Privacy-Conscious Location-Based Queries in Mobile Environments

Jianliang Xu, *Senior Member, IEEE,* Jing Du, Xueyan Tang, *Member, IEEE,* and Haibo Hu

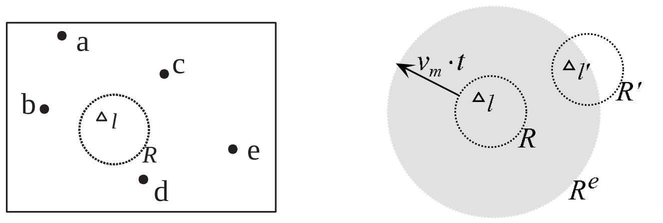
***Abstract*— In location-based services, users with location-aware mobile devices are able to make queries about their surroundings anywhere and at any time. While this ubiquitous computing paradigm brings great convenience for information access, it also raises concerns over potential intrusion into user location privacy. To protect location privacy, one typical approach is to cloak user locations into spatial regions based on user-specified privacy requirements, and to transform location-based queries into region-based queries. In this paper, we identify and address three new issues concerning this location cloaking approach. First, we study the representation of cloaking regions and show that a circular region generally leads to a small result size for region-based queries. Second, we develop a mobility-aware location cloaking technique to resist trace analysis attacks. Two cloaking algorithms, namely *MaxAccu_Cloak* and *MinComm_Cloak*, are designed based on different performance objectives. Finally, we develop an efficient polynomial algorithm for evaluating circular-region-based kNN queries. Two query processing modes, namely *bulk* and *progressive*, are presented to return query results either all at once or in an incremental manner. Experimental results show that our proposed mobility-aware cloaking algorithms significantly improve the quality of location cloaking in terms of an entropy measure without compromising much on query latency or communication cost. Moreover, the progressive query processing mode achieves a shorter response time than the bulk mode by parallelizing the query evaluation and result transmission.**

Index Terms: Location-based services, location privacy, query processing, mobile computing.

I. Introduction

Location-based services (LBS) are emerging as a major application of mobile geospatial technologies [7], [21], [23], [35]. In LBS, users with location-aware mobile devices are able to make queries about their surroundings anywhere and at any time. Spatial range queries and k-nearest-neighbor (kNN) queries are two types of the most commonly used queries in LBS. For example, a user can make a range query to find out all shopping centers within a certain distance of her current location, or make a kNN query to find out the k nearest gas stations. In these queries, the user has to provide the LBS server with her current location. But the disclosure of location information to the server raises privacy concerns, which have hampered the widespread use of LBS [18], [19], [30]. Thus, how to provision location-based services while protecting user location privacy has recently become a hot research topic [6], [13], [15], [24], [25], [26].

Jianliang Xu, Jing Du, and Haibo Hu are with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong. Xueyan Tang is with the School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798.



(a) Location cloaking (b) Isolated cloaking

Fig. 1. Dynamic Location Cloaking

Location cloaking is one typical approach to protecting user location privacy in LBS [13], [14], [15], [26]. Upon receiving a location-based spatial query (e.g., a range query or a kNN query) from the user, the system cloaks the user's current location into a *cloaking region* based on the user's privacy requirement. The location-based spatial query is thus transformed into a *region-based spatial query* before being sent to the LBS server. The LBS server then evaluates the region-based query and returns a *result superset*, which contains the query results for all possible location points in the cloaking region. Finally, the system refines the result superset to generate the exact results for the query location. Figure 1a shows a sample NN query. Instead of providing the exact location l, the system submits a cloaking region R to the LBS server, which then returns the set of objects $\{b, c, d\}$ that are the nearest neighbors of at least one point in R. Finally, among $\{b, c, d\}$, the system finds out the true nearest neighbor b of query location l. Throughout this query processing procedure, the LBS server knows only the region R in which the user is located, not the exact location l. In the literature, a variety of cloaking algorithms based on snapshot user locations have been developed for different privacy metrics (e.g., [13], [14], [24], [26]).

In this paper, we identify and address three new issues concerning the location cloaking approach. We first show that the representation of a cloaking region has an impact on the result superset size of the region-based query. In general, a small result superset is preferred for saving the cost of data transmission and reducing the workload of the result refinement process (especially if this process is implemented on the mobile client). We find that, given a privacy requirement, representing the cloaking region with a circle generally leads to a smaller result superset than using other shapes.

Second, we consider the location cloaking problem for continuous LBS queries. In such scenarios, trace analysis attacks are possible by linking historical cloaking regions with user mobility patterns. Assume that in our previous example, the user issues a second query at location l' with a cloaking region R' (see Figure 1b). If the LBS server somehow learns the user's maximum possible moving speed v_m, the server can draw a region R^e (the shaded area in Figure 1b) expanded from the last cloaking region R based on v_m and the interval t between the two queries. The server is then able to infer that the user must be located in the intersection area of R^e and R', which degrades the quality of location cloaking and may fail to meet the expected privacy requirement. The cloaking quality will further deteriorate with the analysis of more successive queries and cloaking regions. To address this issue, we develop a mobility-aware location cloaking technique that resists trace analysis attacks. Given that the server observes a cloaking region together with any series of historical cloaking regions, our proposed technique makes equal the derivable probability that the user will be located at any one point within the cloaking region. To achieve this, we leverage the probability theory to control the generation of cloaking regions and design two cloaking algorithms, namely *MaxAccu_Cloak* and *MinComm_Cloak*, based on different performance objectives. *MaxAccu_Cloak* is designed to maximize the accuracy of query results, while *MinComm_Cloak* attempts to reduce the network communication cost.

Finally, we investigate how to evaluate efficiently circular-region-based spatial queries on the LBS server. While the evaluation of circular-region-based range queries is straightforward, we develop an efficient $\mathcal{O}(kM^3)$ algorithm for evaluating circular-region-based kNN queries, where M is the cardinality of the spatial object set. In addition, we present two query processing modes, namely *bulk* and *progressive*, which return query results either all at once or in an incremental manner.

We conduct simulation experiments to evaluate the performance of the proposed location cloaking and query processing algorithms. The results show that the proposed mobility-aware cloaking algorithms outperform an isolated cloaking algorithm by up to 34% in terms of an entropy measure of cloaking quality, without compromising much on query latency or communication cost (sometimes performing even better). Regarding the end-to-end system performance, *MaxAccu_Cloak* results in a very high query accuracy, while *MinComm_Cloak* achieves a good balance between communication cost and query accuracy. When the result superset size is small, the bulk and progressive modes of query progressing perform similarly. For large result sets that require a long time to evaluate and transmit, the progressive mode achieves a shorter user-perceived response time than the bulk mode by parallelizing the query evaluation and result transmission.

The rest of this paper is organized as follows. Section II surveys the related work on location privacy protection and spatial query processing. Section III gives an overview of our system model and location privacy metrics. Section IV studies the representation of cloaking regions, followed by Section V, which presents the mobility-aware location cloaking algorithms. The processing of circular-region-based queries is discussed in Section VI. Section VII experimentally evaluates the proposed location cloaking and query processing algorithms. Finally, Section VIII concludes this paper.

II. Related Work

Location Privacy Protection. There are two main approaches to protecting location privacy in LBS. The first approach relies on a trusted LBS server to restrict access to location data based on rule-based policies [10], [11], [36]. The second category of approaches run a trustworthy agent between the client and the LBS server. Every time the user makes a location-based query, the agent anonymizes the user identity and/or location before forwarding the query to the LBS server [5], [13], [26]. Our study falls into the second category.

Early studies on location privacy protection considered object tracking applications, where a proxy server is employed to collect exact locations from moving clients and to anonymize location data through de-personalization before release. In [5], once a client enters a pre-defined zone, its identity is mixed with all other clients in the same zone. It appears that this idea can be extended to deal with trace analysis attacks by associating each LBS request with a different pseudonym. Unfortunately, this approach may not be effective because historical user locations are highly correlative and, hence, they could be re-linked using trajectory tracking methods (e.g., multi-target tracking [27], [32]) even without knowing any identity [34].

Gruteser and Grunwald [13] proposed to achieve identity anonymity in LBS by spatio-temporal cloaking based on a k-anonymity model, that is, the cloaked location is made indistinguishable from the location information of at least $k-1$ other users. To perform the spatial cloaking, they used a Quad-tree-like algorithm. Gedik and Liu [14], [15] extended this to a personalized k-anonymity model, in which users can specify the parameter k at a per-message level. They also developed a new cloaking algorithm called CliqueCloak. While the above cloaking algorithms need a centralized agent to perform location cloaking, Chow *et al.* [8] proposed a peer-to-peer cloaking algorithm based on information exchanges among mobile clients. Ghinita *et al.* [12] proposed a new location cloaking algorithm called hilbASR, in which all user locations are sorted and grouped by Hilbert space-filling curve ordering. They also applied this algorithm to a distributed environment based on an annotated B+-tree index. In [3], Bettini *et al.* presented a framework to model various background attacks in LBS and discussed defense techniques to guarantee users' anonymity. The PrivacyGrid framework [2] investigated location cloaking based on an l-diversity model. But unlike most existing cloaking algorithms, which considered snapshot user locations only, in this paper we investigate the location cloaking problem for continuous LBS queries. In particular, we focus on trace analysis attacks and propose a new mobility-aware cloaking technique to resist them.

More recently, Xu and Cai [34] developed a trajectory cloaking algorithm that aims to reduce the cloaking area and the frequency of location updates. The idea is to use historical user locations as footprints in performing k-anonymity cloaking. To make this idea work, a user needs to provide the location cloaking agent with a future movement trajectory for each LBS request. In contrast, our proposed mobility-aware cloaking technique does not require future locations for any

LBS request. We aim to prevent an adversary from utilizing historical cloaking regions to degrade the quality of current location cloaking.

Spatial Query Processing. A large body of research has investigated spatial query processing, in particular kNN search. Most kNN search algorithms have focused on disk access methods based on R-tree-like index structures [16]. The branch-and-bound approach is often employed in query evaluation to traverse the index and prune search space. Various query evaluation algorithms differ in terms of the visiting order of index nodes and the metric used to prune search space [17], [28], [33]. Whereas the previous studies investigated the kNN problem for a location point or a line segment only, our recent work has developed an evaluation strategy for rectangular-region-based kNN queries that retrieve the k-nearest neighbors of all possible location points in a rectangular region [20]. We remark that the strategy developed in [20] is based on the fact that a rectangle can be decomposed into a set of straight-line segments. But because such decomposition is infeasible for a circle, the strategy of [20] cannot be extended to evaluate circular-region-based kNN queries. In another related work [6], Cheng *et al.* developed algorithms for evaluating probabilistic queries over imprecise object locations. In contrast, we are interested in using imprecise locations to retrieve result supersets for region-based spatial queries.

Parallel to our work, Mokbel *et al.* [26] and Kalnis *et al.* [24] have investigated both the location cloaking and query processing problems. But our work differs from theirs in several respects. First, like other previous studies [13], [14], [15], the location cloaking algorithms in [24] and [26] account for snapshot user locations only. Neither of them considers continuous queries and trace analysis attacks. In contrast, we focus on how to protect against trace analysis attacks for continuous queries through location cloaking. Second, [24] and [26] did not study the issue of how to represent a cloaking region. In this paper, we show that a circular cloaking region generally leads to a small result superset size, and thus we focus on query processing algorithms for circular regions. Finally, [26] investigated *bulk* query processing for rectangular regions only. Though [24] developed a *bulk* processing algorithm for circular-region-based kNN queries, the algorithm has an exponential time complexity of $\mathcal{O}(M^k)$, where M is the cardinality of the spatial object set. In this paper, we propose a polynomial $\mathcal{O}(kM^3)$ algorithm for circular-region-based kNN queries. Furthermore, we develop a novel *progressive* query processing algorithm, which is favorable to slow mobile networks.

III. System Model and Privacy Metrics

A. System Model

This section describes the system model under our study. We consider mobile clients that are equipped with wireless interfaces to communicate with the Internet. We assume that mobile clients are location-aware, that is, they are able to position their locations at any time (e.g., using GPS or other client-based positioning techniques [31]). The users of mobile clients are interested in querying public spatial objects (e.g., hotels, restaurants, gas stations, etc.) related to their current locations.[1] We consider two types of location-based spatial queries. A *range query*, specified with the user's current location l and a distance d_r, retrieves all the objects lying in the circle centered at l with radius d_r. A *kNN query*, specified with the user's current location l and a parameter k, retrieves the k nearest objects to l.

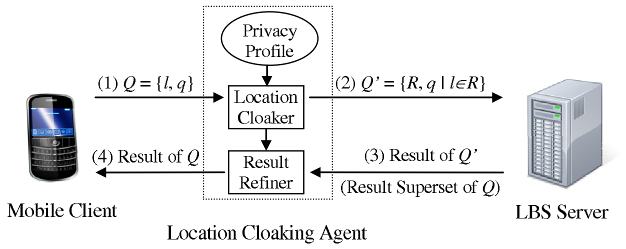


Fig. 2. System Architecture

Figure 2 illustrates the procedure for processing a location-based query. After the user issues such a query, the mobile client sends the query $Q = \{l, q\}$, where l is the current location and q includes other query parameter(s), to a location cloaking agent. The cloaking agent then cloaks the location l into a region R ($l \in R$) based on the user's privacy requirement, and forwards the modified query $Q' = \{R, q\}$ to the LBS server. The LBS server evaluates Q' and returns the result of Q' to the cloaking agent. Since the result of Q' is a superset of the result of Q, the cloaking agent refines the result of Q' to obtain the exact result of Q and finally returns it to the mobile client. In this procedure, we focus on two performance objectives: (1) to optimize the quality of location cloaking with respect to trace analysis attacks while satisfying the user-specified privacy requirement, and (2) to make the size of the result of Q' as small as possible for saving the cost of data transmission and the workload of the cloaking agent in downloading and refining it.

We remark that in the system architecture, the location cloaking agent runs between the mobile client and the LBS server. It may be implemented on an Internet-resident proxy or incorporated into the mobile client. These two solutions have different performance tradeoffs. The first proxy-based solution greatly alleviates the workload of the mobile client by delegating the tasks of location cloaking and result refinement to the resource-richer proxy. But implementing the proxy-based solution is not cost free. First, the connection between the mobile client and the proxy has to be secured to prevent disclosure of location data over the network transmission (e.g., by applying proper encryption and authentication protocols), which incurs extra processing overhead at the mobile client. These measures are not needed in the second client-based solution. Second, since the proxy owns the private information about mobile users (including their privacy preferences as well as current and historical locations), more security risks would be introduced owing to the presence of the proxy. The proxy can become a new target of attacks and a potential

[1]It is noted that "objects" and "mobile users" are different concepts in this paper: "objects" refer to spatial objects (such as hotels and restaurants) to be queried by LBS requests made by "mobile users."

performance bottleneck. A system administrator can determine where to implement the location cloaking agent by taking into consideration the bandwidth budget, client capabilities, and security requirements.

Yet, regardless of which solution the system adopts, the following issues arising from the location cloaking approach deserve our investigation: (1) how to represent cloaking regions in terms of shape such that the result size of the region-based query Q' is minimized (Section IV); (2) how to effectively perform location cloaking on the location cloaking agent so that the cloaking quality is optimized against trace analysis attacks (Section V); and (3) how to efficiently evaluate region-based spatial queries (on the LBS server) to reduce the query response time (Section VI). It is worth noting that the techniques proposed in this paper are beneficial to both proxy-based and client-based solutions.

B. Privacy Metrics

We employ an intuitive privacy metric for location anonymity, that is, the area of the cloaking region (or briefly, *the cloaking area*). A user can specifies a minimum acceptable cloaking area for each query. For example, a user can set the minimum acceptable cloaking area to one square mile. To consider resistance to trace analysis attacks, the quality of location cloaking is measured by *entropy*, a well-known metric for quantifying the amount of uncertainty in information theory [1]. Suppose it can be derived that the probability density function for the user to be at location l in cloaking region R is $p(l)$, the entropy is then defined by

$$-\int_{l\in R} p(l)\ln p(l)\,\mathrm{d}l. \tag{1}$$

Given a cloaking region, entropy will be zero if it is derived that the user is at some location with 100% probability. Entropy will increase if the user location is more uncertain, and will be maximized when the derivable probability for the user to be at any location in the region is equal.

IV. Representation of Cloaking Regions

In this section, we study the representation of cloaking regions. Given a cloaking area, we are interested in finding out how to represent the cloaking region in terms of shape such that the result size of the region-based query is minimized. It is worth noting that the representation of a cloaking region is independent of the issue of maximizing entropy in location cloaking. For any cloaking region of a given area, irrespective of its shape, entropy is maximized when the derivable probability for the user to be at any location in the region is uniform across that region.

Consider a region-based kNN query that retrieves the k nearest neighbors of all the points in the region. The following theorem shows that the result of a region-based kNN query should include all objects in the region as well as the kNNs of the points on the perimeter of the region.

Theorem 1: An object o is in the kNN results of region R if and only if: i) $o \in R$, or ii) o is in the kNN results of some point on the perimeter of R.

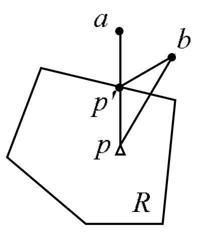
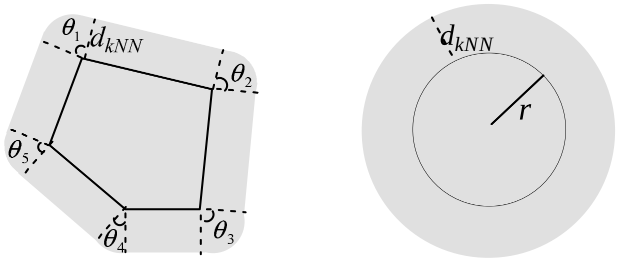


Fig. 3. Proof of Theorem 1



(a) Convex Region (b) Circular Region

Fig. 4. Solution Space of a Region-based kNN Query

Proof: Obviously any object inside R is the NN of the same point it occupies. Next, we use a proof-by-contradiction approach to show that if an object outside R is the i-th NN ($i \le k$) of a point inside R, this object must be in the iNN results (and hence the kNN results) of some point on the perimeter of R.

As shown in Figure 3, suppose that object a is the i-th NN of point p inside R. Assume on the contrary that a is not in the iNN results of any point on the perimeter of R. Consider the intersecting point p' of the segment $\overline{pa}$ and the perimeter of R. It follows that a is not in the iNN results of p'. Thus, the iNN results of p' and p overlap by at most $i-1$ objects. As a result, there must exist an object b in the iNN results of p' that is not in the iNN results of p. This implies $|p'b| \le |p'a|$. Thus, we have $|pb| < |p'b| + |pp'| \le |p'a| + |pp'| = |pa|$. This means that b is closer to p than a, which contradicts the hypothesis that a is the i-th NN of p, and that b is not in the iNN results of p. Hence, the theorem is proven. □

To simplify our analysis, we follow the previous work of [4], [33] and assume that the spatial objects to be queried are uniformly distributed in the search space. Denote by ρ the object density. According to [4], [33], the average distance between a query point and its k-th NN is given by

$$d_{kNN} = \sqrt{\frac{k}{\pi\rho}}. \tag{2}$$

Following Theorem 1, the solution space for a region-based kNN query can be approximated by the area extended from the query region by a distance of d_{kNN} (see the shaded areas in Figure 4).[2] Thus, we estimate the size of the kNN results $|\mathcal{R}_{kNN}|$ by the number of objects lying in the approximated solution space. Let A and P respectively be the area and the perimeter length of the query region. For a general convex

[2]Note that this is neither a necessary nor a sufficient condition for an object to be part of the kNN results.

region (Figure 4a), we obtain

$$\begin{aligned}|\mathcal{R}_{kNN}| &\approx (A + P \cdot d_{kNN} + \sum_i \frac{1}{2}\theta_i d_{kNN}^2) \cdot \rho \\ &= A \cdot \rho + P \cdot \sqrt{\frac{k\rho}{\pi}} + k. \qquad (3)\end{aligned}$$

Similarly, for a region-based range query, we can estimate the size of its query results as

$$|\mathcal{R}_{range}| = (A + P \cdot d_r + \pi d_r^2) \cdot \rho, \qquad (4)$$

where d_r is the radius of the query range.

Theorem 2: Comparing different region shapes of the same area A, a circle gives the smallest value for both $|\mathcal{R}_{kNN}|$ and $|\mathcal{R}_{range}|$ in Eqs. (3) and (4).

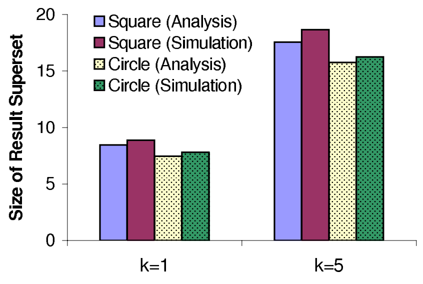
Proof: Given the same value of area A, from Eq. (3) (or (4)), the relative value of $|\mathcal{R}_{kNN}|$ (or $|\mathcal{R}_{range}|$) is determined by the perimeter length P. It is well known that a circle (see Figure 4b) has the shortest perimeter under a fixed area. $\square$

Theorem 2 implies that given a cloaking area, a circular region is expected to give the smallest result set for both range and kNN queries under a uniform distribution of spatial objects. Figure 5a compares the result sizes obtained by using both the circular and square cloaking regions of area 10^{-5} for kNN queries on a dataset containing 300,000 objects randomly distributed in a unit space. The simulation results in Figure 5a are the average of 1,000 random queries on the dataset;[3] the analytical results are computed using Eq. (3). It can be seen that the analytical results well match the simulation results, and the average result size given by a circular cloaking region is less than that given by a square region of the same area. We also compare circular and square cloaking regions for a real California dataset where the objects are not uniformly distributed (see Section VII for more details about this dataset). As shown in Figure 5b, a circular cloaking region again leads to a smaller result size than a square cloaking region. Thus, in the rest of this paper we will use circles to represent cloaking regions.

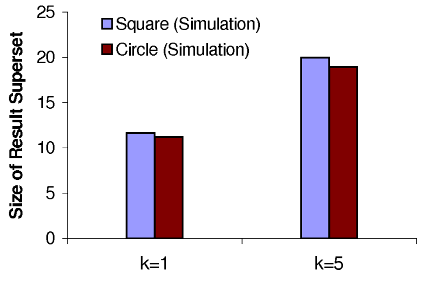
V. Mobility-Aware Location Cloaking

We now study how to generate circular cloaking regions based on privacy requirements. Under isolated cloaking, for each query with a cloaking area requirement A_{min}, a circle with radius $\sqrt{A_{min}/\pi}$ covering the user location l is randomly generated to serve as the cloaking region. But this scheme is vulnerable to trace analysis attacks. As discussed in the Introduction, by correlating the query trace and the mobility pattern, the LBS server (adversary) is likely to derive the probabilities of user locations in the cloaking region. This leads to a significant degradation of the quality of location cloaking. In this section, we develop an optimal mobility-aware cloaking technique that works as follows. For the first query, a random cloaking region is generated. For each subsequent query, we control the generation of cloaking regions

[3]To allow for fair comparison, both the circular and square cloaking regions are formed with the query point at the centroid.



(a) Uniform Dataset



(b) California Dataset

Fig. 5. Size of the kNN Results

such that the cloaking quality (in terms of entropy as defined in Eq. (1)) is maximized, that is, given that the server observes a cloaking region together with any series of historical cloaking regions, the derivable probability for the user to be located at any point in the cloaking region should be equal.

A. Problem Formulation

We consider a general user mobility pattern that is known to the mobile client. We assume, in the worst case, that the adversary also knows the user mobility pattern and thus has the potential to conduct trace analysis attacks. The user mobility pattern may be built by the adversary based on traces (of non-privacy-conscious users of the same type) [37] or mobility scenarios (e.g., the random walk model is good to model the mobility pattern of pedestrians in small-scale urban areas) [22].

Denote by O the center of the old cloaking region produced for the last query (with a radius of $r = \sqrt{A_{min}/\pi}$). Let $u(x)$ be the probability density function of the new user location being distance x away from O at the time of the new query, assuming that the user location is uniformly distributed in the old cloaking region. It follows that

$$\int_0^D u(x)\mathrm{d}x = 1, \qquad (5)$$

where D is the farthest possible distance that the user can travel since the last query, $D = \min\{y \mid u(x) = 0, \forall x \geq y\}$.

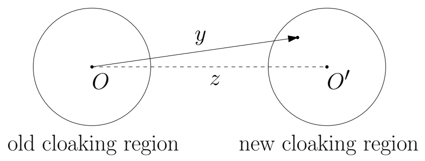


Fig. 6. Old and New Cloaking Regions

Assume that the user is currently distance y away from O (see Fig. 6). Denote by O' the center of the new cloaking region and z the distance between O' and O. Define $p(z|y)$ as the probability density function of z given y. In order for the new cloaking region to cover the user, O' must be within a distance of r from the user's current location. Thus, we have

$$\int_{\max\{0,y-r\}}^{\min\{D-r,y+r\}} p(z|y)\mathrm{d}z = 1. \tag{6}$$

Essentially, the location cloaking is to determine the $p(z|y)$ function with the objective of maximizing entropy, i.e., the user is equally likely to be at any point in the new cloaking region. To mathematically characterize this objective, we define $q(y|z)$ as the probability density function of the new user location being distance y away from the old center O given that the center O' of the new region is distance z away from O. Since the user is equally likely to be at any point in the cloaking region, $q(y|z)$ is proportional to the length of the arc (centered at O and with radius y) overlapping with the new cloaking region (as indicated by the bold arc in Figure 7b; hereafter referred to as the *overlapping arc length*). Below we analyze the value of $q(y|z)$ under maximum-entropy cloaking:

- Assume $z \geq r$. In Figure 7a (i.e., $0 \leq y \leq z-r$) and Figure 7c (i.e., $z+r \leq y$), the overlapping arc length is 0. In Figure 7b (i.e., $z-r \leq y \leq z+r$), the overlapping arc length is $2\alpha y = 2 \cdot \arccos\frac{y^2+z^2-r^2}{2yz} \cdot y$. Therefore, after normalizing to the integration over all possible y values within the conditioned range in which $q(y|z)$ is not zero, we obtain

$$q(y|z) = \begin{cases} \frac{2\cdot\arccos\frac{y^2+z^2-r^2}{2yz}\cdot y}{\int_{z-r}^{z+r} 2\cdot\arccos\frac{y^2+z^2-r^2}{2yz}\cdot y\mathrm{d}y} & \text{if } z-r \leq y \leq z+r, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

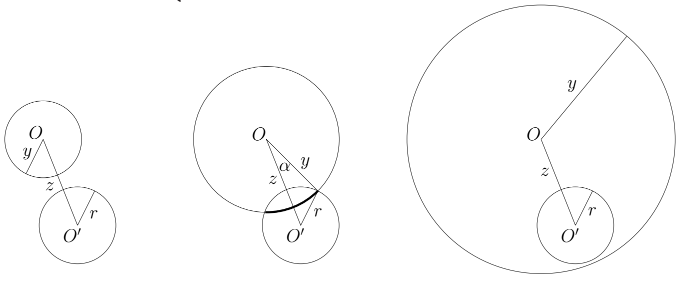


Fig. 7. $z \geq r$

- Assume $z < r$. In Figure 8a (i.e., $0 \leq y \leq r-z$), the overlapping arc length is $2\pi y$. In Figure 8b (i.e., $r-z \leq y \leq z+r$), the overlapping arc length is $2\alpha y = 2 \cdot \arccos\frac{y^2+z^2-r^2}{2yz} \cdot y$. In Figure 8c (i.e., $z+r \leq y$), the overlapping arc length is 0. Therefore, after normalization we obtain

$$q(y|z) = \begin{cases} \frac{2\pi y}{\pi(r-z)^2+\int_{r-z}^{z+r} 2\cdot\arccos\frac{y^2+z^2-r^2}{2yz}\cdot y\mathrm{d}y} & \text{if } 0 \leq y \leq r-z, \\ \frac{2\cdot\arccos\frac{y^2+z^2-r^2}{2yz}\cdot y}{\pi(r-z)^2+\int_{r-z}^{z+r} 2\cdot\arccos\frac{y^2+z^2-r^2}{2yz}\cdot y\mathrm{d}y} & \text{if } r-z \leq y \leq z+r, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Having known $q(y|z)$ as in Eqs. (7) and (8) under maximum-entropy cloaking, our problem becomes to determine $p(z|y)$ given $q(y|z)$. Note that the relation between

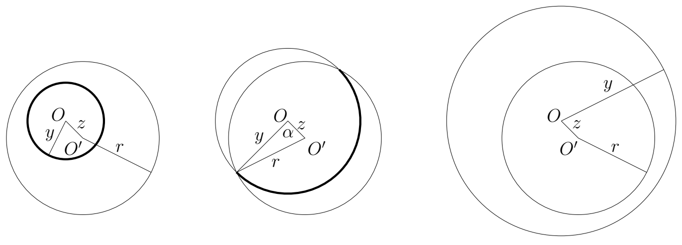


Fig. 8. $z < r$

$p(z|y)$ and $q(y|z)$ can be established by the Bayes' rule, that is,

$$q(y|z) = \frac{p(z|y)\cdot u(y)}{\int_{\max\{0,x-r\}}^{\min\{R-r,x+r\}} p(z|x)\cdot u(x)\mathrm{d}x}. \tag{9}$$

We will discuss how to solve $p(z|y)$ from Eqs. (6), (7), (8), and (9) in the next subsection.

We remark that in our approach, only the last cloaking region is needed to generate a new maximum-entropy cloaking region. The following theorem shows the correctness of this approach.

Theorem 3: Given that the server observes the new cloaking region and all old cloaking regions, the user is equally likely to be at any point in the new cloaking region.

Proof: Denote by (x_n, y_n) the user's location and G_n the cloaking region at the time of the n-th query. Define $p(x_n, y_n)$ as the probability density function of the user being at location (x_n, y_n) in region G_n. We prove the claim by induction: given that the server observes G_1 through G_n, for any two points $(x_n, y_n), (x'_n, y'_n)$ in G_n, $p(x_n, y_n) = p(x'_n, y'_n)$.

First, it is obvious that $p(x_1, y_1) = p(x'_1, y'_1)$ for $n = 1$ since the first cloaking region is randomly generated.

Next, we assume that the claim holds for some n ($n \geq 1$). Then, $p(x_n, y_n)$ is a constant $\frac{1}{\pi r^2}$. We are going to prove that the claim also holds for $n+1$. Given G_1 through G_{n+1}, for any two points (x_{n+1}, y_{n+1}), (x'_{n+1}, y'_{n+1}) in G_{n+1}, we have

$$\begin{aligned} & p(x_{n+1}, y_{n+1}) \\ &= \iint_{G_n} p((x_{n+1}, y_{n+1}), (x_n, y_n))\,\mathrm{d}x_n\mathrm{d}y_n \\ &= \iint_{G_n} p((x_{n+1}, y_{n+1})|(x_n, y_n))\cdot p(x_n, y_n)\,\mathrm{d}x_n\mathrm{d}y_n \\ &= \iint_{G_n} p((x_{n+1}, y_{n+1})|(x_n, y_n))\cdot\frac{1}{\pi r^2}\,\mathrm{d}x_n\mathrm{d}y_n. \end{aligned} \tag{10}$$

Satisfying Eqs. (7) and (8), our cloaking approach ensures

$$\begin{aligned} & \iint_{G_n} p((x_{n+1}, y_{n+1})|(x_n, y_n))\cdot\frac{1}{\pi r^2}\,\mathrm{d}x_n\mathrm{d}y_n \\ &= \iint_{G_n} p((x'_{n+1}, y'_{n+1})|(x_n, y_n))\cdot\frac{1}{\pi r^2}\,\mathrm{d}x_n\mathrm{d}y_n. \end{aligned}$$

Therefore, Eq. (10) can be rewritten as

$$\begin{aligned} & p(x_{n+1}, y_{n+1}) \\ &= \iint_{G_n} p((x'_{n+1}, y'_{n+1})|(x_n, y_n))\cdot\frac{1}{\pi r^2}\,\mathrm{d}x_n\mathrm{d}y_n \\ &= \iint_{G_n} p((x'_{n+1}, y'_{n+1}), (x_n, y_n))\,\mathrm{d}x_n\mathrm{d}y_n \\ &= p(x'_{n+1}, y'_{n+1}). \end{aligned}$$

Hence, the theorem follows. □

B. Problem Discretization

Now what we are left is to solve $p(z|y)$ from Eqs. (6), (7), (8), and (9). Unfortunately, a closed-form solution is difficult to obtain. In this section, we present a discretization-based numerical method. We divide the plane into a set of rings of sufficiently small width Δ. The rings are centered at O. As shown in Figure 9, ring 1 is enclosed by a circle centered at O with a radius of Δ, i.e., ring 1 contains all points that are within distance Δ from O. For each $i > 1$, ring i is enclosed by two circles centered at O with radii of $(i-1)\Delta$ and $i\Delta$ respectively, i.e., ring i includes all points that are $(i-1)\Delta$ to $i\Delta$ away from O.

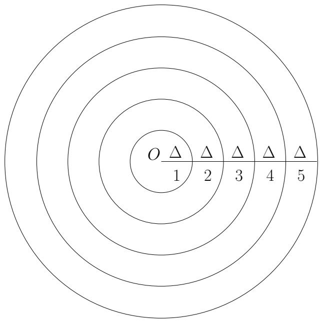


Fig. 9. A Set of Rings

Without loss of generality, we assume that the radius of a region $r = K\Delta$, and the farthest possible distance that the user can travel since the last query $D = L\Delta$, where K and L are integers. Based on the assumption of mobility pattern, the probability $U(i)$ of the new user location being in ring i is given by $U(i) = \int_{(i-1)\Delta}^{i\Delta} u(x)\mathrm{d}x$, and it follows that

$$U(1) + U(2) + \cdots + U(L) = 1.$$

We define $Q(i|j)$ as the probability of the new user location being in ring i given that the center of the new region is in ring j. For ring i, we use the average radius of two enclosing circles (i.e., $(i\Delta + (i-1)\Delta)/2 = (i-1/2)\Delta$) to approximate its distance to O. Thus, following (7) and (8), $Q(i|j)$ should satisfy the following. If $j \geq K$,

$$Q(i|j) = \begin{cases} \dfrac{\arccos \frac{(i-\frac{1}{2})^2+(j-\frac{1}{2})^2-K^2}{2(i-\frac{1}{2})(j-\frac{1}{2})} \cdot (i-\frac{1}{2})}{\sum_{m=j-K+1}^{j+K-1} \arccos \frac{(m-\frac{1}{2})^2+(j-\frac{1}{2})^2-K^2}{2(m-\frac{1}{2})(j-\frac{1}{2})} \cdot (m-\frac{1}{2})} & \text{if } j-K+1 \leq i \leq j+K-1, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

If $j < K$,

$$Q(i|j) = \begin{cases} \dfrac{\pi(i-\frac{1}{2})}{\sum_{m=1}^{K-j} \pi(m-\frac{1}{2}) + \sum_{m=K-j+1}^{j+K-1} \arccos \frac{(m-\frac{1}{2})^2+(j-\frac{1}{2})^2-K^2}{2(m-\frac{1}{2})(j-\frac{1}{2})} \cdot (m-\frac{1}{2})} & \text{if } 1 \leq i \leq K-j, \\ \dfrac{\arccos \frac{(i-\frac{1}{2})^2+(j-\frac{1}{2})^2-K^2}{2(i-\frac{1}{2})(j-\frac{1}{2})} \cdot (i-\frac{1}{2})}{\sum_{m=1}^{K-j} \pi(m-\frac{1}{2}) + \sum_{m=K-j+1}^{j+K-1} \arccos \frac{(m-\frac{1}{2})^2+(j-\frac{1}{2})^2-K^2}{2(m-\frac{1}{2})(j-\frac{1}{2})} \cdot (m-\frac{1}{2})} & \text{if } K-j+1 \leq i \leq j+K-1, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

What we want to find out is $P(j|i)$ — the probability of the center of the new region being in ring j given that the new user location is in ring i. Following (6), the definable probabilities are listed as a matrix in Figure 10. The sum of each row in the matrix equals 1, i.e., $\sum_{j=\max\{1,i-K+1\}}^{\min\{L-K+1,i+K-1\}} P(j|i) = 1$. After discretization, our problem becomes to determine the $P(j|i)$ function given $Q(i|j)$ as in Eqs. (11) and (12).

Following the Bayes' rule, for any i and j,

$$P(j|i) = \frac{Q(i|j)}{U(i)} \cdot \sum_m (P(j|m) \cdot U(m)).$$

Let $v_j = \sum_m (P(j|m) \cdot U(m))$. Thus, we have

$$P(j|i) = \frac{Q(i|j)}{U(i)} \cdot v_j. \tag{13}$$

The matrix we want to compute can be rewritten as shown in Figure 11. Our problem further becomes to find $v_1, v_2, \cdots, v_{L-K+1}$ such that the sum of each row in the matrix equals 1:

$$\begin{cases} \sum_{j=\max\{1,i-K+1\}}^{\min\{L-K+1,i+K-1\}} P(j|i) \\ \quad = \sum_{j=\max\{1,i-K+1\}}^{\min\{L-K+1,i+K-1\}} \frac{Q(i|j)}{U(i)} v_j = 1, & 1 \leq i \leq L, \\ v_j \geq 0, & 1 \leq j \leq L-K+1. \end{cases} \tag{14}$$

C. Practical Cloaking Algorithms

The linear equation set (14) does not always have a feasible solution since the number of equations (L) is more than the number of variables ($L-K+1$). Thus, we have to relax some of the constraints. To this end, we allow the sum of each row in the matrix to be less than 1, i.e., user locations may not be cloaked for some queries. Thus, the set of linear equations is relaxed to

$$\begin{cases} \sum_{j=\max\{1,i-K+1\}}^{\min\{L-K+1,i+K-1\}} \frac{Q(i|j)}{U(i)} v_j \leq 1, & 1 \leq i \leq L, \\ v_j \geq 0, & 1 \leq j \leq L-K+1. \end{cases} \tag{15}$$

To protect location privacy, the queries whose locations are not cloaked will be *blocked* and are not sent to the LBS server. To answer blocked queries, we propose to cache the last result superset for potential reuse. Note that the amount of cache memory needed is minimal since only the result superset for the last query is cached. In fact, if a new query is issued from ring $i \leq K$ (i.e., from the old cloaking region; called *inner query*), we should block it in order to save communication cost as the precise results can be computed from the cached last result superset. Thus, all the inner queries are blocked, except those sent to the server to achieve optimal cloaking. On the other hand, if a query issued from ring $i > K$ (called *outer query*) is blocked, the client might obtain an inaccurate query result based on the cached last result superset and, hence, the accuracy of query results might be sacrificed. Therefore, we formulate two linear programs with different objective functions:

MaxAccu_Cloak:

$$\text{minimize} \quad (1 - \sum_{i=K+1}^{L} \sum_{j=\max\{1,i-K+1\}}^{\min\{L-K+1,i+K-1\}} \frac{Q(i|j)}{U(i)} v_j) \tag{16}$$

MinComm_Cloak:

$$\text{minimize} \quad (1 - \sum_{i=K+1}^{L} \sum_{j=\max\{1,i-K+1\}}^{\min\{L-K+1,i+K-1\}} \frac{Q(i|j)}{U(i)} v_j) \\ -(1 - \sum_{i=1}^{K} \sum_{j=\max\{1,i-K+1\}}^{\min\{L-K+1,i+K-1\}} \frac{Q(i|j)}{U(i)} v_j) \tag{17}$$

$$\begin{pmatrix}
P(1|1) & P(2|1) & \cdots & P(K|1) & & & & & & & & \\
P(1|2) & P(2|2) & \cdots & P(K|2) & P(K+1|2) & & & & & & & \\
\cdots & \cdots & \cdots & \cdots & \cdots & & & & & & & \\
P(1|K) & P(2|K) & \cdots & \cdots & \cdots & P(2K-1|K) & & & & & & \\
 & P(2|K+1) & \cdots & \cdots & \cdots & P(2K-1|K+1) & P(2K|K+1) & & & & & \\
 & & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & & & & \\
 & & & & & & \cdots & \cdots & \cdots & & \cdots & \\
 & & & & & & & P(L-3K+3|L-2K+2) & P(L-3K+4|L-2K+2) & \cdots & P(L-K|L-2K+2) & P(L-K+1|L-2K+2) \\
 & & & & & & & & P(L-3K+4|L-2K+3) & \cdots & P(L-K|L-2K+3) & P(L-K+1|L-2K+3) \\
 & & & & & & & & & \cdots & \cdots & \cdots \\
 & & & & & & & & & & P(L-K|L-1) & P(L-K+1|L-1) \\
 & & & & & & & & & & & P(L-K+1|L)
\end{pmatrix}$$

Fig. 10. Probability Matrix (showing the upper-left corner and lower-right corner only)

$$\begin{pmatrix}
\frac{Q(1|1)}{U(1)}v_1 & \frac{Q(1|2)}{U(1)}v_2 & \cdots & \frac{Q(1|K)}{U(1)}v_K & & & & & & & & \\
\frac{Q(2|1)}{U(2)}v_1 & \frac{Q(2|2)}{U(2)}v_2 & \cdots & \frac{Q(2|K)}{U(2)}v_K & \frac{Q(2|K+1)}{U(2)}v_{K+1} & & & & & & & \\
\cdots & \cdots & \cdots & \cdots & \cdots & & & & & & & \\
\frac{Q(K|1)}{U(K)}v_1 & \frac{Q(K|2)}{U(K)}v_2 & \cdots & \cdots & \cdots & \frac{Q(K|2K-1)}{U(K)}v_{2K-1} & & & & & & \\
 & \frac{Q(K+1|2)}{U(K+1)}v_2 & \cdots & \cdots & \cdots & \frac{Q(K+1|2K-1)}{U(K+1)}v_{2K-1} & \frac{Q(K+1|2K)}{U(K+1)}v_{2K} & & & & & \\
 & & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & & & & \\
 & & & & & & \cdots & \cdots & \cdots & \cdots & & \\
 & & & & & & & \frac{Q(L-2K+2|L-3K+3)}{U(L-2K+2)}v_{L-3K+3} & \frac{Q(L-2K+2|L-3K+4)}{U(L-2K+2)}v_{L-3K+4} & \cdots & \frac{Q(L-2K+2|L-K)}{U(L-2K+2)}v_{L-K} & \frac{Q(L-2K+2|L-K+1)}{U(L-2K+2)}v_{L-K+1} \\
 & & & & & & & & \frac{Q(L-2K+3|L-3K+4)}{U(L-2K+3)}v_{L-3K+4} & \cdots & \frac{Q(L-2K+3|L-K)}{U(L-2K+3)}v_{L-K} & \frac{Q(L-2K+3|L-K+1)}{U(L-2K+3)}v_{L-K+1} \\
 & & & & & & & & & \cdots & \cdots & \cdots \\
 & & & & & & & & & & \frac{Q(L-1|L-K)}{U(L-1)}v_{L-K} & \frac{Q(L-1|L-K+1)}{U(L-1)}v_{L-K+1} \\
 & & & & & & & & & & & \frac{Q(L|L-K+1)}{U(L)}v_{L-K+1}
\end{pmatrix}$$

Fig. 11. Rewritten Probability Matrix (showing the upper-left corner and lower-right corner only)

The first objective function MaxAccu_Cloak attempts to minimize the outer query blocking probability for $i = K+1, K+2, \cdots, L$, thereby maximizing the query accuracy. In contrast, the second MinComm_Cloak trades query accuracy for communication cost. It also aims to maximize the inner query blocking probability for $i = 1, 2, \cdots, K$ to increase the result reuse rate and save remote queries. The performance of these two cloaking algorithms will be evaluated by experiments in Section VII-C.

On solving the linear program and obtaining $v_1, v_2, \cdots, v_{L-K+1}$, we can compute $P(j|i)$'s using Eq. (13). Then, given a new query with user location in ring i, the query has a probability of $(1 - \sum_j P(j|i))$ to be blocked. If the query is not blocked, the distance between the new cloaking region and the old cloaking region can be randomly generated based on the probabilities of $P(j|i)$'s. Given the distance, the center of the new cloaking region can be randomly generated on the corresponding arc. A summary of the optimal mobile-aware cloaking technique is described in Algorithm 1.

Algorithm 1 Overview of Mobility-Aware Location Cloaking

```
Input: mobility pattern $U(\cdot)$, old cloaking region centered at
       $O$, new user location in ring $i$
Output: the center of the new cloaking region
Procedure:
1: compute $Q(i|j)$'s using Eqs. (11) and (12)
2: construct a linear program formed by Eqs. (15) and (16)
   (for MaxAccu_Cloak), or Eqs. (15) and (17) (for Min-
   Comm_Cloak), depending on the performance objective
3: solve the linear program to get $v_j$'s
4: compute $P(j|i)$'s using Eq. (13)
5: determine whether the query is blocked based on the
   probability of $(1 - \sum_j P(j|i))$
6: if the query is not blocked then
7:     generate the distance of the new cloaking region from
       $O$ by following $P(j|i)$'s
8:     randomly generate the center of the new region on the
       corresponding arc
```

VI. Region-based Query Processing

This section discusses how to process circular-region-based queries on the server side. The evaluation of a region-based range query is straightforward since it is still a range query (with an extended range), which simply retrieves all the objects within the spatial range. Thus, we focus on the evaluation of circular-region-based kNN queries (hereafter called kCRNN queries) in this section. Following Theorem 1, the results of a kCRNN query include all the objects in the circular region and the kNNs of the points on the perimeter of the circle (denoted by Ω).

In the following, we propose two kCRNN processing algorithms: a *bulk* algorithm that generates the query results all at once at the end of query evaluation and a *progressive* algorithm that produces the results incrementally during query evaluation.

A. Bulk Query Processing of kCRNN

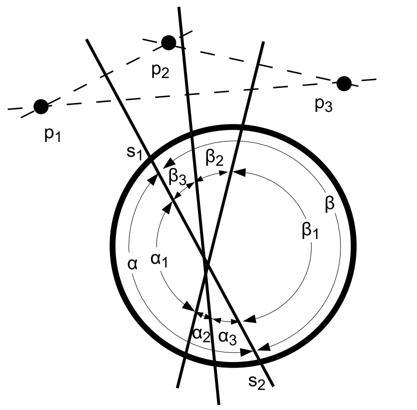
Denote the set of spatial objects by $\{p_1, p_2, \cdots, p_M\}$. The basic idea is to scan the objects one by one, and during each scan we maintain the set of arcs on Ω for each object to which this object is the 1st, 2nd, $\cdots$, and k-th NN. The example shown in Figure 12a is used to illustrate the idea for a 2CRNN query. Suppose that there are three objects p_1, p_2, and p_3. Initially p_1 is the 1st NN to the circumference Ω. Then, p_2 is scanned, and the perpendicular bisector of p_1 and p_2 splits Ω into arcs α and β. As a result, p_2 takes over p_1 to be the 1st NN to β — p_1 is the 1st NN to α and the 2nd NN to β; p_2 is the 1st NN to β and the 2nd NN to α. Next, p_3 is scanned and we check it against p_1 and p_2. The perpendicular bisectors further split α into α_1, α_2, and α_3, and split β into β_1, β_2, and β_3. Now p_3 takes over p_1 to be the 1st NN for α_3 and the 2nd NN for β_2; p_3 takes over p_2 to be the 2nd NN for α_2 and the 1st NN for β_1. In general, when object p_i is scanned, initially we assume that p_i is farther away from any arc than any candidate kCRNN result. Afterwards, we check p_i against each p_j in the candidate kCRNN result set. Given a p_j, the perpendicular bisector of p_j and p_i splits the existing arcs at two points at most. For each of the arcs located on the p_i side of the perpendicular bisector, p_j moves backward in the kNN list (e.g., the 2nd NN becomes the 3rd NN), while p_i advances in the kNN list (e.g., the 3rd NN becomes the 2nd NN). After each scan, those objects which have at least one l-th-NN arc ($l \leq k$) constitute the candidate set of kCRNN results; and if the order of some object p to an arc exceeds k, this arc is removed for p. The algorithm works by scanning the entire set of objects to obtain the final kCRNN results of Ω. Recall that the results of a kCRNN query also include all the objects in the circular region. Thus, when scanning the objects, the algorithm also checks whether they are in the circular region and if so, includes them in the final kCRNN results.

Furthermore, in order to speed up the convergence of the kCRNN candidate set, we sort the objects and apply a heuristic (Heuristic 1) to scan the objects closest to Ω first because they are most likely to appear in the final kCRNN results.

Heuristic 1: The objects are sorted and scanned in the increasing order of their minimum distances to Ω. And those objects whose minimum distances to Ω are $2r$ (r is the radius of Ω) farther than the k-th NN of some arc are removed from scanning.

The second statement of Heuristic 1 sets a stop condition for the scan, because those objects that are $2r$ farther from Ω than the current k-th NN of some arc must be farther away from any arc of Ω than the current kNNs of this arc. This can be explained in Figure 12b for a 3CRNN query. For the moment, p_1, p_2, and p_3 are the 3NNs of an arc of Ω and p_1 is the third NN. The minimum distance from p_1 to Ω, denoted by max_kNN_dist, is used to prune faraway objects in future search. More specifically, if any object is $2r + max\_kNN\_dist$ away from any point on Ω (i.e., outside the outermost circle in Figure 12b), the object does not need to be scanned since it must be farther away from any arc of Ω than p_1, p_2, and p_3.

The complete query processing algorithm is described in



After scanning p_1

	1NN	2NN
p_1	Ω	

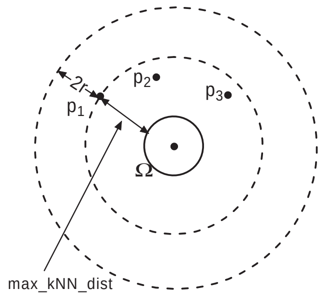
After scanning p_2

	1NN	2NN
p_1	α	β
p_2	β	α

After scanning p_3

	1NN	2NN
p_1	α_1, α_2	α_3, β_3
p_2	β_2, β_3	α_1, β_1
p_3	β_1, α_3	α_2, β_2

(a) Query Processing



(b) Stop Condition

Fig. 12. Finding kCRNN Results

Algorithm 2, where the data structure $\mathcal{F}(p_i, \widehat{ab})$ maintains the order of object p_i to arc $\widehat{ab}$. We call it a *bulk* algorithm as all the candidate kCRNN results are finalized at the end of query evaluation.

We now analyze the complexity of Algorithm 2. Given a kCRNN query with M objects, the while loop iterates through at most M scans. Since each scan increases the number of arcs by $2M$ in the worst case, the total number of arcs for all candidate kCRNN results is bounded by $\mathcal{O}(M^2)$. Each arc can appear in the arc sets of at most k objects. Thus, the worst-case storage complexity is $\mathcal{O}(kM^2)$. For the time complexity, each $\mathcal{F}(p_j, \widehat{ab})$ entry may be updated at most M times, once in each scan. Hence, the worst-case time complexity is $\mathcal{O}(kM^3)$. Nevertheless, in practice the cost would be far less because the candidate set of kCRNN results is normally not large and the scanning may terminate early with the stop condition proposed in Heuristic 1.

B. Progressive Query Processing of kCRNN

The bulk query processing algorithm generates the kCRNN results at the end of query evaluation. Therefore, the server cannot start transmitting the results to the client until the end of query evaluation. We now propose an alternative *progressive* query processing algorithm to parallelize the query evaluation and result transmission.

The idea is to determine whether an object will be a final kCRNN result earlier. The query progressing procedure remains the same as in Algorithm 2 except that i) any object in $in\_circle\_results$ is immediately returned to the client when it is scanned, and ii) after the scan of each object, we add a checking procedure (see Algorithm 3). We randomly pick an

Algorithm 2 Bulk Query Processing of kCRNN

```
Input: query circle $\Omega$ with radius $r$, spatial object set $\mathcal{S}$
Output: the $k$CRNN results of $\Omega$
Procedure:
1:  enqueue the objects in $\mathcal{S}$ into a priority queue in increasing order of their (minimum) distances to $\Omega$ (denoted by $minDist(p_i, \Omega)$)
2:  dequeue the first object $p_1$
3:  $\mathcal{F}(p_1, \Omega)$ := 1 // $\mathcal{F}(p_i, \widehat{ab})$ maintains the order of $p_i$ to $\widehat{ab}$
4:  $cand\_kCRNN\_results$ := $\{p_1\}$
5:  $max\_kNN\_dist$ := $\infty$
6:  dequeue the next object $p_i$
7:  while $minDist(p_i, \Omega) < 2r + max\_kNN\_dist$ do
8:      if $p_i$ is inside $\Omega$ then
9:          $in\_circle\_results$ := $in\_circle\_results \cup p_i$
10:     initialize $\mathcal{F}(p_i, \widehat{ab})$ := $|cand\_kCRNN\_results| + 1$
        for any arc $\widehat{ab}$   // initially assuming $p_i$ is farther
                                     // than any candidate $k$CRNN result
11:     for each object $p_j$ in $cand\_kCRNN\_result$ do
12:         split existing arcs by $\perp p_i p_j$ — the perpendicular
            bisector of $p_i$ and $p_j$
13:         for any arc $\widehat{ab}$ located on $p_i$ side of $\perp p_i p_j$ do
14:             if entry $\mathcal{F}(p_j, \widehat{ab})$ exists then $\mathcal{F}(p_j, \widehat{ab})$++
                                     // move $p_j$ backward in the $k$NN list
15:             $\mathcal{F}(p_i, \widehat{ab})$−−  // move $p_i$ forward in the $k$NN list
16:     let $S$ be the set of scanned objects including $p_i$;
        $cand\_kCRNN\_results$ :=
        $\{p \mid p \in S, \exists$ an arc $\widehat{ab}, \mathcal{F}(p, \widehat{ab}) \le k\}$
17:     remove $\mathcal{F}()$ entries with $\mathcal{F}(p, \widehat{ab}) > k$ for $p \in S$
18:     $max\_kNN\_dist$ := $\min\{max\_kNN\_dist,$
        $\min_{\forall \widehat{ab}}\{minDist(p, \Omega) \mid p$ is the $k$-th NN of arc $\widehat{ab}\}\}$
19:     dequeue the next object $p_i$
20: return $in\_circle\_results \cup cand\_kNN\_results$ as the final results
```

Algorithm 3 Checking Procedure in Progressive Query Processing of kCRNN

```
Input: $\mathcal{F}(p_i, \widehat{ab})$ entries, the queue of unscanned objects
Output: the $k$NN results of a check point
Procedure:  // this procedure is added to between lines
            // 17 and 18 of Algorithm 2
1:  randomly select an unchecked split point $s$ as the check point
2:  retrieve the tentative $k$NN results of $s$: $\{p \mid \mathcal{F}(p, \widehat{ab}) \le k, s \in \widehat{ab}\}$
3:  $current\_kNN\_distance$ := distance of the current $k$-th NN
4:  dequeue the next object $p_i$
5:  while $minDist(p_i, \Omega) < current\_kNN\_distance$ do
6:      if $Dist(p_i, s) < current\_kNN\_distance$ then
7:          update the tentative $k$NN results
8:          update $current\_kNN\_distance$
9:      dequeue the next object $p_i$
10: return the final $k$NN results if not yet
```

unchecked split point on Ω as the check point, and go through the list of unscanned objects to compute its full kNN results. If any of the kNN results has not been returned to the client, it is output for immediate transmission. For our running example shown in Figure 12a, after scanning p_2, we may select s_1 as the check point. We then compute s_1's 2NN results as p_1 and p_2 and return them immediately. Compared to the bulk query processing, since the checking procedure here incurs extra overhead, the overall query processing time would be increased. Nevertheless, the worst-case time complexity remains $\mathcal{O}(kM^3)$ since the checking procedure adds a complexity of $\mathcal{O}(kM^2)$ only. On the other hand, the progressive algorithm can start returning the kCRNN results earlier and, hence, likely to result in a shorter user-perceived response time, as will be demonstrated in Section VII-D.

VII. Performance Evaluation

A. Experiment Setup

We have developed a testbed [9] to evaluate the performance of the proposed location cloaking and query processing algorithms. The client-side query interface and location cloaking agent were implemented on an O2 Xda Atom Exec PDA with Intel PXA 27x 520 MHz Processor and 64 MB RAM. The PDA supports GSM/GPRS/EGDE and WiFi communications. The LBS server was implemented on a Redhat 7.3 Linux server with Intel Xeon 2.80 GHz processor and 2 GB RAM. We assume that the client and the server communicate through a wireless network at a data transfer rate of 114 $kbps$.

Parameter	Setting
kNN Query (k)	5
Query Interval (I)	4 min
Privacy Requirement (r)	0.001
Spatial Object Set Size	2,249,727 objects
Object Record Size	1 kb
Query Size	20 $bytes$
Data Transfer Rate	114 $kbps$

TABLE I
Default Parameter Settings

The spatial object set used in the experiments contains 2,249,727 objects representing the centroids of the street segments in California [29]. We normalize the data space to a unit space and index the spatial objects by an R-tree (with a page fanout of 200 and a page occupancy of 70%) [16]. The size of an object record is set at 1 kb. To process a kCRNN query of a circle Ω, we first use our previously developed disk-based access method [20] to retrieve the kNN results for the minimum bounding rectangle of Ω. By definition, this set of kNN results is a superset of Ω's kCRNN results. The kCRNN processing algorithms developed in Section VI are then applied on this superset to get the kCRNN results of Ω.

We simulate a well-known random walk model [22], in which the user moves in steps. In each step, the user selects a speed and travels along an arbitrary direction for a duration of 2 min. We test two speed settings: 1) constant speed: the moving speed is fixed at 0.0003 $/min$; 2) random speed: for each step, a speed is randomly selected from a range of [0.0001 $/min$, 0.0005 $/min$]. By default, the random speed setting is adopted. The user makes privacy-conscious kNN queries from time to time. The query interval I is set at 4 min by default. The user specifies the privacy requirement by a radius r (i.e., the minimum acceptable cloaking area is πr^2), with a default setting of 0.001. The size of a kNN query message is set at 20 $bytes$. For the numerical method of optimal location cloaking, Δ is set at 0.0001, and the Simplex algorithm is employed to solve the linear program. The default parameter settings are summarized in Table I. The experimental results reported below are averaged over 1,000 randomly generated queries.

B. Effectiveness of Mobility-Aware Cloaking

In this section, we compare the proposed optimal mobility-aware cloaking technique (Algorithm 1) against the isolated cloaking scheme (described at the beginning of Section V). For both the optimal and isolated cloaking techniques, initially a cloaking region is randomly generated based on the user location. In other words, the user is equally likely to be at any location in the cloaking region. We measure the quality of the cloaking region for a subsequent query in terms of entropy based on 1,500 sample locations and 1,000 random queries. As shown in Figures 13a and 13b, when the query interval is small (i.e., 1 min), the entropy of isolated cloaking is nearly 20% lower than that of optimal cloaking for all queries tested. With increasing query interval, the average entropy of isolated cloaking improves (see Figure 13c) but is still far lower than that of optimal cloaking. When the query interval is 8 min, Figure 13a and 13b show that the entropy of isolated cloaking is 40% lower than that of optimal cloaking for over 15% of the queries tested and 20% lower for over 40% of the queries tested. Note that the results shown here are for one successive query only. With more successive queries, the quality of isolated cloaking would further degrade.

To highlight the benefit of achieving higher entropy, we conduct two possible attacks. Recall that through trace analysis attacks, the LBS server can derive the probabilities of the user being at different locations in a cloaking region. The first attack attempts to limit the possible user location to a sub-region with 95% confidence. The second attack calculates the highest aggregate probability for any sub-region with size equal to 5% of the cloaking region. Figures 14a and 14b show the results when the number of cloaking regions used in trace analysis attacks is increased from 1 to 10. We can see from the results that our optimal cloaking is robust against the attacks: for example, with the first attack (Figure 14a), the sub-region size is as large as 95% of the cloaking region since the derivable probability for the user to be at any location is uniform across the region. In contrast, with the same level of confidence, the sub-region size under isolated cloaking could be much smaller due to a skewed probability distribution (see Figure 14c for a sample distribution we observed in the experiment). Similarly, with the second attack (Figure 14b), under isolated cloaking, the server would be able to derive the probability for the user to be in a sub-region of 5% size of the cloaking region with a confidence of 16%-99%. The confidence for the same sub-region is only 5% under optimal cloaking.

C. Comparison of Mobility-Aware Cloaking Algorithms

This section compares the two cloaking algorithms developed in Section V based on the optimal cloaking technique, namely MaxAccu_Cloak (abbreviated as *MaxAccu*) and MinComm_Cloak (abbreviated as *MinComm*). Recall that MaxAccu aims at a higher query accuracy by minimizing the outer query blocking probability while MinComm attempts to achieve a balance between communication cost and query accuracy by maximizing the inner query blocking probability at the same time.

As shown in Figure 15a, MaxAccu has an outer query blocking probability of zero. Hence, its query results are 100% accurate as shown in Figure 15b. In contrast, MinComm has an outer query blocking probability of about 15%. For those blocked outer queries, approximate results are obtained based on cached result supersets. Figure 15b shows that the average error (measured by the ratio of the distance of an approximate kNN result to the actual kNN distance) is pretty small. In the worst case, the approximate kNN distance is no more than 2.3 times of the actual distance.

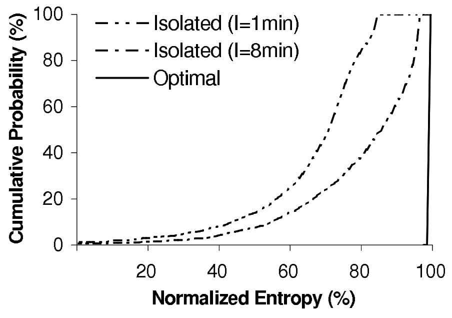
Figure 15a also shows that MinComm has a much higher inner query blocking probability than MaxAccu. Recall that inner queries are sent to the server for evaluation merely for the purpose of optimal cloaking. They do not affect query accuracy but communication cost. With more queries (including both inner and outer queries) being blocked, the communication cost incurred by MinComm is about half that of MaxAccu (see Figure 15c).

D. Comparison of kCRNN Query Processing Algorithms

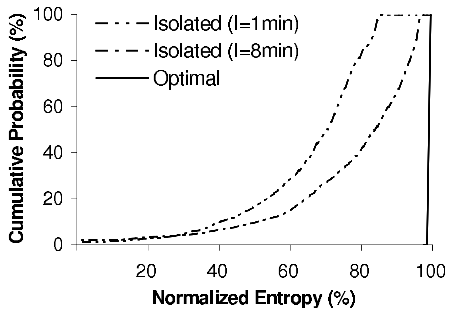
In this section, we evaluate the performance of the *bulk* and *progressive* kCRNN query processing algorithms developed in Section VI. Figure 16 shows the user-perceived response time for both algorithms. When k or r is small, the bulk and progressive algorithms perform similarly. However, when k or r is large, the progressive algorithm clearly outperforms the bulk algorithm. To explain, we show in Figures 17a and 17b the timeline performance of two sample queries with r=0.25$\times 10^{-3}$ and r=4$\times 10^{-3}$, respectively. For the query with r=0.25$\times 10^{-3}$ (Figure 17a), both the bulk and progressive algorithms took a short time to process. Thus, parallelizing the query evaluation and result transmission does not help a lot in user-perceived response time. On the other hand, when r=4$\times 10^{-3}$ (Figure 17b), the result superset size is large and the query requires a long time (over 1000 ms) to evaluate; then by returning the kCRNN results incrementally, the progressive algorithm completes the result transmission earlier.

E. End-to-End System Performance

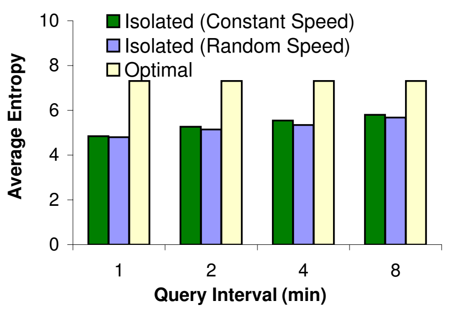
This section evaluates the end-to-end system performance. In this set of experiments, we used the progressive kCRNN query processing. In addition to the MaxAccu and MinComm cloaking algorithms, the existing isolated cloaking method is also included for comparison. For all the cloaking algorithms, the inner queries (i.e., the queries inside the last cloaking region) reuse cached result supersets and compute their answers



(a) Percentile Entropy (Constant Speed)

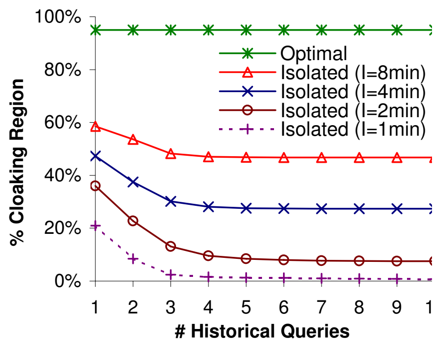


(b) Percentile Entropy (Random Speed)

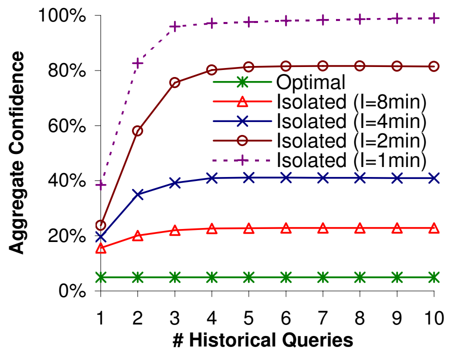


(c) Average Entropy

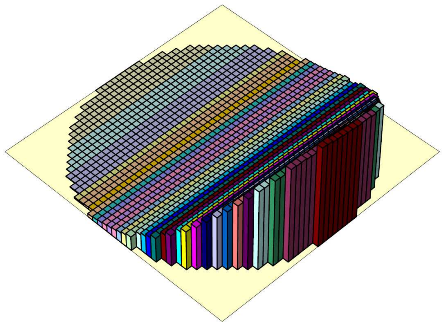
Fig. 13. Performance of Cloaking Quality (Entropy)



(a) First Attack (5% Confidence)

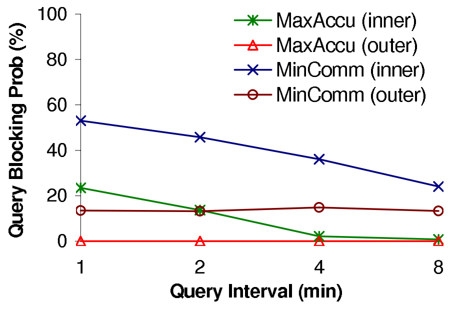


(b) Second Attack (5% Region)

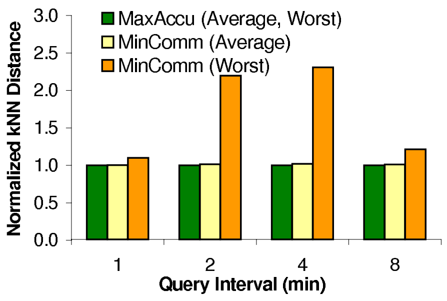


(c) A Probability Distribution of Isolated Cloaking

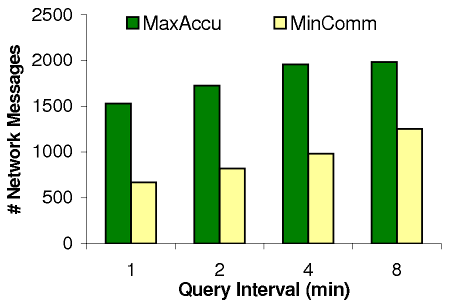
Fig. 14. Trace Analysis Attacks



(a) Query Blocking Probability

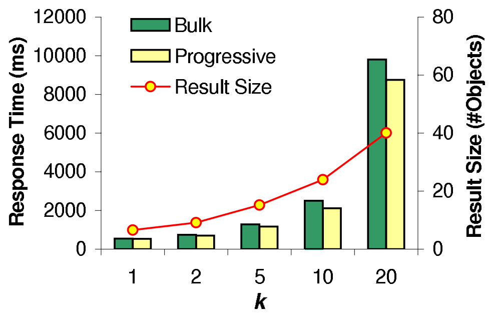


(b) Query Accuracy

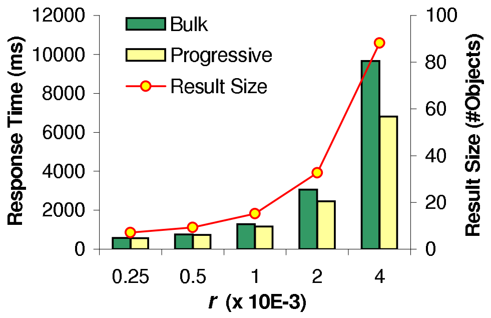


(c) Communication Cost

Fig. 15. Performance Comparison of Cloaking Algorithms



(a) Varying k ($r = 0.001$)

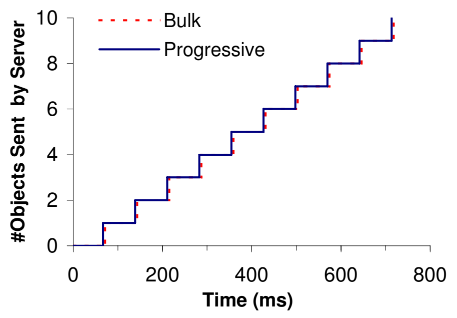


(b) Varying Region Size ($k = 5$)

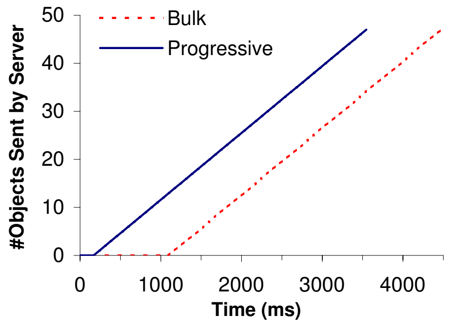
Fig. 16. Performance Comparison of kCRNN Query Processing Algorithms

immediately. The inner queries are not sent to the server by default. However, with MaxAccu and MinComm, some of them might need to be sent to the server to achieve optimal cloaking, depending on the inner query blocking probability. Moreover, as discussed before, the blocked outer queries for MaxAccu and MinComm compute their approximate kNN results based on cached result supersets.

Figures 18a and 18b show the average end-to-end query latency, which is defined as the period from the time when the user issues a location-based query to the time when the exact query results are obtained. It can be seen that MinComm outperforms MaxAccu in all cases tested. This is explained as

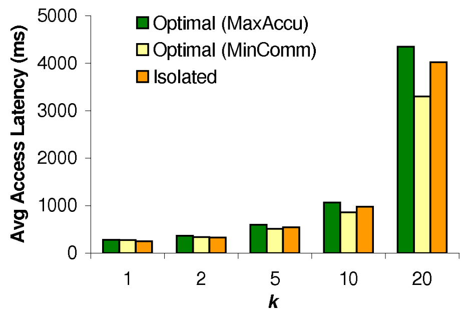


(a) $k = 5, r = 0.25 \times 10^{-3}$

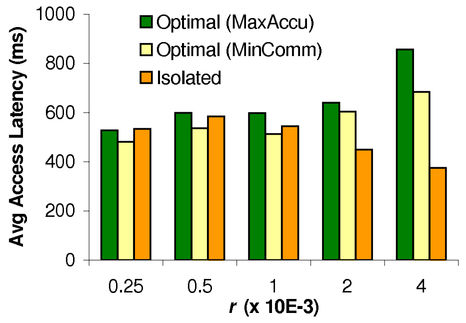


(b) $k = 5, r = 4 \times 10^{-3}$

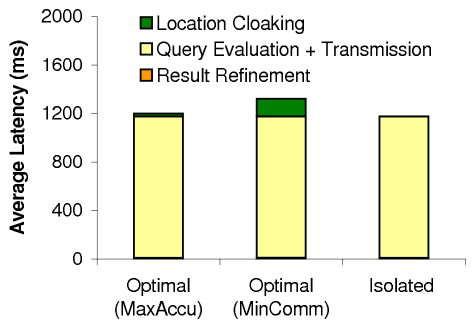
Fig. 17. Timeline Performance



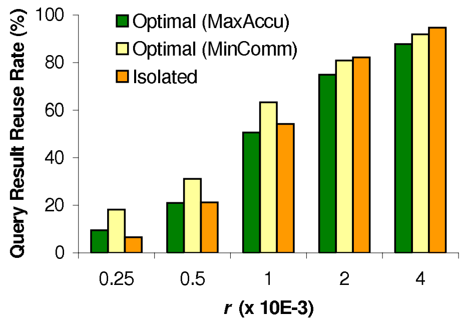
(a) Varying k ($r = 1 \times 10^{-3}$)



(b) Varying Region Size (k=5)

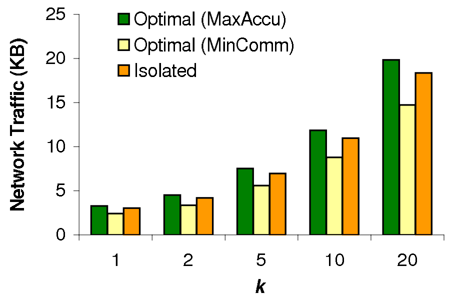


(c) Latency Breakdown ($k = 5, r = 1 \times 10^{-3}$)

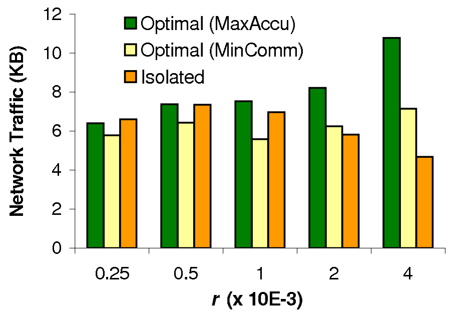


(d) Result Reuse Rate (k=5)

Fig. 18. End-to-End Latency Performance



(a) Varying k ($r = 1 \times 10^{-3}$)



(b) Varying Region Size (k=5)

Fig. 19. Network Traffic Performance

follows. As shown in Figure 18c, the query evaluation and result transmission time dominates the overall query latency. Since MinComm has a higher outer query blocking rate and hence a higher result reuse rate (Figure 18d), it results in a lower average query latency. For the same reason, MinComm outperforms Isolated when the region size is small (see Figure 18b). When the region size is large, Isolated performs better than both MaxAccu and MinComm due to a higher result reuse rate. Their relative performance is insensitive to the value of k (see Figure 18a).

Figures 19a and 19b show the average amount of network traffic incurred for each query, which is a good indicator of energy consumption on the client. The less is the network traffic, the lower is the energy consumption. As expected, MinComm incurs less network traffic than MaxAccu due to a higher query blocking rate. Both MaxAccu and MinComm are competitive compared to Isolated; in particular MinComm outperforms Isolated for most cases tested. Summarizing the results of Figures 13, 18, and 19, it can be concluded that the price to pay for resisting trace analysis attacks is not high. Our proposed MaxAccu and MinComm cloaking algorithms improve the cloaking quality over Isolated without compromising much on query latency or communication cost (sometimes performing even better).

VIII. Conclusion

This paper has presented a complete study on processing privacy-conscious location-based queries in mobile environments. The technical contributions made in this paper are summarized as follows:

- We have studied the representation of cloaking regions and showed that a circular region generally leads to a small result superset.
- We have developed an optimal mobility-aware location cloaking technique to resist trace analysis attacks. Two cloaking algorithms, namely *MaxAccu_Cloak* and *MinComm_Cloak*, have been designed to favor different performance objectives.
- We have developed two efficient polynomial algorithms, namely *bulk* and *progressive*, for processing circular-region-based kNN queries.

We have also conducted simulation experiments to evaluate the proposed algorithms. Experimental results show that the optimal mobility-aware cloaking algorithms is robust against trace analysis attacks without compromising much on query latency or communication cost. MaxAccu_Cloak gets a 100% query accuracy while MinComm_Cloak achieves a good balance between communication cost and query accuracy. It is also shown that the progressive query processing algorithm generally achieves a shorter user-perceived response time than the bulk algorithm.

As for future work, we are going to extend the mobility-aware location cloaking technique to other privacy metrics (e.g., the k-anonymity model and the l-diversity model) and road networks. We are also interested in investigating mobility-aware peer-to-peer cloaking techniques.

Acknowledgements

The authors would like to thank the editor and anonymous reviewers for their valuable suggestions that significantly improved the quality of this paper. This work was supported by the Research Grants Council, Hong Kong SAR, China under Project No. HKBU211206.

References

[1] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. *PODS*, 2001.

[2] B. Bamba, L. Liu, P. Pesti, and T. Wang. Supporting anonymous location queries in mobile environments with PrivacyGrid. *WWW*, 2008.

[3] C. Bettini, S. Mascetti, X. S. Wang, and S. Jajodia. Anonymity in location-based services: Towards a general framework. *8th Intl. Conf. on Mobile Data Management (MDM)*, May 2007.

[4] S. Berchtold, C. Böhm, D. A. Keim, F. Krebs, and H.-P. Kriegel. On optimizing nearest neighbor queries in high-dimensional data spaces. *ICDT* 2001.

[5] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1), 2003.

[6] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar. Preserving user location privacy in mobile data management infrastructures. *Privacy Enhancing Technology Workshop*, Cambridge, UK, June 2006.

[7] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. *ACM MobiCom*, 2000.

[8] C.-Y. Chow, M. F. Mokbel, and X. Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based services. *ACM GIS*, Arlington, VA, 2006.

[9] J. Du, J. Xu, X. Tang, and H. Hu. iPDA: Supporting privacy-preserving location-based mobile services (Demonstration). *8th Intl. Conf. on Mobile Data Management (MDM)*, May 2007.

[10] Geopriv Working Group. [Online] http://www.ietf.org/html.charters/geopriv-charter.html, 2005.

[11] G. Myles, A. Friday, and N. Davies. Preserving privacy in environments with location-based applications. *IEEE Pervasive Computing*, 2(1):56–64, 2003.

[12] G. Ghinita, P. Kalnis, and S. Skiadopoulos. Prive: Anonymous location-based queries in distributed mobile systems. *Proc. WWW'07*, pages 371–380, 2007.

[13] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. *ACM MobiSys*, 2003.

[14] B. Gedik and L. Liu. A customizable k-anonymity model for protecting location privacy. *ICDCS*, 2005.

[15] B. Gedik and L. Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, 2008.

[16] A. Guttman. R-trees: A dynamic index structure for spatial searching. *SIMGOD*, 1984.

[17] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *TODS*, 24(2):265–318, 1999.

[18] J. I. Hong and J. A. Landay. An architecture for privacy-sensitive ubiquitous computing. *ACM MobiSys*, 2004.

[19] Y.-C. Hu and H. J. Wang. Location privacy in wireless networks. *ACM SIGCOMM Asia Workshop*, April 2005.

[20] H. Hu and D. Lee. Range nearest neighbor queries. *TKDE*, 18(1):78–91, 2006.

[21] H. Hu, J. Xu, W. S. Wong, B. Zheng, D. L. Lee, and W.-C. Lee. Proactive caching for spatial queries in mobile environments. *ICDE*, 2005.

[22] Barry D. Hughes. *Random Walks and Random Environments.* Oxford University Press, 1996.

[23] X. Liu, M. D. Corner, and P. Shenoy. Ferret: RFID locationing for pervasive multimedia. *Proc. Ubicomp*, Irvine, CA, Sept. 2006.

[24] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preserving anonymity in location based services. *Technical Report, School of Computing, The National University of Singapore*, 2006.

[25] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. *Intl. Conf. on Pervasive Services (ICPS)*, 2005.

[26] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The New Casper: Query procesing for location services without compromising privacy. *VLDB*, 2006.

[27] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979.

[28] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. *SIGMOD*, 1995.

[29] The R-tree Portal. [Online] http://www.rtreeportal.org/.

[30] B. Schilit, J. Hong, and M. Gruteser. Wireless location privacy protection. *IEEE Computer*, Dec. 2003.

[31] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha. Tracking moving devices with the Cricket location system. *USENIX/ACM Mobisys*, Boston, MA, June 2004.

[32] D. Smith and S. Singh. Approaches to multisensor data fusion in target tracking: a survey. *IEEE Transactions on Knowledge and Data Engineering*, December 2006.

[33] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. *VLDB*, 2002.

[34] Toby Xu and Ying Cai. Exploring historical location data for anonymity preservation in location-based services. *Infocom*, 2008.

[35] J. Xu, B. Zheng, W.-C. Lee, and D. L. Lee. Energy-efficient index for querying location-dependent data in mobile broadcast environments. *ICDE*, 2003.

[36] M. Youssef, V. Atluri, and N. R. Adam. Preserving mobile customer privacy: An access control system for moving objects and custom profiles. *6th Intl. Conf. on Mobile Data Management (MDM)*, 2005.

[37] J. Yoon, B. D. Noble, M. Liu, and M. Kim. Building realistic mobility models from coarse-grained traces. *ACM/USENIX MobiSys*, 2006.