Privacy Protected Spatial Query Processing for Advanced Location Based Services

Wei-Shinn Ku[†], Yu Chen[§] and Roger Zimmermann[‡]
[†]*Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849*
E-mail: weishinn@auburn.edu
[§]*Department of Electrical & Computer Engineering, Binghamton University, Binghamton, NY 13902*
[‡]*Department of Computer Science, National University of Singapore, Singapore 117590*

Abstract. Due to the popularity of mobile devices (e.g., cell phones, PDAs, etc.), location-based services have become more and more prevalent in recent years. However, users have to reveal their location information to access location-based services with existing service infrastructures. It is possible that adversaries could collect the location information, which in turn invades user's privacy. There are existing solutions for query processing on spatial networks and mobile user privacy protection in Euclidean space. However there is no solution for solving queries on spatial networks with privacy protection. Therefore, we aim to provide network distance spatial query solutions which can preserve user privacy by utilizing K-anonymity mechanisms. In this paper, we propose an effective location cloaking mechanism based on spatial networks and two novel query algorithms, PSNN and PSRQ, for answering nearest neighbor queries and range queries on spatial networks without revealing private information of the query initiator. The effectiveness of our cloaking mechanism and privacy protected algorithms has been validated using real world road networks. In addition, we demonstrate the appeal of our technique using extensive simulation results.

Keywords: Privacy protection, mobile communications, mobile data management, location-based services.

1. Introduction

As a result of recent technological advances, mobile devices with significant computational abilities, gigabytes of storage capacities, and wireless communication capabilities have become increasingly popular. In addition, positioning techniques are embedded into more and more mobile devices. Based on these advances, new mobile applications allow users to issue location-dependent queries in a ubiquitous manner. Examples of such location-dependent queries include '*find the nearest gas station*' and '*find the top three closest French restaurants*'. To get location-dependent data, users have to reveal their current locations when launching location-dependent queries. From the location-dependent query logs of location-based service providers (LBSP), it is possible that adversaries could collect the location history and monitor the behavior of some users, which in turn invades their privacy [23]. Therefore, with the popularity of location-based services (LBS), mobile user privacy protection becomes a very important research issue for future generation communication and networking.

Recent research has explored the K-anonymity concept [21][1] in which one trusted server is needed to cloak at least K users' locations for protecting location privacy. In order to implement K-anonymity, one trusted server is set up to collect user location information and perform cloaking procedures in which the exact location of the query requester is blurred as

[1] Note that we use the symbol K for the degree of anonymity and k for k nearest neighbor queries.

a cloaked spatial area whose boundary is defined by the locations of $K-1$ other users. Then, the trusted server will send the location-dependent query along with the cloaked spatial area to location-based service providers to retrieve location-dependent data. Note that since the query location is an area instead of a single query point, location-dependent service providers should fetch those query results based on the cloaked spatial region. Prior work in [16] proposed a framework for location-based services without compromising location privacy. However, only a free space environment is considered, which is not fully applicable in real world environments. On the other hand, recent research has produced novel mechanisms to compute location-dependent queries on spatial networks. For example, executing nearest neighbor queries based on the spatial network distance provides a more realistic measure for applications where mobile user movements are constrained by underlying networks. Though devising spatial query schemes in spatial networks, the prior works in [17, 13] did not consider location privacy issues. Thus, we aim to provide spatial query (nearest neighbor query and range query) solutions with privacy protection concerns in this study. With our techniques location-based service users can enjoy high quality results without sacrificing their privacy. The contributions of our study are as follows.

- We design an improved cloaking mechanism for spatial network search space.
- We propose a novel algorithm for solving privacy protected nearest neighbor queries on spatial networks.
- We extend our nearest neighbor query solution to answer range queries with protection of privacy.
- We demonstrate the feasibility and efficiency of our approaches through extensive simulations.

The rest of the paper is structured as follows. Section 2 surveys the related work of spatial queries, location-based services, and privacy protection techniques for mobile applications. The system architecture and the improved cloaking mechanism are illustrated in Section 3. Our novel query processing algorithms are detailed in Section 4 and the experimental results are presented in Section 5. Finally, Section 6 concludes the paper and outlines future research directions.

2. Related Work

In this section, we introduce the background information and related research regarding spatial queries, location-based services, and location privacy preservation.

2.1. Spatial Queries

We focus on two common types of spatial queries, namely k nearest neighbor queries and range queries. With R-tree [11] based spatial indices, depth-first search (DFS) [18] and best-first search (BFS) [12] have been the prevalent branch-and-bound techniques for processing nearest neighbor (NN) queries. The DFS method recursively expands the index nodes for

searching nearest neighbor candidates. At each newly visited non-leaf node, DFS computes the ordering metrics for all its child nodes and applies pruning strategies to remove unnecessary branches. When a leaf node is reached, the data objects are retrieved and the nearest neighbor candidates are updated. In addition, the BFS technique utilizes a priority queue to store nodes to be explored through the search process. The nodes in the queue are sorted according to their minimum distance (MINDIST) to the query point. During search, the BFS repeatedly dequeues the top entry in the queue and enqueues its child nodes with their MINDIST into the queue. When a data entry is dequeued, it is inserted into the result set. In order to increase the NN query accuracy, recent research proposed solutions based on spatial networks. Kolahdouzan et al. [13] presented a novel approach to efficiently evaluate k NN queries in spatial network databases using a first order Voronoi diagram. Papadias et al. [17] proposed two algorithms, the Incremental Euclidean Restriction (IER) algorithm and the Incremental Network Expansion (INE) algorithm to solve nearest neighbor queries on spatial networks.

2.2. Location Privacy Preservation

With the popularity of LBS, privacy protection for mobile users has become an important issue [10, 19]. Gruteser et al. [9] proposed a middleware architecture and algorithms for maintaining location K-anonymity. Their algorithms adjust the resolution of location information along spatial or temporal dimensions to fulfill the required anonymity constraints. Based on the work in [9], a unified privacy personalization framework is proposed in [8] to support different levels of anonymity according to the requests of users. However, these previous research approaches mainly focused on the system architecture and the location anonymizer design rather than query processing. Mokbel proposed to employ a trusted third party, the *location anonymizer*, which expands the user location into a spatial region for protecting user privacy [15]. Mokbel et al. also proposed related privacy-aware query processing algorithms [16]. However their query processing solutions are based on Euclidean metrics. In real life, mobile users cannot move freely in space but are usually constrained by underlying networks (e.g., cars on roads, trains on tracks, etc.). Therefore, we need solutions for processing privacy protected queries on spatial networks [14].

3. System Architecture

In this section, we describe the system architecture for supporting privacy protected spatial queries with underlying spatial networks. Figure 1 depicts our operating environment with three main entities: *mobile users*, the *location cloaker*, and *location-based service providers*. We consider mobile clients such as cell phones, personal digital assistants (PDA), and laptops, that are instrumented with global positioning systems (GPS) for continuous position information. Furthermore, we assume that there are access points/base stations distributed in the system environment for mobile devices to communicate with the location cloaker. All users are mobile and travel on the underlying network and they also hold *privacy policies* which specify the privacy requirements of each user. In this research we focus on three parameters, K-anonymous, the minimum cloaked region size – R_{min}, and the minimum covered network segments – S_{min}, that are included in the user privacy policies. A user can demand

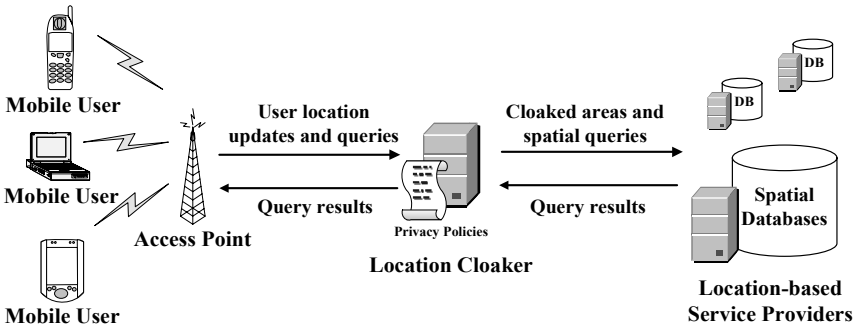

Figure 1. The system architecture.

the cloaked area to cover the locations of $K-1$ closest peers for anonymizing its exact location. In order to keep a reasonable size of the cloaked area in high user density regions, the user decides the minimum acceptable size of R_{min}. Since mobile users are running on underlying spatial networks, the cloaked region has to cover a minimum number of road segments, S_{min}, to avoid location tracking by adversaries. Based on privacy requirements at different locations or time slots, a user can update his/her privacy policies at any time. Table I summarizes the symbolic notation used throughout this paper.

Table I. Symbolic notations.

Symbol	Meaning
A_c	A cloaked area
K	The degree of anonymity
R_{min}	The minimum cloaked region size
S_{min}	The minimum road segment number within a cloaked region
q	A query mobile user
Q	A priority queue
Seg_i	The network segments inside the cloaked area
Seg_o	The network segments outside the cloaked area
$\mathbb{T}$	The set of all the points that intersect between the cloaked area A_c and the underlying networks
$Dist(a,b)$	The network distance between objects a and b
$\lvert\mathbb{A}\rvert$	The cardinality of set $\mathbb{A}$

3.1. THE LOCATION CLOAKER

Compared with location-based service providers, the location cloaker is an intermediate agent which can be trusted by mobile users. The location cloaker receives continuous location updates from mobile users and blurs their exact locations into cloaked areas A_c according to individual user privacy policies before forwarding the information to location-based service providers (e.g., for buddy searching services). In addition, the location cloaker also anonymizes the location of any query requesting user q to a cloaked region before forwarding the query to related location-based service providers. Note that any user identity

related information in the query is also removed by the location cloaker during the cloaking process.

According to previous research [16, 9, 8, 10] there are several different mechanisms to support location anonymization. Compared with existing solutions, the location anonymizing technique proposed by Mokbel et al. [16] has prominent efficiency (low cloaking time) and flexibility (user defined privacy profile) for the cloaking process in free space environments. However, the proposed location anonymizer does not consider the characteristics of spatial network environments. Figure 2 demonstrates an example. The cloaking region in Figure 2a is generated without considering the underlying spatial networks and there is only one road segment (i.e., the W 35^{th} Street) covered in A_c. Advisories with the map of this city can easily infer that the mobile user is currently driving on the W 35^{th} Street. Consequently, the effect of the location anonymizing mechanism proposed in [16] is drastically weakened in spatial network environments.

In our system, we adopt the grid-based pyramid data structures proposed in [22, 1] for managing the locations of mobile users. The pyramid structure is dynamically maintained for monitoring the current number of mobile users within each unit space of the whole spatial area [16]. Consequently, the size of the cloaked region can be efficiently figured out. In order to avoid the aforementioned drawbacks, users have to define a minimum number of road segments which should be covered in the cloak region. The updated cloaked region after taking S_{min} into account is demonstrated in Figure 2b ($S_{min} = 3$). With a minimum number of road segments within A_c, mobile user privacy can be positively secured.

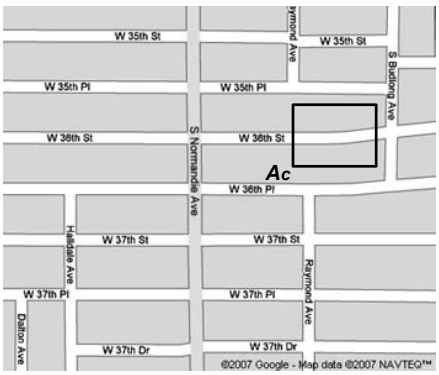

Fig. 2a. Location cloaking without considering of underlying spatial networks.

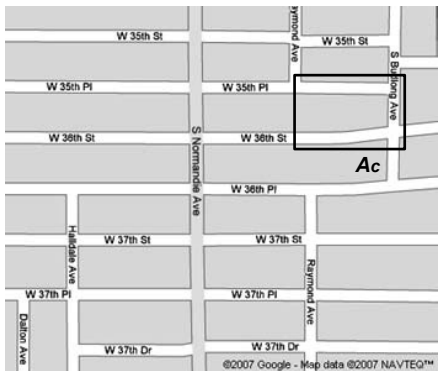

Fig. 2b. Location cloaking with considering of underlying spatial networks.

Figure 2. Location cloaking in spatial network environments.

3.2. LOCATION-BASED SERVICE PROVIDERS

Location-based service providers play the role of spatial data maintainers and spatial query processors in our system. In order to handle privacy protected spatial queries, location-based service providers implement *privacy protected query processors* in their databases. The privacy protected query processor has the ability to process cloaked spatial queries

efficiently and retrieves the *inclusive result set* (i.e., the minimal set which covers all the possible answers) for query requesters. After receiving the result set, mobile users can distill the exact answers from their locations in linear time. The privacy policies of a user determine the computational complexity of his/her spatial queries. Strict privacy requirements (i.e., large K, R_{min}, and S_{min} values) increase the complexity of processing the query.

In addition to a privacy protected data processor, a LBSP also needs to maintain spatial databases for storing (cloaked) user locations, spatial data and road networks. The stored spatial data can be categorized as public data and private data. Public data covers static objects such as restaurants, hotels, and gas stations and also the dynamic information (e.g., real-time bus locations) which are directly open to public queries. In contrast, private data mainly comprise cloaked mobile user locations from the location cloaker. Based on the two data categories, the spatial queries submitted to a LBSP can be classified as four types: (1) *public queries over public data*, (2) *public queries over private data*, (3) *private queries over public data*, and (4) *private queries over private data*. For the first query type, there were already existing solutions proposed in [17, 13]. Because the movement of mobile users is limited by the underlying road networks, we can easily extend the mechanisms proposed in [17, 13] for solving queries of the second query type (e.g., Figure 3a) with probability density functions [5]. To the best of our knowledge, there is no existing solution for the third query type. Similarly the fourth query type (e.g., Figure 3b) can be answered by extending the algorithms of the third query type. Therefore, we propose our novel techniques for solving private queries over public data on spatial networks in Section 4.

In addition, as discussed in [17], we assume that the spatial network database supports the following primitive operations:

- $inside\_segments(A_c)$: returns a set of subsegments of a network N which intersects with the cloaked area A_c.
- $find\_objects(segment_x)$: returns the data objects which fall on the input network segment $segment_x$.

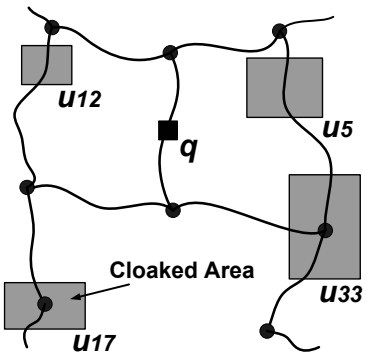

Fig. 3a. Public query over private data.

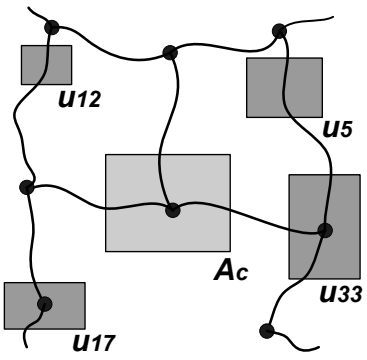

Fig. 3b. Private query over private data.

Figure 3. Two novel query types.

– $Dist(p_1, p_2)$: calculates the network distance of two input points, p_1 and p_2, in the underlying network by applying an algorithm (e.g., Dijkstra's algorithm [6]) to compute the shortest path between p_1 and p_2.

4. Privacy Protected Query Processing

We illustrate our mechanisms for solving private queries over public data on road networks in this section. We focus on two popular query types, nearest neighbor queries and range queries.

4.1. Privacy Protected Nearest Neighbor Query on Spatial Networks

Given a query point q and an object data set $\mathbb{S}$, a network k nearest neighbor query retrieves the k objects of $\mathbb{S}$ closest to q based on the network distance. Papadias et al. [17] proposed two algorithms (*incremental euclidean restriction* and *incremental network expansion*) to efficiently solve nearest neighbor queries with spatial network databases. However in order to protect user privacy, a location-based service provider can only receive cloaked spatial areas from users. Therefore, LBSPs need to have a competent mechanism for retrieving an inclusive query result set based on the input cloaked area and the underlying spatial network. We design a *privacy protected spatial network nearest neighbor query* (PSNN) algorithm by extending the incremental network expansion solution [17].

PSNN first locates all the intersection points between the edges of the input cloaked area A_c and the spatial network as a point set $\mathbb{T}$ by executing primitive database operations. If $\mathbb{T}$ is not empty, A_c covers at least one network segment. We denote these network segment(s) within A_c as Seg_i and the segments outside A_c as Seg_o. According to the network topology, the network edges inside A_c can be fully connected or comprise several separate subgroups. Figure 4 illustrates the two cases. Since we designed efficient solutions for the case in which the network edges inside A_c are fully connected, we can utilize the *divide and conquer strategy* to handle the case demonstrated in Figure 4b. Consequently, we perform a pre-process for splitting the input cloaked area until each subregion contains only one connected

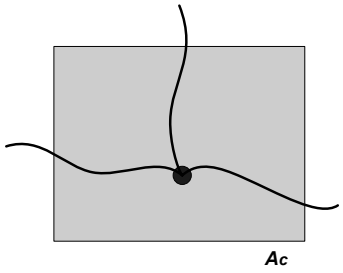

Fig. 4a. The network segments in A_c are totally connected.

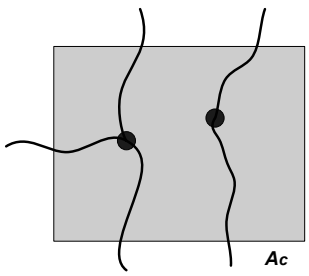

Fig. 4b. The network segments in A_c comprise two separate subgroups.

Figure 4. The two possible connection statuses of the network segments inside a cloaked area A_c.

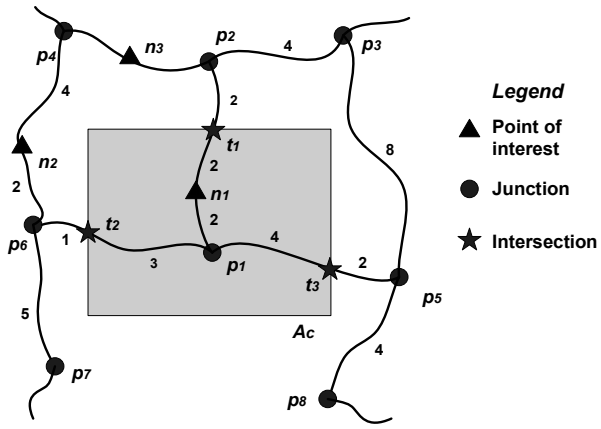

Figure 5. Searching k network nearest neighbors with PSNN where the cloaked area contains k objects ($k = 1$ in this example).

network segment set. Then we execute our algorithms on each subregion separately and merge their results after the whole computation. For ease of presentation, we assume the pre-process has been done in the following sections.

We observe that the number of data objects inside a cloaked area A_c can meet one of two conditions: (i) there are at least k objects inside A_c or (ii) there are fewer than k objects within A_c.

4.1.1. *The Cloaked Area Contains at Least k Objects*

Since Seg_i contains a limited number of network segments, PSNN starts the search on Seg_i for retrieving data objects inside A_c. If Seg_i covers more than or equal to k data objects, the search can be finished by expanding the intersection points in $\mathbb{T}$. First PSNN includes the data objects within A_c into the result set $\mathbb{R}$. Next, for each point t_i in $\mathbb{T}$, PSNN calculates the distance from t_i to its k^{th} nearest object inside A_c as $Dist(t_i, n_k)$, and then expands outward from point t_i to search for data objects on Seg_o within distance $Dist(t_i, n_k)$ (the search upper bound). Consequently, we can cover the special case when q is located exactly at t_i. All the newly discovered data objects are inserted into the result set $\mathbb{R}$.

Figure 5 demonstrates an example where the circles represent the nodes in the modeling graph, triangles indicate data objects, and the gray rectangle stands for the cloaked area. Assuming that only one nearest neighbor is queried ($k = 1$), PSNN first retrieves n_1 by searching the network segments inside A_c. Since the number of data objects found in A_c is equal to 1, PSNN computes the network distance from t_1, t_2, and t_3 to n_1 as 2, 5, and 6 respectively. Afterwards, PSNN expands the search space outbound from the three intersection points according to their distance to n_1. No data object is found from the expansion of t_1 and t_3. The expansion of t_2 reaches n_2 and it is inserted into the result set. Consequently, the final search result set covers objects n_1 and n_2.

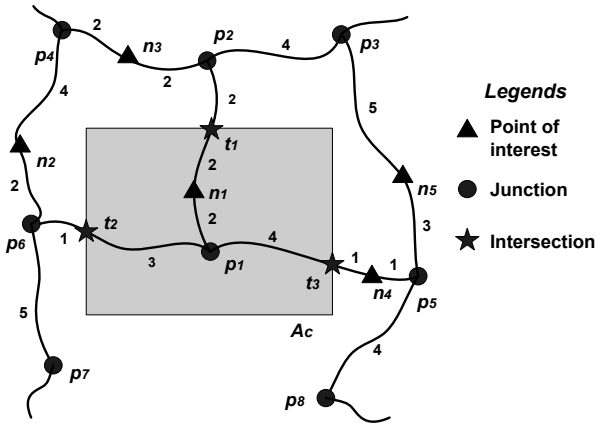

Figure 6. Searching k network nearest neighbors with PSNN where the cloaked area contains fewer than k objects ($k = 2$ in this example).

4.1.2. *The Cloaked Area Contains Fewer than k Objects*

If there are fewer than k data objects found on Seg_i, PSNN has to search the network segments which are outside of A_c. Since the points in $\mathbb{T}$ are all on the boundary of A_c, they determine the network search expansion upper bound. Consequently, PSNN executes the network expansion from all the intersection points for retrieving an inclusive result set. For every point t_i in $\mathbb{T}$, the PSNN algorithm first retrieves the network segment $p_m p_n$ which passes through t_i and searches all data objects on this segment. In the mean time, the two end points p_m and p_n are inserted into a queue Q with their distance to t_i. Afterward, if the search found fewer than k objects on $p_m p_n$ or the search retrieved no fewer than k objects but one end point of $p_m p_n$ whose distance to t_i is shorter than $Dist(t_i, n_k)$ (n_k is the k^{th} nearest neighbor of t_i), the end point p_x which is closer to t_i will be popped from Q and expanded. For each non-visited adjacent point p_y of p_x, PSNN searches $p_x p_y$, updates the result set, and inserts p_y with its distance to t_i into Q. Then the point in Q with the shortest distance to t_i is de-queued. The procedure repeats until k nearest neighbors of t_i are found and the k objects are inserted into the result set $\mathbb{R}$. PSNN repeats the whole process until it has expanded all the points in $\mathbb{T}$.

Assuming k is equal to two, Figure 6 illustrates an example. PSNN retrieves only n_1 inside A_c and it has to expand the three intersection points, t_1, t_2, and t_3 outward. First, PSNN locates the segment $p_1 p_2$ which covers t_1 and the segment $p_1 p_2$ only covers object n_1. Since p_2 is closer to t_1 than p_1, it is expanded and p_1 is inserted into Q with its distance to t_1. The expansion of p_2 reaches p_3 and p_4, and object n_3 is found on $p_2 p_4$. At this moment, since Q contains $\langle (p_1, 4), (p_3, 6), (p_4, 6) \rangle$ and since $Dist(t_1, n_3) = 4$, the search terminates. The top two nearest neighbors of t_1 are n_1 and n_3 and they are inserted into $\mathbb{R}$. PSNN continues the process with t_2 and retrieves the top two nearest neighbors of t_2 as n_2 and n_1. The search of t_3 demonstrates why we have to expand the joint road segments until we reach the search bound. The search on $p_1 p_5$ retrieves n_4 and we know A_c covers n_1, however n_4 and n_1 are not the true top two nearest neighbors of t_3. After finishing the complete search procedure, we can retrieve the correct nearest neighbor set of t_3 as n_4 and n_5. The final result set $\mathbb{R}$

Algorithm 1 PSNN (q, k, A_c)

1: initialize R_i
2: locate the intersection points between A_c and the underlying network as $\mathbb{T} = \{t_1, \ldots, t_m\}$
3: $Seg_i = inside\_segments(A_c)$
4: search objects on Seg_i and insert the retrieved objects into $\mathbb{R}$
5: **if** Seg_i covers $\geq k$ objects **then**
6: **for** $\forall t_i \in \mathbb{T}$ **do**
7: expand t_i outward A_c for searching objects within distance $Dist(t_i, n_k)$
8: insert any discovered objects into $\mathbb{R}$
9: **end for**
10: **else**
11: **for** $\forall t_i \in \mathbb{T}$ **do**
12: $p_m p_n = find\_segment(t_i)$
13: $R_i = R_i \cup find\_objects(p_m, p_n)$
/* $\{n_1, \ldots, n_k\}$ = the k nearest objects in R_i sorted in ascending order, $n_j, n_{j+1} \ldots, n_k$ may be $\emptyset$, if $|R_i| < k$ */
14: $Dist_{max} = Dist(t_i, n_k)$
/* $Dist_{max} = \infty$, if $n_k = \emptyset$ */
15: $Q = \langle (p_m, Dist(p_m, t_i)), (p_n, Dist(p_n, t_i)) \rangle$
16: de-queue the node p in Q with smaller $Dist(p, t_i)$
17: **while** $Dist(p, t_i) < Dist_{max}$ and the data set contains at least k objects **do**
18: **for** each non-visited adjacent vertex p_x of p **do**
19: $R_i = R_i \cup find\_objects(p_x, p)$
20: update $Dist_{max}$ with sorted R_i
21: en-queue$(p_x, Dist(p_x, t_i))$
22: **end for**
23: de-queue the next node p in Q
24: **end while**
25: $\mathbb{R} = \mathbb{R} \cup R_i$
26: **end for**
27: **end if**
28: Sort $\mathbb{R}$ for removing duplicates
29: **return** $\mathbb{R}$

covers n_1, n_2, n_3, n_4, and n_5. Then, the result set $\mathbb{R}$ will be returned to the query mobile user q and q will evaluate the query locally over the received $\mathbb{R}$ for retrieving the exact query result. The complete algorithm of PSNN is shown in Algorithm 1. Since in the worst case we have to search all the road segments in the search space, the complexity of Algorithm 1 is $O(n)$ where n is the total number of road segments.

4.2. Privacy Protected Range Query on Spatial Networks

We define a spatial network range query as follows: given a query point q, a range value r, and an object data set $\mathbb{S}$, the query retrieves all elements of $\mathbb{S}$ that are within network distance r from q. For executing a *privacy protected spatial network range query* (PSRQ), first we have to locate the intersection points of the cloaked spatial area A_c and the underlying road network as a point set $\mathbb{T} = \{t_1, \ldots, t_m\}$. Then, PSRQ searches the network segments inside A_c and inserts the retrieved objects into $\mathbb{R}$. For each point t_i in $\mathbb{T}$ we can compute a set of *candidate segments* within network range r from t_i and then retrieve the data objects falling

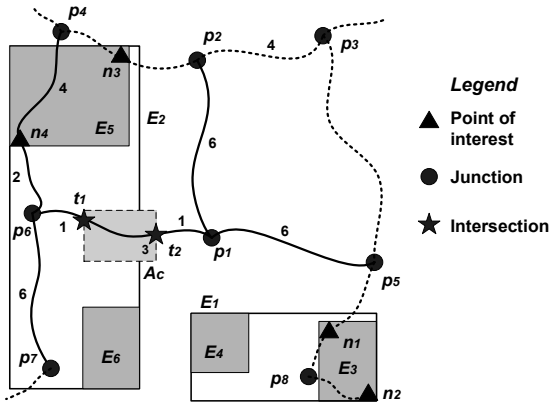

Figure 7. Network range query with PSRQ.

on these segments. Because the search range r could be a large number and it may cover many candidate segments, it is inefficient to check each candidate segment with the primitive operation $find\_objects(segment)$. Therefore, we utilize the intersection join function [4] for retrieving all intersection object pairs from the spatial network R-tree and the object R-tree. When reaching the leaf node level, PSRQ executes the plane-sweep method with the object R-tree nodes which intersect with the MBR of at least one candidate segment and stores the qualified objects into $\mathbb{R}$.

An example is demonstrated in Figure 7. Assuming r is equal to a 7 unit distance, PSRQ joins the candidates segments (solid lines) with the object R-tree and retrieves leaf node E_5 intersecting with segment p_4p_6. After executing the intersection test (plane-sweep method), PSRQ retrieves n_4 as the query result. The complete algorithm of PSRQ is shown in Algorithm 2.

Algorithm 2 PSRQ (q, r, A_c)

1: locate the intersection points between A_c and the underlying network as $\mathbb{T} = \{t_1, \ldots, t_m\}$
2: $Seg_i = inside\_segments(A_c)$
3: search objects on Seg_i and insert the retrieved objects into $\mathbb{R}$
4: **for** each point t_i in $\mathbb{T}$ **do**
5: Compute all the candidate segments within distance r from t_i as C
6: Intersection join C with the object R-tree for finding intersection leaf nodes
7: **for** each retrieved leaf node E_i **do**
8: R_{ti} = intersection test of the intersected segments with data objects in E_i
9: **end for**
10: $\mathbb{R} = \mathbb{R} \cup R_{ti}$
11: **end for**
12: Sort $\mathbb{R}$ for removing duplicates
13: **return** $\mathbb{R}$

5. Experimental Validation

In this section, we present extensive simulation results of our location cloaker and query processing algorithms. We implemented our cloaking technique and privacy protected query

algorithms in a simulator to evaluate the performance of our approach. Our main objectives are to observe the influence of performance related factors (e.g., cloaked region size) on the system and to test the feasibility of our approach with real world parameter sets. Performance is measured in terms of the cloaked region size, result set size, and CPU time. All simulation results were recorded after the system model reached steady state.

5.1. Simulator Implementation

Our simulator consists of three main components, the *mobile environment*, the *location cloaker*, and the *location-based service provider*. For the mobile environment, we utilized the *network-based moving objects generation framework* [3] to generate a set of mobile users and the underlying road network inside a geographical area, measuring 10 miles by 10 miles. Each mobile user is an independent object which encapsulates all its related parameters (e.g., its current speed and destination). We implemented the location cloaker as a new module for interacting with mobile users to anonymize spatial queries in the framework. Our privacy protected query approaches (Section 4.1 and Section 4.2) were also implemented inside the framework as new functions and play the role of the LBSP. We obtained our road network data of the City of Los Angeles and Riverside County from the TIGER/Line street vector data available from the U.S. Census Bureau.

5.2. Exploring Performance Influencing Factors

We are interested in the effect of four major performance influencing factors: the minimum road segment number (S_{min}), the cloaked region size, the number of k, and the Point of Interest (POI) number, of spatial queries. We experimented with both nearest neighbor queries and range queries with these factors as follows.

5.2.1. *Effect of the Minimum Road Segment Number*

We first varied the minimum road segment number (S_{min}) from 1 to 20 with real world road street vector data sets. We define a road segment as a street section between two adjacent intersections. The influence of S_{min} was experimented with the street vector data from an urban area (in the City of Los Angeles) and a suburban area (in Riverside County). Figure 8 demonstrates the cloaked region size with different minimum road segment number. Since the road segments are sparse and the distance between two adjacent intersections are longer in suburban areas, the cloaked region size expanded exponentially in the suburban space when we raised the minimum road segment number. On the contrary, the expansion of the cloaked region in the urban area is quite stable because the road segment density in urban areas is much higher.

5.2.2. *Effect of the Cloaked Region Size*

Next we increased the cloaked region size from 2% to 10% of the whole experimental region and the cloaked region size also reflects the influences of the two privacy policy settings – K and R_{min}. Figure 9 demonstrates the result set size and query processing time with different cloaked region sizes. Since a bigger cloaked region usually intersects with more underlying network segments, it generates a larger candidate result set and takes longer to process. As

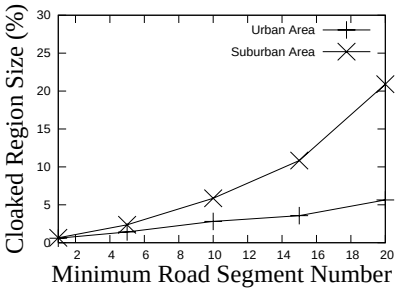

Figure 8. The effect of the minimum road segment number.

shown in Figure 9a., the curve of PSRQ remarkably increases because the result set covers all the POIs within the cloaked region and the search range r. In contrast, PSNN removes duplicated objects from $\mathbb{R}$. In addition, the query processing time of PSNN increases notably with an enlarged cloaked region size as illustrated in Figure 9b.

5.2.3. *Effect of k*

We tested the impact of varying the number of requested nearest neighbors, i.e., k. We altered k in the range from 4 to 20. As shown in Figure 10, the result set size grows when we raised k from 4 to 20 and we also observe that the result set increases super-linearly when k is a relatively large number. For example, the result set covers around 25% more objects than the queried k number when k is equal to 20. This factor has a similar super-linear influence on the query processing time as demonstrated in Figure 10b.

5.2.4. *Effect of the Number of POI*

To see the effect of varying the total POI number, we increased the total POI number from 200 to 1000 in the simulation environment. Figure 11 illustrates the result set size and query processing time of PSNN and PSRQ with increasing POI numbers. We notice that with a higher POI density the result set of PSRQ increases conspicuously compared with PSNN. This is expected, since PSRQ has to report all the objects inside the cloaked region. In addition, the query processing time of PSNN is always longer than that of PSRQ.

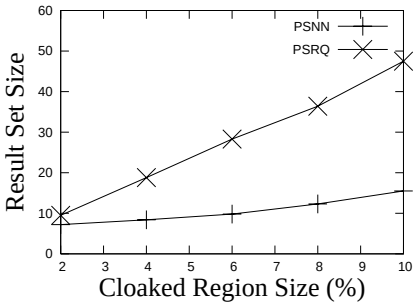

Fig. 9a. Result Set Size.

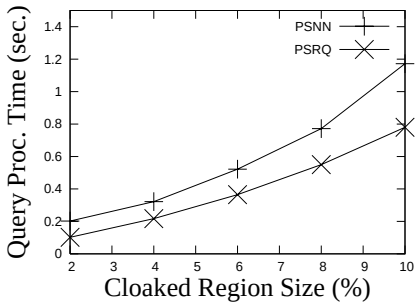

Fig. 9b. Query Processing Time.

Figure 9. The effect of the cloaked region size.

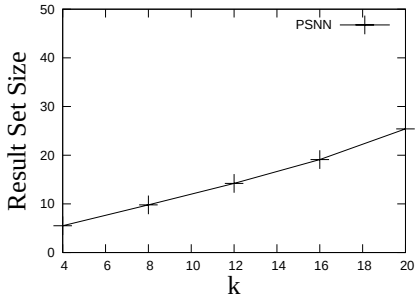

Fig. 10a. Result Set Size.

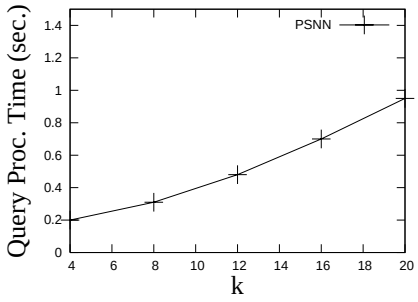

Fig. 10b. Query Processing Time.

Figure 10. The effect of k.

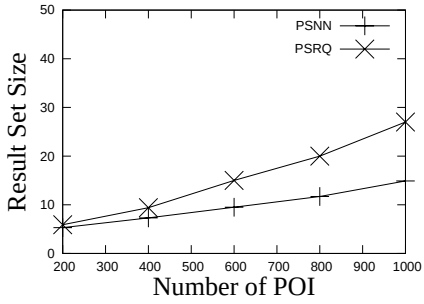

Fig. 11a. Result Set Size.

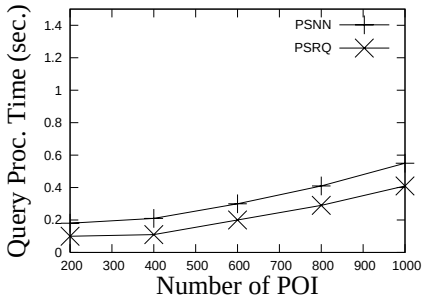

Fig. 11b. Query Processing Time.

Figure 11. The effect of the POI number.

5.3. EXPERIMENTS WITH REAL WORLD PARAMETER SETS

In order to test our approach with realistic environments, we obtained our simulation parameters from public data sets which report vehicle density and POI density in the City of Los Angeles and Riverside County. We term the two parameter sets based on these real-world statistics the *City of Los Angeles* parameter set and the *Riverside County* parameter set.

The City of Los Angeles and the Riverside County parameter sets represent a very dense, urban area and a low-density, more rural area respectively. For experimental purpose, we mixed the two real parameter sets to generate a third, synthetic data set. Table II lists the two parameter sets where POI_{Num} stands for the total POI number and MU_{Num} stands for the total mobile user number.

We used both input parameter sets, City of Los Angeles and Riverside County, to simulate our techniques for solving kNN queries and range queries. The simulation results are demonstrated in Figure 12. The City of Los Angeles data set generates a larger result and requires a longer CPU time because of its high user mobility and POI density. However, the performance of the City of Los Angeles data set did not deteriorate much compared with the far sparser Riverside County parameter set. The results for range queries exhibit a similar trend as for kNN queries.

Table II. The simulation parameter sets.

Parameter	City of Los Angeles	Riverside County	Synthetic Suburbia	Units
POI_{Num}	580	260	420	
MU_{Num}	15325	3430	9378	
K-anonymity	150	150	150	peers
R_{min}	1	1	1	mile^2
S_{min}	5	5	5	
k	5	5	5	
r	1	1	1	mile

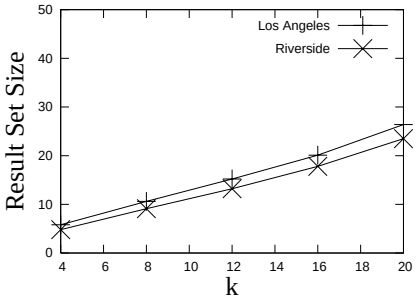

Fig. 12a. Result Set Size.

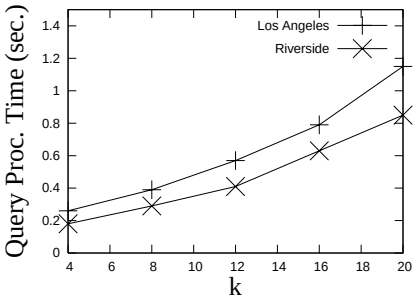

Fig. 12b. Query Processing Time.

Figure 12. The effect of k with real-world parameter sets.

6. Conclusions

In this paper we present an advanced location cloaker and two novel algorithms for processing k nearest neighbor queries and range queries on spatial networks with privacy protection. The main idea is to hide the exact mobile user location with a cloaked region. The cloaked region covers the query requester and at least $K - 1$ other users based on the K-anonymity concept. The spatial queries are executed based on both the cloaked region and the underlying networks. A candidate result set will be returned to the requesting user who filters out the exact answer. Our comprehensive simulations with real and synthetic parameter sets demonstrate the efficiency of our methods. We plan to extend the proposed solutions to support spatial network based query processing in other privacy protection models. In addition, a road network based indexing technique will also be studied in the future.

References

1. Walid G. Aref and Hanan Samet. Efficient Processing of Window Queries in the Pyramid Data Structure. In *PODS*, pages 265–272, 1990.

2. Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *SIGMOD Conference*, pages 322–331, 1990.
3. Thomas Brinkhoff. A Framework for Generating Network-Based Moving Objects. *GeoInformatica*, 6(2):153–180, 2002.
4. Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient Processing of Spatial Joins Using R-Trees. In *SIGMOD Conference*, pages 237–246, 1993.
5. Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Evaluating Probabilistic Queries over Imprecise Data. In *SIGMOD Conference*, pages 551–562, 2003.
6. Edsger. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959.
7. Alon Efrat and Arnon Amir. Buddy Tracking - Efficient Proximity Detection Among Mobile Friends. In *INFOCOM*, 2004.
8. Bugra Gedik and Ling Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *25th International Conference on Distributed Computing Systems*, pages 620–629, 2005.
9. Marco Gruteser and Dirk Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *MobiSys*, 2003.
10. Marco Gruteser and Xuan Liu. Protecting Privacy in Continuous Location-Tracking Applications. *IEEE Security & Privacy*, 2(2):28–34, 2004.
11. Antonin Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD Conference*, pages 47–57, 1984.
12. Gísli R. Hjaltason and Hanan Samet. Distance Browsing in Spatial Databases. *ACM Trans. Database Syst.*, 24(2):265–318, 1999.
13. Mohammad R. Kolahdouzan and Cyrus Shahabi. Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases. In *VLDB*, pages 840–851, 2004.
14. Wei-Shinn Ku, Roger Zimmermann, Wen-Chih Peng, and Sushama Shroff. Privacy Protected Query Processing on Spatial Networks. In *ICDE Workshops*, pages 215–220, 2007.
15. Mohamed F. Mokbel. Towards Privacy-Aware Location-Based Database Servers. In *ICDE Workshops*, page 93, 2006.
16. Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *VLDB*, pages 763–774, 2006.
17. Dimitris Papadias, Jun Zhang, Nikos Mamoulis, and Yufei Tao. Query Processing in Spatial Network Databases. In *VLDB*, pages 802–813, 2003.
18. Nick Roussopoulos, Stephen Kelley, and Frédéic Vincent. Nearest Neighbor Queries. In *SIGMOD Conference*, pages 71–79, 1995.
19. Bill N. Schilit, Jason I. Hong, and Marco Gruteser. Wireless Location Privacy Protection. *IEEE Computer*, 36(12):135–137, 2003.
20. Sarah Spiekermann. General Aspects of Location Based Services. In *Location-Based Services*, pages 9–26. 2004.
21. Latanya Sweene. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.
22. Steven L. Tanimoto and Theodosios Pavlidis. A Hierarchical Data Structure for Picture Processing. *Computer Graphics and Image Processing*, 4(2):104–113, June 1975.
23. John Voelcker. Stalked by Satellite: An Alarming Rise in GPS-enabled Harassment. *IEEE Spectrum*, 47(7):15–16, 2006.