2 \\ &< d(v_j,p)^2 - 2d(v_j,m_f)d(v_j,o') + d(v_j,o')^2 \\ &< d(v_j,p)^2 - 2d(v_j,m_p)d(v_j,o') + d(v_j,o')^2 \\ &= d(p,m_p)^2 + (d(v_j,m_p) - d(v_j,o'))^2 \\ &= d(p,m_p)^2 + d(o',m_p)^2 \\ &= d(p,o')^2 \end{aligned}$$

The inequality follows that $d(v_i,f) < d(v_i,p)$ and $d(v_i,m_f) > d(v_i,m_p)$.

(iii) $o'$ is on $\overline{m_p v_j}$:

$$\begin{aligned} d(f,o')^2 &= d(f,m_f)^2 + d(o',m_f)^2 \\ &= d(f,m_f)^2 + (d(v_j,m_f) - d(v_j,o'))^2 \\ &= d(v_j,f)^2 - 2d(v_j,m_f)d(v_j,o') + d(v_j,o')^2 \\ &< d(v_j,p)^2 - 2d(v_j,m_p)d(v_j,o') + d(v_j,o')^2 \\ &= d(p,m_p)^2 + (d(v_j,m_p) - d(v_j,o'))^2 \\ &= d(p,m_p)^2 + d(o',m_p)^2 \\ &= d(p,o')^2 \end{aligned}$$

The inequality follows that $d(v_i,f) < d(v_i,p)$ and $d(v_i,m_f) > d(v_i,m_p)$.

Then, $d(f,o) < d(f,o') < d(p,o') < d(p,o)$. It contradicts to the assumption that $d(f,o) > d(p,o)$. Therefore, $VN(p) \subseteq CVN(p)$.

□

*Proof of Lemma 13:* Suppose that $p' \in OP(p)$ and $p'$ is outside $\cup_{v_i \in V(p).vertex} VC(v_i)$. Then, $\exists o \in V(p)$ such that $\forall f \in F, d(o,f) > d(o,p')$. Without loss of generality, assume that $o$ is inside of $\Delta p v_i v_j$, such that $v_i$ and $v_j$ are the adjacent Voronoi vertices of $p$. Let denote their corresponding Voronoi neighbor be as $f^*$. Then, $d(v_i,f^*) = d(v_i,p)$ and $d(v_j,f^*) = d(v_j,p)$. The followings are hold by the assumption.

$$\begin{aligned} d(f^*,o) &\le \max(d(f^*,v_i), d(f^*,v_j)) \\ &\le \max(d(p,v_i), d(p,v_j)) \\ &< d(p',o) \end{aligned}$$

This contradicts to that $\forall f \in F, d(o,f) > d(o,p')$. Therefore, if $p' \in OP(p)$, then $p'$ is outside $\cup_{v_i \in V(p).vertex} VC(v_i)$.

□

## REFERENCES

[1] S. Cabello, J. M. Diaz-Banez, S. Langerman, C. Seara, and I. Ventura, "Reverse facility location problems," in *Proc. CCCG*, Aug. 2005, pp. 68–71.

[2] R. C.-W. Wong, M. T. Özsu, P. S. Yu, A. W.-C. Fu, and L. Liu, "Efficient method for maximizing bichromatic reverse nearest neighbor," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 1126–1137, Aug. 2009, doi: 10.14778/1687627.1687754.

[3] D. Yan, R. C.-W. Wong, and W. Ng, "Efficient methods for finding influential locations with adaptive grids," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Oct. 2011, pp. 1475–1484, doi: 10.1145/2063576.2063788.

[4] D. Zhang, Y. Du, T. Xia, and Y. Tao, "Progressive computation of the min-dist optimal-location query," in *Proc. 32nd Int. Conf. Very Large Data Bases Endowment*, Sep. 2006, pp. 643–654.

[5] J. Qi, R. Zhang, L. Kulik, D. Lin, and Y. Xue, "The min-dist location selection query," in *Proc. IEEE 28th Int. Conf. Data Eng.*, Apr. 2012, pp. 366–377, doi: 10.1109/ICDE.2012.45.

[6] X. Xiao, B. Yao, and F. Li, "Optimal location queries in road network databases," in *Proc. IEEE 27th Int. Conf. Data Eng.*, Apr. 2011, pp. 804–815.
[7] Z. Chen, Y. Liu, R. C.-W. Wong, J. Xiong, G. Mai, and C. Long, "Efficient algorithms for optimal location queries in road networks," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, Jun. 2014, pp. 123–134, doi: 10.1145/2588555.2612172.
[8] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.
[9] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proc. 35th SIGMOD Int. Conf. Manage. Data (SIGMOD)*, Jun. 2009, pp. 19–30, doi: 10.1145/1559845.1559850.
[10] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries," in *Proc. SIGMOD*, May 2000, pp. 201–212.
[11] M. L. Yiu, D. Papadias, N. Mamoulis, and Y. Tao, "Reverse nearest neighbors in large graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 4, pp. 540–553, Apr. 2006.
[12] A. Vlachou, C. Doulkeridis, K. Nørvåg, and Y. Kotidis, "Branch-and-bound algorithm for reverse top-k queries," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, Jun. 2013, pp. 481–492.
[13] J. Lu, Y. Lu, and G. Cong, "Reverse spatial and textual k nearest neighbor search," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, Jun. 2011, pp. 349–360.
[14] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu, "Differentially private spatial decompositions," in *Proc. IEEE 28th Int. Conf. Data Eng.*, Apr. 2012, pp. 20–31, doi: 10.1109/icde.2012.16.
[15] W. Qardaji, W. Yang, and N. Li, "Differentially private grids for geospatial data," in *Proc. IEEE 29th Int. Conf. Data Eng. (ICDE)*, Apr. 2013, pp. 757–768, doi: 10.1109/icde.2013.6544872.
[16] C. Li, M. Hay, G. Miklau, and Y. Wang, "A data- and workload-aware algorithm for range queries under differential privacy," *Proc. VLDB Endowment*, vol. 7, no. 5, pp. 341–352, Jan. 2014, doi: 10.14778/2732269.2732271.
[17] J. Zhang, X. Xiao, and X. Xie, "PrivTree: A differentially private algorithm for hierarchical decompositions," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, Jun. 2016, pp. 155–170, doi: 10.1145/2882903.2882928.
[18] F. Aurenhammer, "Voronoi diagrams—A survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, Sep. 1991, doi: 10.1145/116873.116880.
[19] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. SIGMOD*, Jun. 1984, pp. 47–57.
[20] K. Buchin, O. Devillers, W. Mulzer, O. Schrijvers, and J. Shewchuk, "Vertex seletion for 3D delaunay triangulations," in *Proc. ESA*, in Lecture Notes in Computer Science, vol. 8125, 2013, pp. 253–264.
[21] *Yelp Open Dataset*. Accessed: Apr. 3, 2019. [Online]. Available: https://www.yelp.com/dataset/challenge
[22] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng, "On trip planning queries in spatial databases," in *Advances in Spatial and Temporal Databases*. Real Datasets for Spatial Databases: Road Networks and Points of Interest, 2005, pp. 273–290. Accessed: Feb. 13, 2019. [Online]. Available: https://www.cs.utah.edu/~lifeifei/SpatialDataset.htm
[23] G. Birkho, *Lattices Theory*, vol. 25, 3rd ed. Providence, RI, USA: AMS, 1967.
[24] I. Lazaridis and S. Mehrotra, "Progressive approximate aggregate queries with a multi-resolution tree structure," in *Proc. SIGMOD Conf.*, 2001, pp. 401–412.
[25] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *ACM Trans. Database Syst.*, vol. 24, no. 2, pp. 265–318, Jun. 1999.
[26] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, and D. Zhang, "Principled evaluation of differentially private algorithms using DPBench," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, 2016, pp. 139–154.
[27] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 1082–1090.

**SEHWA PARK** was born in Seoul, South Korea, in 1986. He received the B.E. and M.E. degrees in computer science and engineering from Sogang University, in 2013 and 2015, respectively, where he is currently pursuing the Ph.D. degree in computer science and engineering. His research interests include spatial database, spatial data query processing, location-based service, information retrieval, and data privacy.

**JUNKYU LEE** was born in Anyang, South Korea, in 1995. He received the B.E. degree in computer science and engineering from Sogang University, in 2018, where he is currently pursuing the M.E. degree in computer science and engineering. His research interests include spatial database, spatial data query processing, location-based service, and spatial group query.

**SEOG PARK** (Member, IEEE) was born in Kyungju, South Korea, in 1956. He received the B.S. degree from Seoul National University and the M.S. degree and the Ph.D. degree in computer science from the Korea Advanced Institute of Science and Technology, in 1980 and 1983, respectively. From 2002 to 2003, he was a Visiting Professor with the University of Virginia, USA. He has been a member of the Korean Institute of Information Science and Engineers (KIISE), since 1980. He has been a Faculty Member with Sogang University, since 1983. His research interests include data privacy, differential privacy, database security, spatial database, location-based service, information retrieval, real-time systems, multimedia database systems, digital library, data warehouse, role-based access control, and web database. From 1999 to 2007, he was a member of the Database Systems for Advanced Application (DASFAA) Steering Committee. He was the Panel Co-Chair of Very Large Data Bases (VLDB), in 2006.