The Journal of Supercomputing (2022) 78:18336–18360
https://doi.org/10.1007/s11227-022-04601-5

Nearly most influential location selection with differentially private user locations in a road network

Sehwa Park[1] · Seog Park[1]

Accepted: 6 May 2022 / Published online: 6 June 2022

Abstract

During the past decades, maximum influential location selection (Max-inf) problems have been of intense interest to the spatial database community. The Max-inf problem searches for a location that attracts as many clients as possible, so it is essential to collect the location information of each client for such a query. However, the client location is considered sensitive information, and location privacy has become an emerging issue. To resolve the privacy issue, we present a novel Max-inf problem in a differentially private manner, which is called DP-Max-inf in a road network. Differential privacy is a de-facto standard privacy protection technique that injects controlled noise into statistical query results. In addition, we present the influence region overlapping problem while applying differential privacy to the Max-inf problem using the conventional approach. To remedy this problem, we propose a network Voronoi region-based technique to guarantee query accuracy and a network Voronoi envelope-based pruning heuristic to improve query performance.

Keywords Spatial databases · Maximum influential location selection problem · Optimal location selection query · Differential privacy

1 Introduction

Recently, optimal location queries (OLQs) have been of intense interest in the spatial database community. Given a set of clients and a set of facilities, an optimal location query (OLQ) finds a location such that a certain cost function is optimized when a new facility is set up. Due to the widespread use of mobile devices and social network services, it is possible to collect the location information for each user. The

✉ Seog Park
spark@sogang.ac.kr

Sehwa Park
sehwapark@sogang.ac.kr

1 Database Laboratory, Department of Computer Science and Engineering, Sogang University, Seoul, Korea

increasing amount of location data can provide great opportunities to support market analysis and facility location decision-making problems; therefore, researchers have focused on the OLQ processing technique, such as the maximum influential location selection (Max-inf) problem. Specifically, the Max-inf problem in a road network is a real-life domain problem, so researchers have intensely studied such problems. The Max-inf problem in a road network [1–3] is a traditional problem that identifies the most influential object in a given network database. When potential objects P, existing facilities F, and client locations C are given, the Max-inf determines the location that attracts the most clients based on the assumption that each client uses the nearest facility, as depicted in Fig. 1. The influence of each facility is calculated using the following equations.

$$\mathrm{MaxINF(P)} = \arg\max_{p \in P} \sum_{c \in IS(p)} w(c), \tag{1}$$

$$\mathrm{IS(p)} = \{c \in C | \forall f \in F, d(c,p) \le d(c,f)\}. \tag{2}$$

In (2), $d(,)$ is a network distance function, and 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, so we set $w(c)$ to 1 for all clients in this study. As mentioned above, the Max-inf is useful for market analysis, so it is beneficial in several real-life applications. For example, there are eight existing convenience stores $\{f_1, f_2, \dots, f_8\}$, and eight clients, $\{c_1, c_2, \dots, c_8\}$ in Fig. 1. We occasionally use the words "user," "client," and "customer" interchangeably. We assume that the query requester who is the owner of a new convenience store wants to know the best location to attract the most clients. Therefore, we aim to find the optimal location between four different potential locations, $\{p_1, p_2, p_3, p_4\}$, to establish a new facility. As shown in Fig. 1, f_1 is the nearest facility to $\{c_8\}$, f_2 is the nearest facility to $\{c_1\}$,

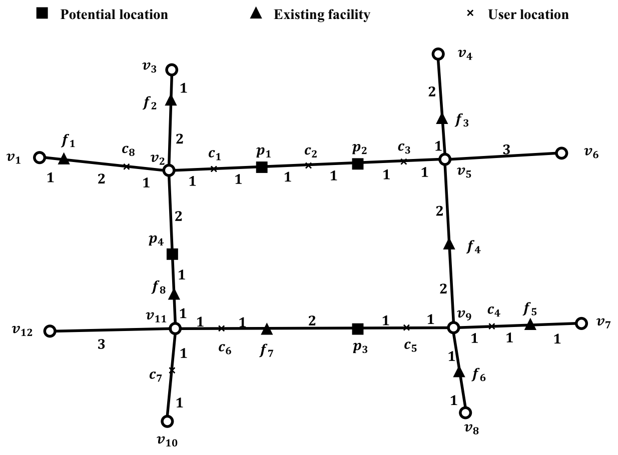


Fig. 1 Example of a Max-inf problem in a road network

and f_3 is the nearest facility to $\{c_2, c_3\}$. If a new facility is established at p_1, then it becomes the nearest facility to $\{c_1, c_2\}$. In addition, if we select location p_2, then we attract clients $\{c_2, c_3\}$. However, if we select location p_3, then only $\{c_5\}$ is interested in the new facility. Nobody would be interested in p_4. Therefore, p_1 or p_2 is the most attractive location.

To solve this example, conventional approaches use computational geometry techniques with the exact locations of customers. In general, these techniques do not consider the location privacy problem. They assume that customers trust data analysts and agree to collect their exact location. However, people do not want to disclose their exact location, and location privacy has become an emerging issue in the spatial database community. To resolve such an issue, we propose a method to apply differential privacy to the Max-inf problem, that is a de-facto standard privacy preserving technique [4]. Differential privacy [5] exploits a randomized mechanism to add controlled noise in statistical query results. A naïve approach to applying differential privacy to a Max-inf problem is to add Laplace noise to (1). We call this value a noisy count. Then, we can find the best location by computing the potential location with the maximum noisy count. However, our previous work [6] discovered that this naïve approach suffers from poor accuracy while applying differential private techniques to the Max-inf problem in Euclidean space. In this paper, we show that the same problem occurs in a road network environment.

To resolve the Max-inf problem in a differentially private manner, we first invoke reverse nearest neighbor (RNN) queries in a road network for each potential location $p_i \in P$ and identify clients whose nearest neighbor is p_i. Then, we add Laplace noise to the number of clients and select the most influential location. However, if the potential locations are close to each other, then the client location information is disclosed recursively. For instance, the RNN clients of p_1 and p_2 are $\{c_1, c_2\}$ and $\{c_1, c_2, c_3\}$, respectively, as shown in Fig. 1. The intersection of the two RNN clients is $\{c_1, c_2\}$, which indicates that their location is leaked twice by p_1 and p_2. This is because the influence regions of p_1 and p_2 overlap each other. The influence region is the network Voronoi region [7] of the corresponding potential location. The influence regions of p_1 and p_2 are described in Fig. 2. The red line is influence regions of p_1, and the blue one is for p_3. As shown in Fig. 2, these influence regions overlap each other, and clients, $\{c_1, c_2\}$ are located in the overlapping region. This problem is called sequential composition [5] in differential privacy and requires dividing the privacy budget ϵ by the number of potential locations. Thus, we should insert more noise proportional to the cardinality of P, which degrades the query accuracy exponentially due to the property of differential privacy. To remedy this issue, we propose a network Voronoi region partitioning method (NVPM) that partitions the influence regions of each potential locations to alter the problem from a sequential composition to parallel composition [8]. Although the NVPM mitigates the degradation of accuracy, it suffers from expensive computational cost. Therefore, we also propose a pruning technique called the network Voronoi envelope-based filtering method (NVEM), which precomputes the upper bound of the noisy count of the influence regions to reduce the search space.

In summary, our contributions are the following. First, we present a Max-inf problem in a road network with a differentially private user location. To the best of

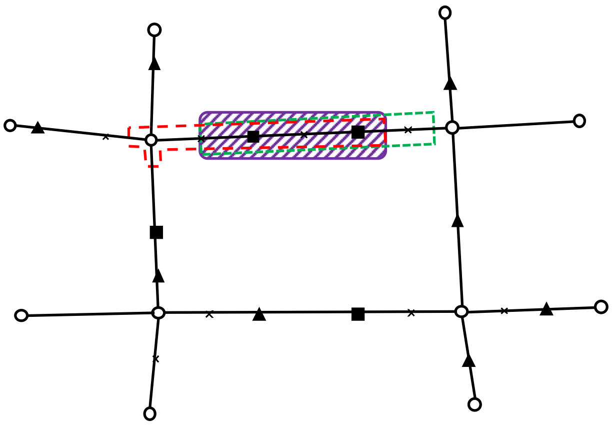


Fig. 2 Influence regions and the influence region overlapping problem

our knowledge, this study is the first attempt to apply differential privacy directly to the query processing of the Max-inf problem in a road network environment. Second, we present the influence region overlapping problem while applying differential privacy to the Max-inf problem in a road network. Third, we propose two algorithms to resolve the influence region overlapping problem, namely the NVPM and NVEM. Finally, we empirically study the properties of the proposed methods on actual road network datasets. The remainder of the paper is organized as follows. Section 2 reviews studies relevant to the Max-inf problem in a road network and differential privacy, and Sect. 3 formalizes the problem definitions and presents system models. Sections 4 and 5 propose two query-processing techniques, the NVPM and NVEM, respectively, and Sect. 6 provides the experimental results from actual datasets. Finally, Sect. 7 concludes the paper and presents directions for future works.

2 Related works and backgrounds

In this section, we briefly review previous studies of the Max-inf problem and differential privacy. The Max-inf problem is one of the OLQ, and it is closely relevant to RNN queries. Therefore, we first review RNN studies.

2.1 RNN queries

Korn et al. [9] first introduced RNN queries and proposed an r-tree-based query-processing technique to solve the RNN problem. Their approach is as follows. First, they computed the distance between each data object and its nearest neighbor, and they generated the circle with the nearest neighbor distance. Then, each point p is the RNN for query point q if its circle contains q. Yiu et al. [10] first studied the RNN queries in a road network environment. They proposed a filter and refinement approach called the eager algorithm, which is based on the network expansion technique. It expands the network from the query point q, like Dijkstra's algorithm [11]. For each node n retrieved, the eager algorithm performs a range-NN query in the range of $d(n, q)$. If the data object p is retrieved, then the expansion does not proceed further from node n. As a result, it can prune node n', which is the shortest path between q and n' passes through n. Gao et al. [12] investigated reverse top-k Boolean spatial keyword queries (RkBSK) on road networks. An RkBSK retrieves the points that have query point q as one of their top-k Boolean spatial keyword query on the road network. The authors proposed the filter and refinement-based query-processing method with a connectivity-clustered access method (CCAM) [13]. Zhao et al. [14] proposed using reverse top-k geo-social keyword queries (RkGSK) in road networks. An RkGSK does not only consider spatial and textual information, but it also takes into account social data to identify prospective customers. The authors assume that customers are influenced by their friends, and it is important as much as the proximity and the textual similarity of objects. To support the RkGSK, they introduced a hybrid index structure the GIM-tree, which consists of the location, keywords, and social information for users and objects. They also proposed pruning strategies using the GIM-tree.

2.2 Max-inf problems

Cabello et al. [15] first introduced the Max-inf problem. As mentioned, the Max-inf finds the facility, o, that has the maximum influence. The influence refers to the number of clients whose nearest neighbor facility is o. The authors proposed the query-processing algorithm based on the nearest location circle (NLC). If the facilities are established outside of the NLC of c, it cannot be an RNN of c. Therefore, the optimal solution can be obtained by finding regions enclosed by the largest number of NLCs. Wong et al. [16] introduced the first polynomial-time algorithm, called the MaxOverlap, to support the Max-inf query. They also exploited NLCs to determine the optimal area in Euclidean space. They constructed the overlap table, which is an inverted list of each client. The overlap table consists of the pointer to each client c and the information for the other clients whose NLC overlaps with the NLC of c. Then, they determined the optimal region using the influence-based pruning technique. Yan et al. [17] presented a relaxation version of the Max-inf problem. They introduced the concept, the relaxed location circle (RNLC) of client c, which is the NLC with a buffer. They designed a kd-tree-based approach called fast influential location miner (FILM), which returns a small grid cell where all locations have an influence guarantee. Contrary to previous studies, the FILM finds a nearly optimal location in much less time. Zhang et al. [18] introduced the Min-dist location selection problem, which finds a location in a given rectangular query region Q that minimizes the average nearest neighbor distance. This query can be applied to the market analysis of a franchise restaurant because it returns the position that maximizes the benefit of the customers. To solve this problem, the authors proposed a progressive algorithm that divides the query region gradually until an answer set is found. Qi et al. [19] proposed the maximum nearest facility circle distance (MND) method to resolve the Min-dist problem efficiently. The MND is a variation of the minimum bounding rectangle, which is combined with the NLC. They exploited the r-tree with MND to solve the Min-dist problem. Moreover, Xiao et al. [1] and Yao et al. [2] first presented an optimal location selection query in a road network environment. They proposed the attraction set that records clients that are closer to each vertex than to their nearest facilities. They also exploited the graph partitioning technique to prune the unnecessary road segments with an attraction set. Chen et al. [20] solved the Min-max query in a road network. The Min-max query minimizes the maximum distance from a client to the closest facility. The authors introduced the nearest location component, which is a variant of NLC in a road network. They also proposed a piece-wise linear function-based algorithm with the nearest location component. Liu et al. [21] resolved the Min-max problem using multiple new servers. It is a more generalized version of the Min-max problem that finds locations of k servers to minimize the maximum distance of clients. The authors also employed the nearest location component and proposed the client cost line (CCL). The CCL is the piece-wise linear function of distance from a point to each client, and the intersection points of CCLs are potential server points. Chen et al. [3] presented kNN-based optimal location querying (KOLQ), which is a generalized Max-inf problem. The KOLQ assumes that each client can use the kNN facilities. It is more generalized than the nearest neighbor assumption. They extended the nearest location component to the k nearest location components and proposed a pruning technique by computing the upper bound count of each edge.

2.3 Differential privacy

Differential privacy was first introduced by Dwork [5], and its purpose is masking the evidence for a single individual. Its definition and properties are as follows.

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

$$Pr[M(D_1) \in S] \le exp(\epsilon)Pr[M(D_2) \in S]. \tag{3}$$

where ϵ is the privacy budget, and $\| \cdot \|_1$ is the norm of a vector. The Laplace mechanism can achieve ϵ-differential privacy that simply adds sampled Laplace noise to the query results. In this paper, we use the Laplace mechanism, but any other method can be applied, if it is ϵ-differentially private. Differential privacy satisfies simple composition properties, which are called the sequential composition and parallel composition, as follows.

Definition 2 (*Sequential composition*) If we let M_1 and M_2 be two differential private mechanisms and their privacy budgets be ϵ_1 and ϵ_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 ϵ total privacy budget.

Definition 3 (*Parallel composition*) Let M_i provide ϵ_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 (ϵ_i)-differential privacy.

The private spatial decomposition technique is the most related to our work among existing studies on differential privacy. Cormode et al. [22] first proposed differentially private spatial decomposition methods, which exploits the dataset partitioning technique to apply differential privacy to spatial data query processing. They instantiated a hierarchical tree structure, such as the kd-tree, to decompose a geometric space from large to small areas. Then, they added Laplace noise to the count for each node. Qardaji et al. [23] introduced the adaptive grid method to minimize the error of Laplace noise. They also identified the selection of partition granularity to balance errors from the noise error and uniformity error. Li et al. [24] proposed a two-stage mechanism for answering sets of range queries under differential privacy. The mechanism first partitions the domain into approximately uniform regions and then derives a bucket count using the workload queries. Zhang et al. [25] exploited a quadtree for spatial datasets. They proposed a threshold to determine the minimum of a subdomain and another threshold to limit the height of a quadtree. The amount of noise added to the quadtree can be limited to a constant using the two thresholds.

2.4 Limitation of existing studies

Most existing research on the Max-inf directly uses the information on clients to resolve the problem efficiently. If these techniques are applied to our problem, then the client information would be leaked recursively during query processing. Therefore, it is inevitable to insert more noise that is proportional to $O(n|C|)$ where n is the number of iterations. This method leads to significant degradation of query accuracy. Meanwhile, previous studies of differential privacy have focused on the private spatial decomposition method to preserve spatial data privacy. However, private spatial decomposition is inadequate to handle the Max-inf problem due to data skewness which is called non-uniformity error. In other works [26–29] related to our problem, there are no studies have proposed a method to resolve the non-uniformity error. Generally, the facilities and users are not evenly distributed in a road network; therefore, it increases the non-uniformity error. In conclusion, existing studies are not suitable for solving the Max-inf problem in a differentially private manner; thus, we use the network Voronoi region-based approach and propose a pruning technique to remedy these problems.

3 Problem definition

We formally define the problems. Table 1 summarizes the notation frequently used in this paper. All data objects are represented by points in the road network. Thus, we define a road network first.

Table 1 Frequently used symbols

Notation	Description
v, V	A vertex and a set of vertices in the road network
e, E	An edge and a set of edges in the road network
o	A point in the road network
C, F, P	Set of clients, existing facilities, and potential locations, respectively
c, f, p	A client in C, an existing facility in F, and a potential location in P, respectively
$d(,)$	Network distance between two points
ϵ	Privacy budget
$ncount(\,)$	Noisy count
$UBncount(\,)$	Upper-bound noisy count
N_i, N_c	Node of B+-tree (CCAM index) and its child node

Definition 4 A road network is represented by graph $G = (V, E)$ where V is a set of vertices, such as terminal points or road junctions, and E is a set of edges representing the network edges where each edge connects two vertices. Each edge $e \in E$ is denoted as $e(v_i, v_j)$ where v_i and v_j are starting and ending vertices, respectively. Any object point, o, including a potential location, an existing facility, and a client is located on an edge e.

In this paper, we assume that a road network is a weighted undirected graph because the direction is not important in our problem, and it is easily extensible to a directed graph. To resolve the Max-inf problem, we should compute the influence region of each potential location; therefore, we define the influence region in a road network as follows.

Definition 5 (*Influence region in a road network*) For any object o in the road network environment, the influence region of the potential location $p \in P$ on the set $F \cup \{p\}$ is a set of points in a road network that is denoted as $V(p)$:

$$V(p) = \{o | d(p, o) \leq d(p', o), \forall p' \in F \cup \{p\} \wedge p \neq p'\}. \tag{4}$$

The influence region of p encloses all of the client in $IS(p)$, and only the clients in $IS(p)$ are affected by p. We used the influence region to quickly identify $IS(p)$ by redefining (2) as $IS(p) = \{c | c \in C \wedge c \in V(p)\}$. Finally, the Max-inf problem in a road network using differentially private user locations is defined as follows.

Definition 6 (*Max-inf in a road network with differentially private user locations (DP-Max-inf)*) Given existing facilities F and client locations C, the DP-Max-inf determines a location that maximizes the influence with differentially private user locations between potential locations P, which are query-requester defined points on a road network.

$$ncount(IS(p)) = \sum_{c \in IS(p)} w(c) + Lap(1/\epsilon'), \tag{5}$$

$$DPMaxINF(P) = \arg\max_{p \in P} ncount(IS(p)). \tag{6}$$

The DP-Max-inf simply changes the objective function from (1) to (5) by adding Laplace noise, and we denote this as a noisy count. To achieve ϵ-differential privacy, ϵ' must be $\epsilon/|P|$ because of the influence region overlapping problem. However, it suffers from significantly poor performance accuracy, so we present an enhanced version of the sequential composition. First, it generates the influence regions of each potential location. Then, it determines the overlapping pairs of influence regions. Next, it counts the number of potential locations whose influence region overlap. Finally, it calculates (5) for each influence region with overlapping potential locations. This improves the accuracy by providing tighter noise bounds than the naïve approach because not all influential regions of the potential location overlap. However, it still suffers from a decrease in query accuracy as the number of potential locations increases. To remedy this problem, we propose a method that partitions an influence region with overlapping conditions.

4 Network Voronoi region-partitioning method

As mentioned in the last paragraph of the previous section, the sequential composition approach suffers from poor potential location scalability on the query accuracy. Therefore, we propose the NVPM to further improve accuracy.

4.1 Concept of the network Voronoi region-partitioning method

The NVPM exploits a parallel composition to mitigate the degradation of accuracy rather than using sequential composition. First, the NVPM constructs an influence region for each potential location. Next, it partitions each influence region using combinations of overlapping regions. Then, it computes the noisy count for each partitioned region and determines the nearly optimal location. As shown in Fig. 3, the influence regions of potential locations are partitioned using combinations of overlapping regions. The red line represents the influence region of p_1, whereas the blue and green lines denote the influence regions of p_2 and p_3, respectively. These influence regions overlap each other, as depicted in Fig. 3. Therefore, the NVPM partitions them, so Region A consists only of the sub-region of $V(p_3)$, which does not overlap with other influence regions. In contrasts, Region B is the overlapping region of $V(p_1)$ and $V(p_3)$. In addition, Region C is the overlapping region of $V(p_1)$, $V(p_2)$, and $V(p_3)$. After partitioning the influence regions, the NVPM computes the noisy count for each partitioned region. For example, we can calculate the noisy count of Region A as $ncount(\{o|o \in IS(p_1) \wedge o \notin IS(p_2) \wedge o \notin IS(p_3)\})$. Then, we can compute the total noisy count of $IS(p_1)$ by summing the noisy counts of Regions A, B, and C.

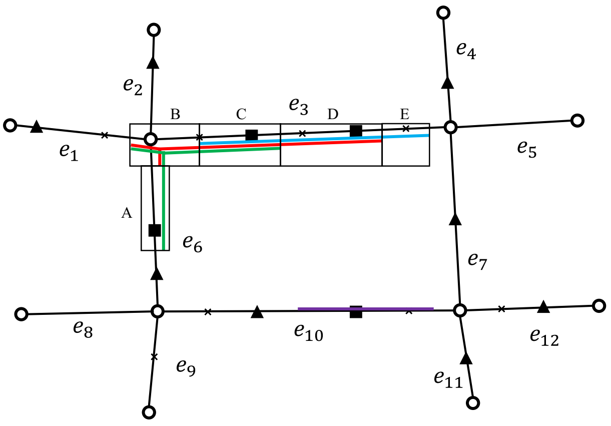


Fig. 3 The partitioned regions of influence regions

The simplest way to apply the NVPM directly is to follow the aforementioned steps. The shortest path tree (SPT) algorithm can be used to construct influence regions. The first step is to build an SPT from each vertex by expanding the edges of a road network until it meets the nearest existing facility. Then, each vertex v is associated with its nearest existing facility and nearest neighbor distance. Afterward, the influence region is generated by the construction of the SPT from a given potential location p. Let an expended edge be $e(v, v')$. Assume that v is closer to p than v' and that the shortest paths from p to v are included in the influence region of p. Then, if $d(p, v)$ is less than the nearest neighbor distance of v, $e(v, v')$ is included in the influence region. Otherwise, a boundary point on e is computed which is between two end vertices, v and v'. Finally, the influence region of p is a set of road segments consisting of vertices and boundary points. Next, the combinations of overlapping regions must be computed to partition the influence regions. It takes exponential time complexity to compute the combinations of overlapping regions in a naïve way. In the last step, the noisy count is calculated using traversing for each partitioned regions. This naïve approach is inefficient because it traverses the road network repeatedly. Therefore, we introduce a method to construct an influence region, partitioned region, and noisy count simultaneously.

4.2 Query processing for network Voronoi region partitioning

In real-life applications, the size of a road network is too large to process in the main memory, so we designed a disk-based storage model to support the algorithms. We exploited the CCAM [13] index, which is the most popular index structure for query processing in a road network environment. It consists of an adjacency file and an inverted file for points in the road network, as depicted in Fig. 4. The CCAM collects the points of interests (POIs) that are on the same edge to store in one group. For every group, it maintains the information of the edge where the POIs are located and the number of POIs. In our setting, we stored the existing facilities and client locations as POIs with their labels. We computed the distance from the vertex using

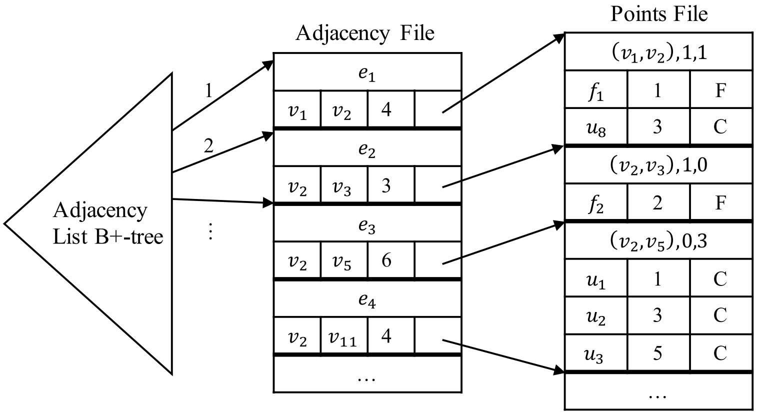


Fig. 4 An example of CCAM index structure

smaller ID for each POI. For example, an existing facility f_1 is located on $e(v_1, v_2)$, so the distance is calculated from v_1 as 1. A B+-tree is employed to facilitate efficient access to adjacency files, where the vertices are sorted by the identifiers.

As previously explained, the NVPM consists of the influence region construction, partitioned region generation, and computation of the noisy count. These steps are based on the network expansion approach, so we can reduce computation time by performing them concurrently during one traversal. Algorithm 1 provides the overall query-processing steps of our proposed NVPM using CCAM index. We first define the data structures used in Algorithm 1. The overlapped Voronoi segment is for the overlapping region of influence regions and is defined as follows.

Definition 7 An overlapped Voronoi segment (OVS) is a sub-edge that contains the following attributes.

- eid: the identifier of the corresponding edge,
- plist: the list of corresponding potential locations,
- start: the starting point of OVS on the edge,
- end: the ending point of OVS on the edge, and
- clist: the list of customers located on the OVS, where each customer contains identification and the location.

Without loss of generality, we assume that vertices are ordered by their identifiers. Therefore, the following condition holds if the OVS is on $e(v_i, v_j)$ where $i < j$.

$$0 \leq start \leq end \leq d(v_i, v_j) \tag{7}$$

Moreover, the influence region is represented by a set of triplets (*eid*, *start*, *end*), where the attributes are the same as those of the OVS. While traversing a new edge e_i, the NVPM checks the OVS of e_i. If the OVS of e_i does not exist, it generates a new OVS of e_i with the corresponding potential location, *start*, *end*, and clients.

Otherwise, it finds the overlapping regions with the existing *start* and *end* points. In addition, it partitions the OVS with a new segment. For example, the influence region of p_1 is $\{(e_1, 2.5, 3), (e_3, 0, 4.5), (e_{12}, 0, 0.5)\}$ in Fig. 3. Then, the OVS of e_1 is $(e_1, \{p_1\}, 2.5, 3, \{\})$, the OVS of e_3 is $(e_3, \{p_1\}, 0, 4.5, \{c_1, c_2\})$, and the OVS of e_{12} is $(e_{12}, \{p_1\}, 0, 0.5, \{\})$. Next, the NVPM constructs the influence region of p_2, which is $\{(e_3, 1, 5.5)\}$. Because the OVS of e_3 already exists, the NVPM partitions it as $(e_3, \{p_1\}, 0, 1, \{c_1\})$, $(e_3, \{p_1, p_2\}, 1, 4.5, \{c_2\})$ and $(e_3, \{p_2\}, 4.5, 5.5, \{c_3\})$ (assuming that the ties are broken). Then, the final OVSs are generated and listed in Table 2. After generating the influence regions for all potential locations, we can obtain the noisy count of each partitioned region by adding the Laplace noise to the number of clients in the OVS that has the same *plist*. For example, ov_1, ov_2, and ov_6 have the same *plist*, $\{p_1, p_3\}$; therefore, they are merged, and the noisy count is $1 + Lap(1/\epsilon)$.

Algorithm 1 Network Voronoi Region Partitioning (NVPM)

```
Input: $P$ - a set of potential locations, $I$ - a CCAM index structure, $\epsilon$ - a privacy budget
Output: $\tilde{p}$ - a nearly optimal location
 1: $\tilde{p} \leftarrow \emptyset$;
 2: $\tilde{w} \leftarrow -\infty$;
 3: $OVS \leftarrow \emptyset$ ;
 4: for $p \in P$ do
 5:     $IR_p \leftarrow$ Generate the influence region of $p$;
 6:     for $e \in IR_p$ do
 7:         if $e.eid$ is in $OVS$ then
 8:             Partition $OVS.get(e.eid)$;
 9:         else
10:             $plist \leftarrow$ find corresponding potential locations;
11:             $clist \leftarrow$ find corresponding clients;
12:             $OVS.put(e.eid, e.start, e.end, plist, clist)$;
13: Merge segments of $OVS$ that have the same $plist$;
14: for $s \in OVS$ do
15:     $\tilde{n} \leftarrow len(s.clist) + Laplace(\frac{1}{\epsilon})$;
16:     for $p \in s.plist$ do
17:         $ncount(p) \leftarrow ncount(p) + \tilde{n}$;
18:         if $ncount(p) > \tilde{w}$ then
19:             $\tilde{w} \leftarrow ncount(p)$
20:             $\tilde{p} \leftarrow p$
21: Return $\tilde{p}$
```

Table 2 The overlapped Voronoi segment list example

eid	plist	start	end	clist
e_1	$\{p_1, p_3\}$	2.5	3	$\{\}$
e_3	$\{p_1, p_3\}$	0	1	$\{c_1\}$
	$\{p_1, p_2, p_3\}$	1	2.5	$\{\}$
	$\{p_1, p_2\}$	2.5	4.5	$\{c_2\}$
	$\{p_2\}$	4.5	5.5	$\{c_3\}$
e_{12}	$\{p_1, p_3\}$	0	0.5	$\{\}$
	$\{p_3\}$	0.5	2.5	$\{\}$

5 Network Voronoi envelope-based filtering

In this section, we propose an NVEM. The NVPM suffers of long query-processing time because it does not use any pruning strategies. Even if no overlapping regions, the NVPM requires $O(|P||V|log|V|)$ time complexity to generate the influence regions of potential locations where $|V|log|V|$ is for the SPT construction, and $|P|$ is for the iteration of potential locations. Thus, the time complexity of the NVPM increases proportionally to the cardinality of the potential locations. However, we observe that customers and facilities are generally skewed in real-world problems. Thus, most potential locations are much less influential than the optimal location. Therefore, we can reduce query-processing time if we know the upper-bound noisy count of the potential location in advance. Motivated by this observation, we propose a network Voronoi envelope to compute the upper-bound noisy count and a pruning technique to filter out unnecessary potential locations.

5.1 Network Voronoi envelope

To compute the upper-bound noisy count, we first find the upper-bound region of each influence region. We define this region as the network Voronoi envelope.

Definition 8 A network Voronoi envelope (NVE) is a partitioned sub-graph of G, such that the terminal vertices are existing facilities.

Figure 5 illustrates the NVEs of the previous example. As in the definition of NVE, the road network is partitioned by eight cells (a to g) where each cell has the existing facilities as terminal vertices. Then, the potential locations correspond to each cell where they are located. For example, p_1, p_2, and p_3 are in cell d, and only p_4 is in cell f. Thus, cell d is an NVE of p_1, p_2 and p_3, and cell f is an NVE of p_4. Let $NVE(p_i)$ be a corresponding NVE of the potential location p_i. Then, the influence region of p_i is inside of the $NVE(p_i)$, because the network expansion of p_i stops when it meets the terminal facilities of $NVE(p_i)$. Then, the upper-bound noisy count of p_i, which is denoted $UBncount(p_i)$, is the number of clients in the $NVE(p_i)$ with Laplace noise. To explain this easily, we present the example of the upper-bound noisy count without noise. There are four clients, $\{c_1, c_2, c_3, c_8\}$ in cell d, whereas only two clients, $\{c_4, c_5\}$ exist in cell f. The $UBncount(p_4)$ is 2, whereas $UBncount(p_1)$, $UBncount(p_2)$, and $UBncount(p_3)$ are 4. The upper-bound noisy count is used to prune unnecessary potential locations during query processing. Because the NVEM uses the information of clients twice, we divide ϵ into ϵ_1 and ϵ_2. In addition, ϵ_1 is used to compute the upper-bound noisy count, and ϵ_2 is used for query processing the NVPM. Then, the overall query processing of NVEM is $(\epsilon_1 + \epsilon_2)$ - differentially private.

5.2 CCAM index with network Voronoi envelope

As mentioned, the NVEM can filter out potential locations whose upper-bound noisy count is less than the current best location during query processing. Thus, we modified the B+-tree of CCAM to hold identifiers of edges instead of vertices and to store aggregation values of upper-bound noisy counts as depicted in Fig. 6. The leaf nodes of the B+-tree are composed of pointers to edges and the upper-bound noisy count for each edge. Let *NVE*(*e*) be a set of NVEs that intersect with *e*. Then, the upper-bound noisy count of *e* is the maximum value of *NVE*(*e*).

$$UBncount(e) = \max_{NVE \in NVE(e)} UBncount(NVE) \tag{8}$$

The intermediate node stores the maximum upper-bound noisy count of the children nodes. Let *N* be the node of B+-tree and *N.children* be its children nodes. Then, the upper-bound of the noisy count of *N* is calculated as follows:

(i) *N* is an intermediate node (including the root node):

$$UBncount(N) = \max_{e \in N.children} UBncount(e) \tag{9}$$

(ii) Otherwise:

$$UBncount(N) = \max_{N' \in N.children} UBncount(N') \tag{10}$$

The intermediate nodes of the B+-tree only use the noisy count of NVEs that are partitioned cells of the road network. Therefore, it does not require an additional privacy budget due to the post-processing property of differential privacy [5]. In conclusion, each node of our modified B+-tree has the following attributes:

- keys: the separation values that divide its sub-trees,
- UBncount: the upper-bound noisy count, and

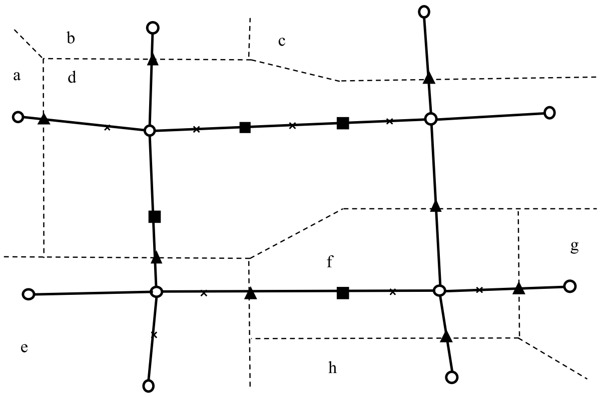


Fig. 5 Network Voronoi envelopes example

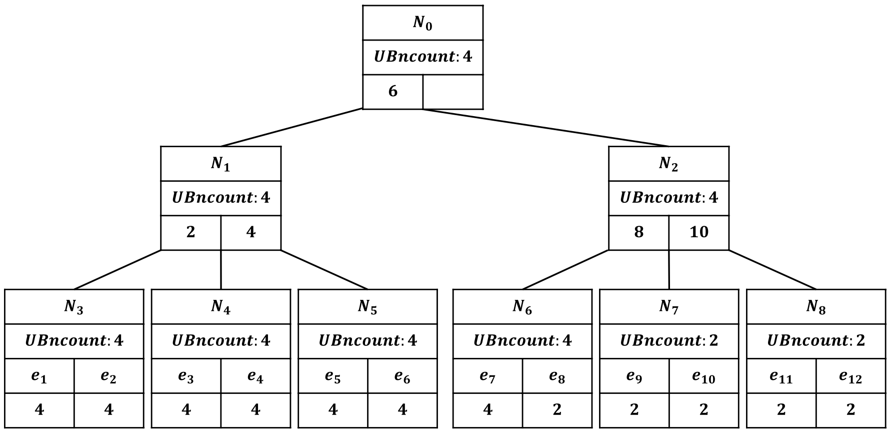


Fig. 6 A B+tree with NVEs

– children: the list of pointers to children nodes.

5.3 Query processing of the network Voronoi envelope-based filtering

The NVEM traverses the B+-tree in the best-first search approach to find the corresponding edge of potential locations. First, the NVEM groups the potential locations on the same edge. Then, the NVEM traverses from the root node of the B+-tree and classifies potential locations according to their corresponding edge IDs while visiting each node. During traversing B+-tree, NVEM prunes out unnecessary potential locations whose upper-bound noisy cost is greater than the current best cost. Algorithm 2 presents the overall query-processing steps of the NVEM. The NVEM maintains the triplet $(N_i, UBncount(N_i), plist(N_i))$ in a maxheap sorted by $UBncount(N_i)$, where N_i is the node of B+-tree and $plist(N_i)$ is a set of corresponding potential locations. Each p in $plist(N_i)$ has the edge id where p is located and the flag whether p is already checked or not. Then, the NVEM dequeues the triplet $(N_i, UBncount(N_i), plist(N_i))$ of the maxheap at each step and performs the appropriate tasks according to the following cases.

– N_i is a leaf node of the B+-tree:

1. Filter out any $p \in plist(N_i)$ already checked.
2. For each $p \in plist(N_i)$, compute the $NVE(p)$ and find the potential locations $P_{NVE(p)}$ inside of $NVE(p)$.
3. Invoke NVPM with $P_{NVE(p)}$ for each $p \in plist(N_i)$.

– N_i is an intermediate node of the B+-tree:

1. Partition $plist(N_i)$ according to the key of child node N_c.

2. Enqueue $(N_c, \mathit{UBncount}(N_c), \mathit{plist}(N_c))$ to the max-heap for each child node N_c.

We explain the algorithm with previous example. First, the NVEM enqueues $(N_0, 4, \{p_1, p_2, p_3, p_4\})$ to the maxheap. Then, the NVEM dequeues $(N_0, 4, \{p_1, p_2, p_3, p_4\})$ at the next iteration. The NVEM partitions $\{p_1, p_2, p_3, p_4\}$ according to their locations. As shown in Fig. 6, p_1 and p_2 are on $e3$, p_4 is on $e4$, and p_3 is on $e10$. Further, $e3$, $e4$ are on the sub-trees of N_1, and $e10$ is a descendent of N_2. Therefore, the NVEM partitions the group of potential locations as $\{p_1, p_2, p_4\}$ and $\{p_3\}$. Next, it enqueues $(N_1, 4, \{p_1, p_2, p_4\})$ and $(N_2, 4, \{p_3\})$ to the maxheap. Again, the NVEM dequeues $(N_1, 4, \{p_1, p_2, p_4\})$ from the maxheap. The NVEM does not partition the $\{p_1, p_2, p_4\}$ because they are on the same node, N_4. Therefore, the NVEM enqueues $(N_4, 4, \{p_1, p_2, p_4\})$. Next, the NVEM dequeues $(N_2, 4, \{p_3\})$, and the NVEM enqueues $(N_7, 2, \{p_3\})$ to the maxheap. Then, $(N_4, 4, \{p_1, p_2, p_4\})$ pops out, and the potential locations, $\{p_1, p_2, p_4\}$, correspond to the same *NVE*. Therefore, it invokes the NVPM with $\{p_1, p_2, p_4\}$ and computes the noisy count of each potential location. The noisy count of p_2 is 3, and it is the current best location. Finally, the upper-bound noisy count of $(N_7, 2, \{p_3\})$ is less than the current best noisy count; therefore, the algorithm terminates and returns p_2 as the nearly optimal location.

Algorithm 2 Network Voronoi Envelope-based Filtering (NVEM)

```
Input: $P$ - a set of potential locations, $I$ - a modified CCAM index structure, $\epsilon$ - a privacy
    budget, $\alpha$ - $\epsilon$ ratio
Output: $\tilde{p}$ - a nearly optimal location
 1: $\tilde{p} \leftarrow \emptyset$
 2: $\tilde{w} \leftarrow -\infty$
 3: $Q \leftarrow$ priority queue;
 4: Group the potential locations on the same edge;
 5: while $\tilde{w} < Q.top.UBncount$ do
 6:     $N \leftarrow Q.dequeue()$
 7:     if $N$ is an edge then
 8:         for $p \in N.plist$ do
 9:             Check whether $p$ is already processed;
10:             Compute the $NVE(p)$ and $P_{NVE(p)}$;
11:             $\bar{p} \leftarrow$ Do $NVPM(P_{NVE(p)}, I, \epsilon \cdot (1-\alpha))$;
12:             if $ncount(\bar{p}) > \tilde{w}$ then
13:                 $\tilde{w} \leftarrow ncount(\bar{p})$;
14:                 $\tilde{p} \leftarrow \bar{p}$;
15:     else
16:         for $N_c \in N.children$ do
17:             Partition $N.plist$ according to $N_c.key$;
18:             $Q.enqueue(N_c, UBncount(N_c), plist(N_c))$;
19: Return $\tilde{p}$
```

6 Experimental results

In this section, we report the experimental results of the proposed method. Section 6.2 studies the scalability of proposed methods with varying dataset sizes. Section 6.3 evaluates the performance of the methods with varying parameters. Section 6.4 reveals the index build time and maintenance cost.

6.1 Experimental environments

We evaluate our proposed methods, NVPM and NVEM using two real road networks, namely CAL [30] and NA [31]. No dataset was proper for our problem in road network settings, so we generated synthetic datasets based on CAL and NA. To evaluate the performance with generality, we generated 10 sub-datasets of each road network, and the experiments are conducted 10 times with varying parameters. The results of existing facility scalability are excluded because they had the opposite effect from those of potential locations. The CAL dataset is an actual road network dataset of California that contains 21,048 vertices, 21,693 edges, and 85,070 POIs. We divided the POIs into potential locations and existing facilities. Next, we randomly generated client locations based on the Zipfian distribution, varying the skewness of the data. We sampled the existing facility from the *zipf* distribution and computed the distance between it and nearest vertex. Then, we generated the locations of each user from the *zipf* distribution using the calculated distance. In contrasts, the NA dataset is an actual road network dataset for North America that contains 175,813 vertices and 179,179 edges. It has no POI information, so we generated the POIs from a uniform distribution. The dataset configuration is the same as that of the CAL dataset. Table 3 summarizes the experimental setup in which the bold text indicates the default settings. The experiments were conducted using an Ubuntu 14.04 operating system with an Intel(R) Xeon(R) CPU E5-2620 v2@2.10 GHz and 64 GB RAM. We compared the enhanced version of sequential composition method (SCM) we explained in Sect. 3, the private spatial decomposition method (PSD), and proposed methods (NVPM and NVEM). The PSD is based on PrivSem [32], which is the most recent private spatial decomposition method in a road network environment. To determine the nearly optimal location with PSD index, we constructed each influence region of the potential locations. Then, we found the overlapping areas of the influence region and PSD index. Next, we computed the noisy count of each potential location with the overlapping ratio. Finally, we determined the potential location using the highest noisy count. We measured mean absolute error (MAE), query accuracy, and query-processing time for the optimal location selection. If we let $p_o(i)$ be an actual optimal location and $p_m(i)$ be the query output of each method in the i_{th} iteration, then, the query error indicates the difference between the number of clients attracted by $p_o(i)$ and $p_m(i)$. The query accuracy is the number of times that $p_m(i)$ is the same as $p_o(i)$. With the number of test cases, n_t, the MAE and query accuracy are calculated as follows.

$$MAE = \sum_{i=1}^{n_t} \frac{|IS(p_o(i))| - |IS(p_m(i))|}{n_t}, \tag{11}$$

$$ACC = \sum_{i=1}^{n_t} \frac{I(p_m(i))}{n_t}, I(x) = \begin{cases} 0 & x \neq p_o(i) \\ 1 & x = p_o(i) \end{cases}. \tag{12}$$

6.2 Scalability test

Figure 7 illustrates the results of the potential location scalability. We evaluated the accuracy, MAE, and query time by varying the number of potential locations. As depicted in the figures, all methods degrade the MAE and accuracy as the number of potential locations increases. However, the NVPM outperforms other methods in terms of MAE and accuracy. The accuracy of the NVEM is slightly worse than the NVPM, but the query-processing time is significantly faster than that of the others. The accuracy of the SCM is better than NVEM when the number of potential locations is small. However, it becomes worse than NVEM as the number of potential locations increases because the influence region overlapping problem becomes worse as $|P|$ increases. The results of client scalability are described in Fig. 8, showing that the accuracy of all methods increases as the number of users increases. In contrast, the MAE decreases as the number of users increases because that the effect of Laplace noise on the query results decreases as the number of clients increases. The query-processing time increases in all methods as the number of users increases, except for the NVEM. Because the clients are skewed in a road network, the pruning probability increases as the number of clients increases. Moreover, the PSD is less affected than other methods in the query-processing time. The PSD just computes the overlapping areas of each influence region and the PrivSem index; thus, the query-processing time is independent of the number of clients.

Table 3 Experimental settings

Parameters	CAL	NA
$\|P\|$	8K, 16K, **24K**, 32K, 40K	25K, 50K, **75K**, 100K, 125K
$\|F\|$	**43,817**	**126,855**
$\|C\|$	10K **0.1M**, 1M	10K **0.1M**, 1M
N (Zipfian)	**100**	**100**
θ (Zipfian)	0.1, 0.3, **0.5**, 0.7, 0.9	0.1, 0.3, **0.5**, 0.7, 0.9
ϵ ratio (α)	0.1, 0.3, **0.5**, 0.7, 0.9	0.1, 0.3, **0.5**, 0.7, 0.9
Total ϵ	0.25, 0.5, **1.0**, 2.0, 4.0	0.25, 0.5, **1.0**, 2.0, 4.0

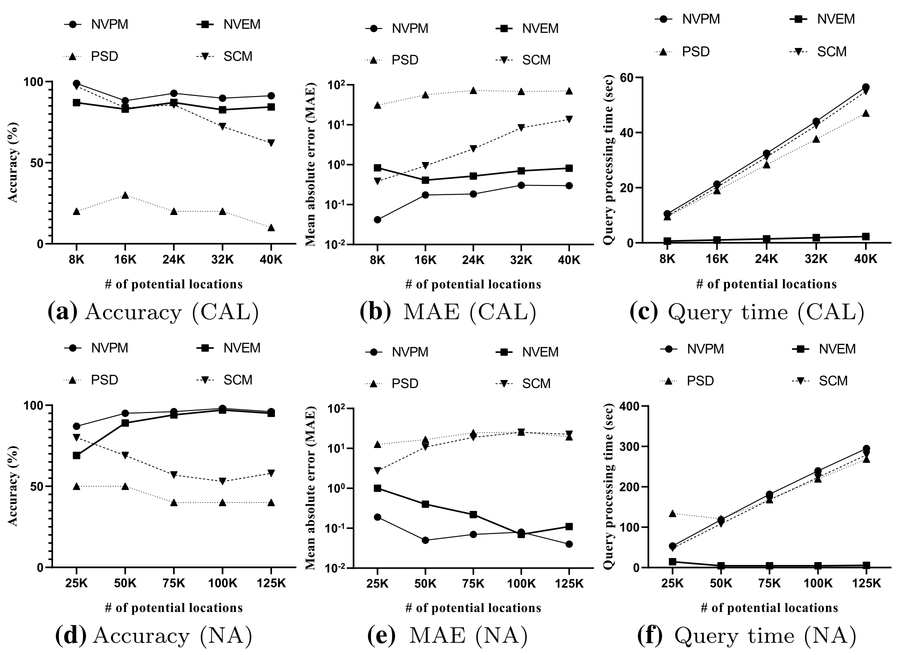


Fig. 7 The scalability of potential locations

6.3 Parameter test

The second set of experiments demonstrates the performance evaluation using varying parameters. We evaluated the effectiveness of the total privacy budget ϵ, ϵ ratio, and skewness parameter θ. As explained in Sect. 5, we divided the total privacy budget ϵ into ϵ_1 and ϵ_2. The ϵ ratio (α) indicates the budget amount used to construct the index. We evaluated the accuracy and MAE by varying the total privacy budget ϵ from 0.25 to 4. The experimental results are indicated in Fig. 9. As expected, the accuracy increases in the overall methods as ϵ increases, except for the PSD. As mentioned in Sect. 2, the PSD suffers from the uniformity error; thus, its accuracy does not improve as ϵ increases. In general, the NVPM outperforms the other methods, and the NVEM performs second best in terms of accuracy and MAE. However, the query-processing time of NVEM is the best, and it is not affected by the privacy budget. Figure 10 displays the results of the experiments that vary the ϵ ratio, α. They demonstrate that the accuracy and MAE of the NVEM decrease as α increases, and they are the best when α is 0.3 in CAL and 0.1 in NA. The noise added in the index construction and query processing is the same at $\alpha = 0.3$ in CAL and $\alpha = 0.1$ in NA. Moreover, the NVEM is second best in terms of MAE and accuracy except when the α is 0.9. In addition, α does not affect the query processing time of the NVEM. Figure 11 displays the results of the experiments at varying client skewness, θ. As θ increases, the clients become more skewed. Every scheme does not have good accuracy when the

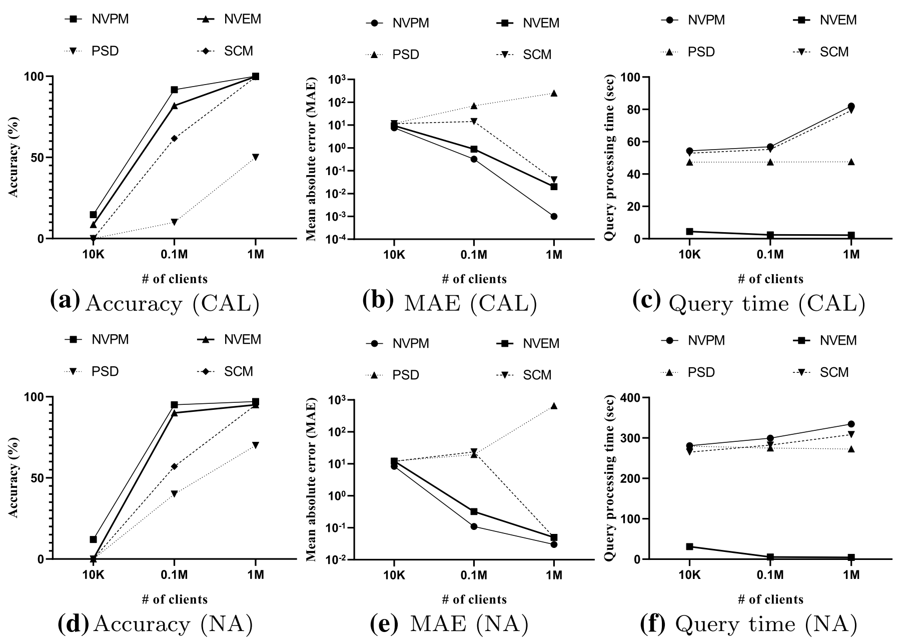
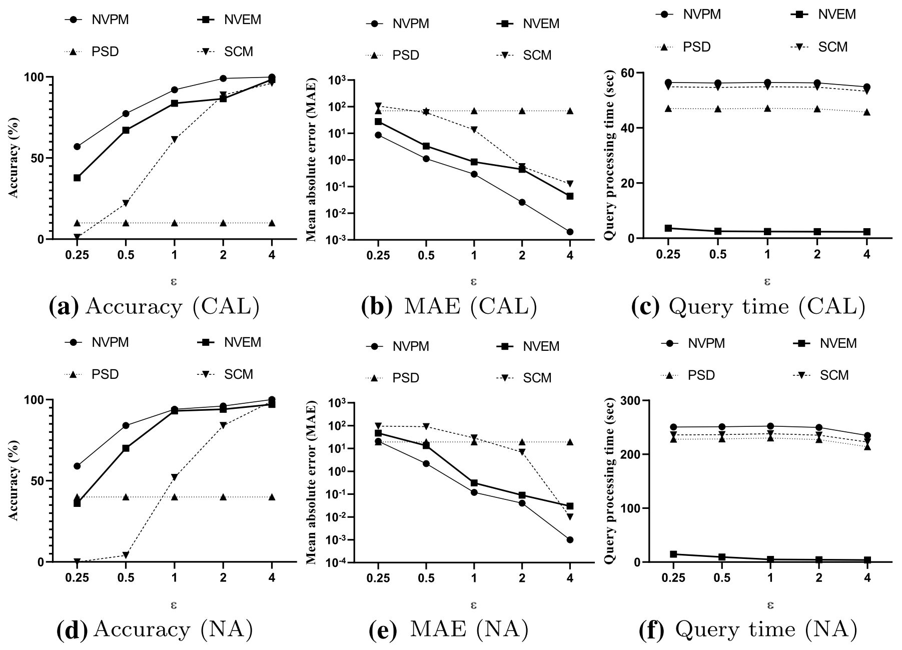


Fig. 8 The scalability of clients

clients are uniformly distributed, but the schemes become better as θ increases. The query-processing time is not affected by θ for most methods, but the NVEM becomes slightly faster as θ increases.

6.4 Index build time and storage

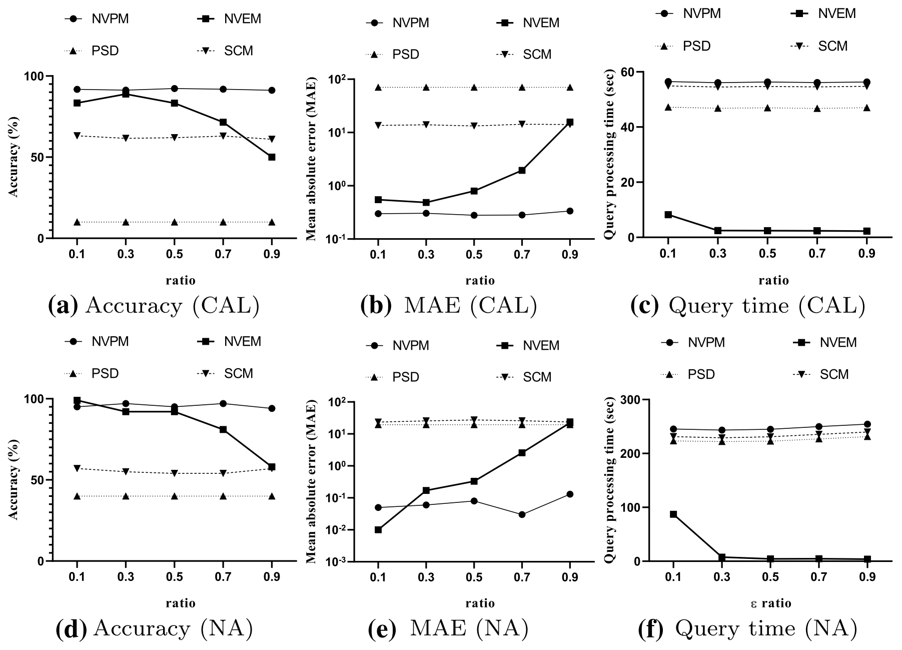
Figure 12 describes the results of the last experiments for the index build time and storage. We compared the index build time of the NVEM with the CCAM index structures. Because the scalability of potential locations does not affect the build time of the NVEM, we conducted the experiments for the clients. The NVEM exhibits inferior performance in all datasets. However, the index build time is not on-demand service time, but precomputation time. In addition, the index size of NVEM is larger than that of CCAM, but the storage of NVEM is almost the same as the number of clients increases.



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

(d) Accuracy (NA) (e) MAE (NA) (f) Query time (NA)

Fig. 9 The effect of ϵ



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

(d) Accuracy (NA) (e) MAE (NA) (f) Query time (NA)

Fig. 10 The effect of ϵ ratio

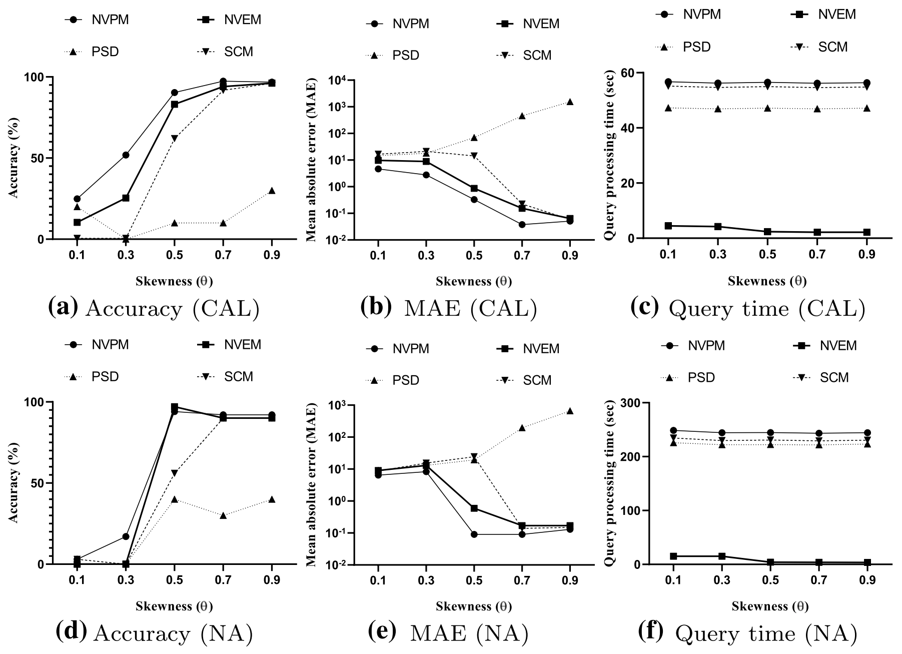


Fig. 11 The effect of θ

7 Conclusion and future works

In this paper, we presented the influence region overlapping problem while applying differential privacy to the Max-inf problem in a road network. To resolve the problem, it is inevitable to insert more noise due to the sequential composition of differential privacy. However, this naïve method suffers from poor query accuracy. Therefore, we proposed a novel method called the network Voronoi partition method to process the Max-inf query with differentially private user locations in a road network. We also proposed a Voronoi envelope-based filtering heuristic to significantly reduce the query-processing time with a slight degradation in accuracy. We conducted experiments with the real road network datasets and demonstrated that the NVEM is the best method in terms of accuracy and query-processing time. Compared to the NVMP, the NVEM had a less than 10% degradation of the accuracy, while query-processing time is reduced more than 90%. In conclusion, our proposed methods are applicable for determining an adequate location in a differentially private manner, such as the analysis of trade areas and establishment of public facilities, while preserving the individual locations of users. In the future, we aim to optimize the parameter of the ϵ ratio and extend the proposed methods to other OLQs.

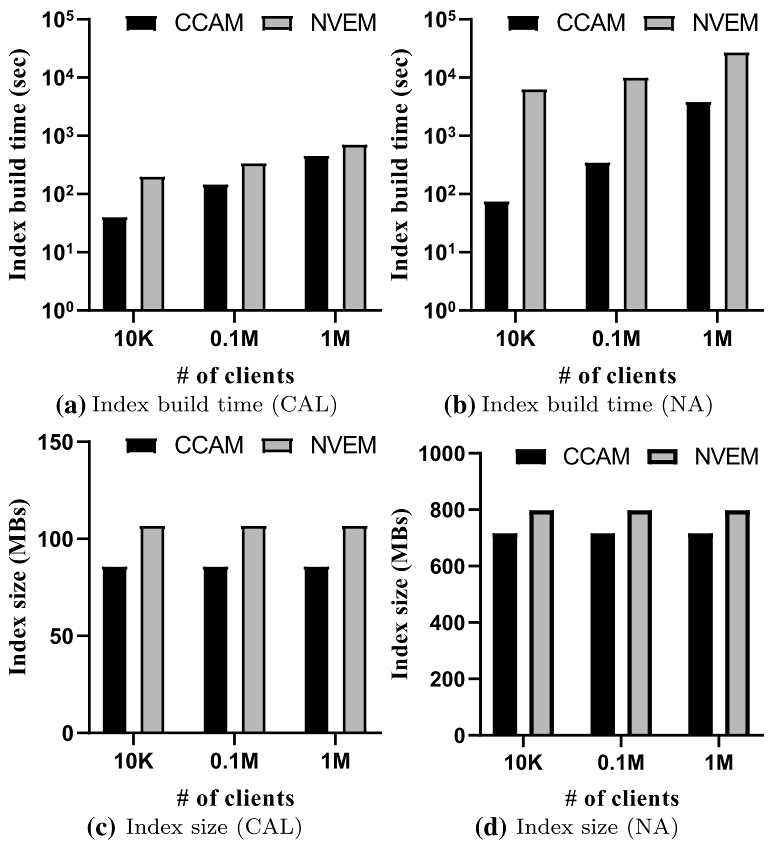


Fig. 12 Index build time and storage

Acknowledgements This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2022R1H1A2005355).

References

1. Xiao X, Yao B, Li F (2011) Optimal location queries in road network databases. In: 2011 IEEE 27th International Conference on Data Engineering, pp 804–815

2. Yao B, Xiao X, Li F, Wu Y (2014) Dynamic monitoring of optimal locations in road network databases. VLDB J 23(5):697–720
3. Chen Z, Liu Y, Fu AW, Wong RC, Dai G (2019) Kolq in a road network. In: 2019 20th IEEE International Conference on Mobile Data Management (MDM), pp 81–90
4. Li N, Lyu M, Su D, Yang W(2016) https://doi.org/10.2200/S00735ED1V01Y201609SPT018
5. Dwork C, Roth A (2014) The algorithmic foundations of differential privacy. Found Trends Theor Comput Sci 9(3–4):211–407. https://doi.org/10.1561/0400000042
6. Park S, Lee J, Park S (2020) Maximum influential location selection with differentially private user locations. IEEE Access 8:83728–83744. https://doi.org/10.1109/ACCESS.2020.2990706
7. Okabe A, Satoh T, Furuta T, Suzuki A, Okano K (2008) Generalized network Voronoi diagrams: Concepts, computational methods, and applications. Int J Geogr Inf Sci 22(9):965–994. https://doi.org/10.1080/13658810701587891
8. McSherry FD (2009) Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, SIGMOD '09, Association for Computing Machinery, New York, NY, USA, pp 19-30. https://doi.org/10.1145/1559845.1559850
9. Korn F, Muthukrishnan S (2000) Influence sets based on reverse nearest neighbor queries. SIGMOD Rec 29(2):201–212. https://doi.org/10.1145/335191.335415
10. Yiu ML, Papadias D, Mamoulis N, Tao Y (2006) Reverse nearest neighbors in large graphs. IEEE Trans Knowl Data Eng 18(4):540–553
11. Dijkstra EW (1959) A note on two problems in connexion with graphs. Numer Math 1(1):269–271. https://doi.org/10.1007/BF01386390
12. Gao Y, Qin X, Zheng B, Chen G (2015) Efficient reverse top-k Boolean spatial keyword queries on road networks. IEEE Trans Knowl Data Eng 27(5):1205–1218
13. Shekhar S, Liu D-R (1997) Ccam: a connectivity-clustered access method for networks and network computations. IEEE Trans Knowl Data Eng 9(1):102–119. https://doi.org/10.1109/69.567054
14. Zhao J, Gao Y, Chen G, Jensen CS, Chen R, Cai D (2017) Reverse top-k geo-social keyword queries in road networks. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp 387–398. https://doi.org/10.1109/ICDE.2017.97
15. Cabello S, Diaz-Banez JM, Langerman S, Seara C, Ventura I (2006) Reverse facility location problems. Department of Mathematics, University of Ljubljana, Ljubljana
16. Wong RC-W, Özsu MT, Yu PS, Fu AW-C, Liu L (2009) Efficient method for maximizing bichromatic reverse nearest neighbor. Proc VLDB Endow 2(1):1126–1137. https://doi.org/10.14778/1687627.1687754
17. Yan D, Wong RC-W, Ng W (2011) Efficient methods for finding influential locations with adaptive grids. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11, Association for Computing Machinery, New York, NY, USA, pp 1475–1484. https://doi.org/10.1145/2063576.2063788
18. Zhang D, Du Y, Xia T, Tao Y (2006) Progressive computation of the min-dist optimal-location query. In: Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB '06, VLDB Endowment, pp 643–654
19. Qi J, Zhang R, Kulik L, Lin D, Xue Y (2012) The min-dist location selection query. In: 2012 IEEE 28th International Conference on Data Engineering, pp 366–377
20. Chen Z, Liu Y, Wong RC-W, Xiong J, Mai G, Long C (2014) Efficient algorithms for optimal location queries in road networks. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD '14, Association for Computing Machinery, New York, NY, USA, pp 123–134. https://doi.org/10.1145/2588555.2612172
21. Liu R, Fu AW-C, Chen Z, Huang S, Liu Y (2016) Finding multiple new optimal locations in a road network. In: Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPACIAL '16, Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2996913.2996927
22. Cormode G, Procopiuc C, Srivastava D, Shen E, Yu T (2012) Differentially private spatial decompositions. In: 2012 IEEE 28th International Conference on Data Engineering, pp 20–31
23. Qardaji W, Yang W, Li N (2013) Differentially private grids for geospatial data. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp 757–768
24. Li C, Hay M, Miklau G, Wang Y (2014) A data- and workload-aware algorithm for range queries under differential privacy. Proc VLDB Endow 7(5):341–352. https://doi.org/10.14778/2732269.2732271

25. Zhang J, Xiao X, Xie X (2016) Privtree: a differentially private algorithm for hierarchical decompositions. In: Proceedings of the 2016 International Conference on Management of Data, SIGMOD ’16, Association for Computing Machinery, New York, NY, USA, pp 155–170. https://doi.org/10.1145/2882903.2882928
26. Wei J, Lin Y, Yao X, Zhang J (2019) Differential privacy-based location protection in spatial crowdsourcing. IEEE Trans Serv Comput 2:1. https://doi.org/10.1109/TSC.2019.2920643
27. Yilmaz E, Ferhatosmanoglu H, Ayday E, Aksoy RC (2019) Privacy-preserving aggregate queries for optimal location selection. IEEE Trans Depend Secure Comput 16(2):329–343. https://doi.org/10.1109/TDSC.2017.2693986
28. Yang X, Gao L, Zheng J, Wei W (2020) Location privacy preservation mechanism for location-based service with incomplete location data. IEEE Access 8:95843–95854. https://doi.org/10.1109/ACCESS.2020.2995504
29. Gu X, Li M, Cao Y, Xiong L (2019) Supporting both range queries and frequency estimation with local differential privacy. In: IEEE Conference on Communications and Network Security (CNS), pp 124–132. https://doi.org/10.1109/CNS.2019.8802778
30. Li F, Cheng D, Hadjieleftheriou M, Kollios G, Teng S-H (2005) On trip planning queries in spatial databases. In: Proceedings of the 9th International Conference on Advances in Spatial and Temporal Databases, SSTD’05. Springer, Berlin, pp 273–290. https://doi.org/10.1007/11535331_16
31. Real datasets for spatial databases: road networks and points of interest. https://www.cs.utah.edu/~lifeifei/SpatialDataset.htm
32. Li Y, Cao X, Yuan Y, Wang G (2019) Privsem: protecting location privacy using semantic and differential privacy. World Wide Web 22(6):2407–2436