SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services

Man Lung Yiu [1], Christian S. Jensen [1], Xuegang Huang [1], Hua Lu [1]

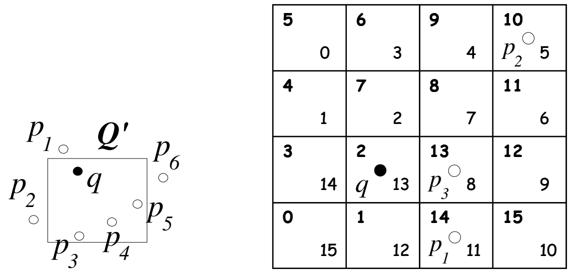
[1]*Department of Computer Science, Aalborg University*
DK-9220 Aalborg, Denmark
{mly,csj,xghuang,luhua}@cs.aau.dk

Abstract— **In a mobile service scenario, users query a server for nearby points of interest but they may not want to disclose their locations to the service. Intuitively, location privacy may be obtained at the cost of query performance and query accuracy. The challenge addressed is how to obtain the best possible performance, subjected to given requirements for location privacy and query accuracy. Existing privacy solutions that use spatial cloaking employ complex server query processing techniques and entail the transmission of large quantities of intermediate result. Solutions that use transformation-based matching generally fall short in offering practical query accuracy guarantees. Our proposed framework, called SpaceTwist, rectifies these shortcomings for k nearest neighbor (kNN) queries. Starting with a location different from the user's actual location, nearest neighbors are retrieved incrementally until the query is answered correctly by the mobile terminal. This approach is flexible, needs no trusted middleware, and requires only well-known incremental NN query processing on the server. The framework also includes a server-side granular search technique that exploits relaxed query accuracy guarantees for obtaining better performance. The paper reports on empirical studies that elicit key properties of SpaceTwist and suggest that the framework offers very good performance and high privacy, at low communication cost.**

I. Introduction

The emerging mobile Internet will offer services that retrieve the locations and other information pertaining to so-called points-of-interest (POIs), e.g., stores, restaurants, and tourist attractions. Such services will rely on the k nearest neighbor (kNN) query [1], [2] that retrieves the k POIs closest to a user's location q. Figure 1a shows an example with 6 POIs p_i, maintained by a central server, and a user location q. To retrieve (information about) the nearest POI, the user sends location q to the server, in response to which the server identifies and returns p_1. As a complication to this scenario, users may wish to not disclose their (exact) locations to the server. Existing approaches to preserving location privacy can be classified as using either spatial cloaking and transformation-based matching.

Spatial cloaking techniques [3]–[9] enlarge an exact user location q into a cloaked region Q' so that it is impossible to reconstruct q from the region Q' at the server. To ensure that an accurate query result can be computed, the server computes a candidate set that includes the nearest POI for any location in Q'. This result may then subsequently be refined by the user's trusted mobile client. Figure 1a exemplifies the case where Q' is a rectangle [3], [4]. Here, the server reports the POIs $p_1, \ldots, p_6$. All POIs inside Q' must be reported since the user can be at any location in Q'. As the server maintains a large number of POIs, the processing of a cloaked query may incur high processing and communication costs. In addition, the server must include support for the specialized processing techniques needed to process cloaked queries.



(a) spatial cloaking (b) transformation-based matching

Fig. 1. Existing Location Privacy Solutions

With *transformation-based matching* [10], [11], the query is evaluated in a transformed space, in which the points and/or distances between points are encoded. A key drawback is that query results may not always be accurate. One approach [11] defines a specific Hilbert ordering with a key $\mathcal{H}$, whose value is known only by the client and a trusted entity. Not having the key value, the server cannot decode a Hilbert value into a point. In preparation for querying, the trusted entity transforms each POI p_i into Hilbert value $\mathcal{H}(p_i)$ and uploads it to the server. In Figure 1b, the Hilbert value of each cell is shown in the top-left corner of the cell. At query time, the client q submits its Hilbert value $\mathcal{H}(q)$=2 to the server. The server then reports the closest Hilbert value $\mathcal{H}(p_2)$=10 of $\mathcal{H}(q)$, which is eventually decoded by the client into point p_2.

This paper presents a framework, called *SpaceTwist*, that aims to improve on the above approaches. POIs are retrieved from the server *incrementally* [2]. The process starts with an *anchor*, a location different from that of the user, and it proceeds until an accurate query result can be reported. This approach is capable of offering location privacy, as we will see in Section III. The approach differs fundamentally from previous approaches: unlike in spatial cloaking, no cloaked region is applied, and unlike in transformation-based matching, the query is evaluated in the original space.

In addition, our approach requires only simple query processing on the server—namely incremental nearest neighbor (INN) retrieval, which has been studied extensively [2] and readily implemented on existing servers. In contrast, spatial cloaking and transformation-based matching approaches require specialized query processing algorithms.

The framework also includes a granular search technique that aims to exploit a tolerance for relaxing the accuracy of query results, but with guaranteed accuracy bounds, for reducing the communication cost and server load. Note that, existing transformation-based matching solutions fall short in offering accuracy guarantees, and spatial cloaking solutions may incur high costs.

The paper's contributions are as follows.

- A client-side query processing technique that retrieves POIs from the server incrementally, supports location privacy, and reuses existing server-side functionality.
- A study of the privacy afforded by this technique.
- A server-side granular search technique that is able to exploit relaxed query accuracies (with guarantees) for reducing the communication cost and server load.
- Empirical studies that suggest that the proposed techniques are highly performant.

Section II reviews the related work. Section III presents the query processing technique and analyzes its privacy protection. Section IV then covers the granular search technique. Section V discusses the configuration of parameter values in the SpaceTwist framework. Section VI covers the results of extensive studies of the proposed techniques. Section VII contains further discussion of our privacy model. Finally, Section VIII concludes and identifies research directions.

II. Related Work

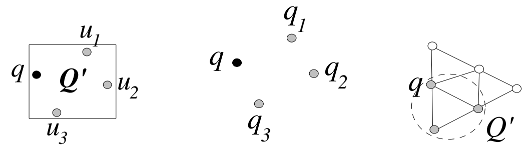
We review existing location privacy protection techniques, which use either spatial cloaking or transformation-based matching.

A. Spatial Cloaking

With cloaking, the user location q is enlarged into a *cloaked region* Q' that is then used for querying the server [3]–[9], [12]. This way, q is hidden in Q'. The existing cloaking solutions differ with respect to (i) the representation of Q', (ii) the architecture for cloaking, and (iii) the query processing.

Cloaked Region Representation Cloaked regions come in two forms: they are either plain, connected regions (e.g., rectangles) or they are discrete and posses "multiple parts" (e.g., sets of point locations).

Thus, some work [3]–[5], [8] represents the cloaked region Q' by a K-anonymous [13] rectangle, which contains the query location q and at least $K - 1$ other user locations. Figure 2a illustrates a 4-anonymous region Q', where u_1, u_2, and u_3 are $(4 - 1)$ user locations. Other work [4], [14], [15] uses circular cloaked regions. The study of Ardagna et al. [15] takes location positioning inaccuracy into account, models the user location as a circular region, and develops several geometric operators for deriving cloaked regions.



(a) K-anonymous (b) dummies (c) obfuscation

Fig. 2. Example Cloaked Regions

The cloaked region has also been represented by a point set containing q and a number of dummy locations (generated by the client) [7]. In Figure 2b, q_1, q_2, q_3 are dummy locations, and the cloaked region is $Q' = \{q, q_1, q_2, q_3\}$. In another approach, the possible locations are restricted to be vertices in a graph (e.g., a road network), and the cloaked region (or "obfuscation") is represented by a set of vertices [9], [16]. The cloaked region Q' in Figure 2c is the set of gray vertices, one of them being q. The approaches that represent Q' as a set quantify the privacy of Q' by its cardinality.

Cloaking Architecture A simple approach to construct a cloaked region is to do so at the client [7], [9], [16]. However, client-based cloaking does not support the use of K-anonymous regions. This requires knowing the locations of other users, which may be achieved by introducing a trusted, third party, location anonymizer [3], [4] that knows the locations of a population of users.

Given a user location q, the anonymizer is capable of computing a plain cloaked region Q' (e.g., a rectangle or circle) that contains q and at least $K - 1$ other user locations. However, the location anonymizer becomes a performance bottleneck and may also be vulnerable to malicious attack. Some users may not trust the anonymizer.

A K-anonymous region can also be derived through peer-to-peer communication [8], [17], [18]. Users close together form a group and set their cloaked region as a rectangle containing them all. The drawback is that group formation and maintenance incur communication overhead and latency.

Server Side Query Processing A discrete cloaked region query may be processed by processing each point in the query in turn, returning the union of results [7], [9], [16]. In this setting, a negotiation protocol for trading between privacy and result accuracy has also been proposed [9], [16].

The processing of plain cloaked regions is more complex. Specialized (server-side) algorithms have been proposed for identifying a candidate set that includes the NN for any location in a cloaked region [3], [4]. These algorithms go beyond well-known point NN algorithms [1], [2] and introduce complexity. In addition, this approach is prone to high server-side processing costs and communication cost for large numbers of POIs. This restricts the applicability of plain cloaked regions.

Summary Cloaking solutions incur expensive processing and communication cost for large number of POIs. When using plain cloaked regions, specialized query processing algorithms are needed, which may not be present in the available LBS servers. Next, discrete cloaked regions lack concise representations, so sending these to the server incurs high communication costs. To avoid the above shortcomings, our proposal avoids the use of cloaked regions.

B. Transformation-Based Matching

Recently, transformation-based matching techniques [10], [11] have been proposed to enable location privacy. However, these do not offer query accuracy guarantees.

A theoretical study on a client-server protocol for deriving the nearest neighbor of q has recently been reported [10]. Its communication cost is asymptotic to $\sqrt{N}$, where N is the number of POIs. No experimental evaluation of the communication cost and result accuracy of the protocol with real data is available.

Another study defines a specific Hilbert ordering based on a key $\mathcal{H}$, whose value is known by only the client and a trusted entity [11]. It is shown that without the key value, it is impossible to decode a Hilbert value into a location correctly. However, as exemplified in Section I, a Hilbert curve does not completely preserve spatial proximity, so the reported result can be far from q. To improve the accuracy, the use of two keys $\mathcal{H}$ and $\mathcal{H}'$ with orthogonal Hilbert curves has been considered [11]. In Figure 1b, the numbers in the top-left corners and bottom-right corners of the cells represent the Hilbert values $\mathcal{H}(p_i)$ and $\mathcal{H}'(p_i)$ respectively, for point(s) p_i inside the cells. At query time, the user sends the values $\mathcal{H}(q) = 2$ and $\mathcal{H}'(q) = 13$ to the server, which reports the corresponding closest Hilbert values $\mathcal{H}(p_2) = 10$ and $\mathcal{H}'(p_1) = 11$. The client subsequently decodes the received Hilbert values into points and selects the closer point (i.e., p_1) to be its NN.

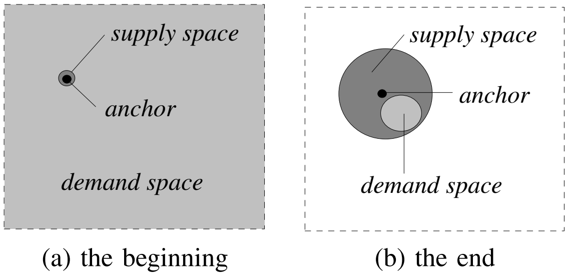
III. SpaceTwist: Incremental Processing

We propose an algorithm that computes exact kNN query results in incremental fashion and affords the user location privacy. We assume only a simple client-server architecture, and we assume that the server indexes the dataset P (of POIs) by an R-tree [19] and supports incremental nearest neighbor retrieval [2]. This way, our solution can be readily applied to existing LBS servers. In particular, we only consider the snapshot kNN query [3], [4] and leave continuous query [20] for future work.

Following an overview in Section III-A, Section III-B describes the client-side algorithm, and Section III-C analyzes the location privacy achieved.

A. Overview

Figure 3 offers an overview of our technique. The client specifies an *anchor* (a "fake" location) and iteratively requests POIs from the server in ascending distance order [2] from the anchor. The *supply space* centered at the anchor is the part of space already explored. The *demand space* denotes the space to be covered before the client is guaranteed to be able to produce an accurate result. The client knows both the demand space and the supply space, whereas the server knows only the supply space. In the beginning (see Figure 3a), the demand space is set to the domain space, and the supply space is empty. As points are retrieved from the server, the supply space expands. When a retrieved point p is the closest point to the client seen so far, the results are updated, and the demand space shrinks. When the supply space eventually covers the demand space (see Figure 3b), it is termed *final* and the client is guaranteed to produce an accurate result.



(a) the beginning (b) the end

Fig. 3. Demand Space and Supply Space

Considering the communication cost of the above technique, the major data flow is that consisting of data points being sent from the server to the client. We assume that such data points are transmitted through TCP/IP packets. In a naive solution, each time the server retrieves a data point p, a packet containing p is sent to the client. This yields a high communication cost as the capacity of the packets is utilized poorly. In practice, the server accumulates multiple points, packs them into the same packet, and sends the packet to the client. System parameter β denotes the number of points that fit in a packet. Its value will be discussed in Section VI. We measure the *communication cost* of our solution as the total number of packets received by the client.

B. The SpaceTwist Client Algorithm

We proceed to present the client-side algorithm for accurate kNN retrieval. We use the notation $dist(q, p)$ to denote the Euclidean distance between two points q and p.

The client (i.e., user) executes Algorithm 1 to obtain its k nearest objects from the server (i.e., query processor). The *anchor location* q' is first sent to the server. On the other hand, the user location q is known only by the client. Intuitively, if q and q' are close then few objects are retrieved (i.e., low cost) but less location privacy is achieved. Guidelines for selecting an appropriate q' are discussed in Section V.

A max-heap W_k, initialized with k virtual objects, maintains the k nearest objects (of q) seen so far. Let γ be the maximum distance in W_k. The *demand space* is then the circle with radius γ and center q (see Line 3). Let τ be the largest distance to q' of any object examined so far. The *supply space* is then the circle with radius τ and center q' (see Line 4). Next, the server is requested to return incremental nearest neighbors (INNs) [2] of q'.

Algorithm 1 Space Twist Client (for kNN query)

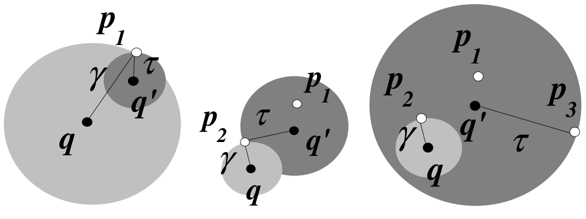
```
    algorithm SpaceTwistClient(Value $k$, Point $q$, Point $q'$)
    system parameter: packet capacity $\beta$
 1: $W_k \leftarrow$ new max-heap of pairs $\langle p, dist(q,p)\rangle$;
 2: insert $k$ pairs of $\langle NULL, \infty\rangle$ into $W_k$;
 3: $\gamma \leftarrow$ the top distance of $W_k$;          ▷ $k^{th}$ best distance from $q$
 4: $\tau \leftarrow 0$;                                     ▷ furthest distance seen from $q'$
 5: send an INN query with $q'$ to the server;
 6: while $\gamma + dist(q,q') > \tau$ do
 7:     $S \leftarrow$ get the next packet of points from the server;
 8:     $\tau \leftarrow \max_{p \in S} dist(q',p)$;                 ▷ update supply space
 9:     for all $p \in S$ do
10:         if $dist(q,p) < \gamma$ then                    ▷ check demand space
11:             update $W_k$ (and $\gamma$) by using $p$;
12: terminate the INN query at the server;
13: return $W_k$;
```

In Line 7, the client retrieves the next packet of INNs (of q') from the server. τ is updated to the furthest distance of point in S from q'. For each retrieved point p, we check whether $dist(q,p)$ is less than γ (i.e., whether q is closer to p than some object in W_k). If so, then W_k and γ are updated. According to Lemma 1, the loop continues as long as $\gamma + dist(q,q') > \tau$. Finally, the client returns the result set W_k after terminating the INN query at the server.

Lemma 1: If $\gamma + dist(q,q') \leq \tau$ then the actual k^{th} nearest object (say, $p_\star$) of q has been retrieved.

Proof: Since $p_\star$ has an upper bound distance of γ from q, its upper bound distance from q' is $\gamma + dist(q,q')$, according to the triangular inequality. Based on the property of incremental nearest neighbor retrieval [2], all objects within distance τ from q' have been seen. Thus, we conclude that $p_\star$ has already been retrieved. ■

Example Figure 4 exemplifies the algorithm, for the case k=1 and β=1. When we discover point p_1 (see Figure 4a), we set the best result to p_1 and define the demand space (light gray area) around q as well as the supply space (dark gray area) around q'. Next, in Figure 4b, point p_2 is discovered and the supply space expands. Since q is closer to p_2 than the previous result (i.e., p_1), the best result is updated to p_2 and the demand space shrinks. Then point p_3 is retrieved (see Figure 4c) and the supply space grows. As the supply space encloses the demand space, the algorithm terminates and returns p_2 as the nearest neighbor of q.



(a) first point (b) second point (c) third point

Fig. 4. Query Processing Example

We observe that, based on the given parameters k, q, and q', the algorithm computes the *exact result* and terminates without requesting unnecessary packets from the server, due to the correctness of Lemma 1.

C. Privacy Analysis

This section studies how an adversary is able to *infer* the possible locations of the user. We assume that the adversary knows only: (i) the anchor q' and the value k, (ii) reported points from the server, and (iii) termination condition of Algorithm 1. As we assume a simple client-server architecture, the concept of K-anonymity [3], [13] is inapplicable.

In the following, we denote a *possible* user location by q_c, in order to distinguish it from the *actual* user location q. Let m be the number of packets received by the client and let the points received (in their retrieval order) be $p_1, p_2, \cdots, p_{m\beta}$. Since the algorithm did not terminate at the final point of the penultimate packet received, we have:

$$dist(q_c, q') + \overset{k}{\min_{1 \leq i \leq (m-1)\beta}} dist(q_c, p_i) > dist(q', p_{(m-1)\beta}) \tag{1}$$

where the middle term represents the k^{th} smallest distance of the first $(m-1)\beta$ points from q_c.

Due to the packet capacity β, the adversary does not know the specific data point (in the last packet) leading the algorithm to terminate. Thus, the adversary only deduces:

$$dist(q_c, q') + \overset{k}{\min_{1 \leq i \leq m\beta}} dist(q_c, p_i) \leq dist(q', p_{m\beta}) \tag{2}$$

Clearly, a possible user location q_c must satisfy both inequalities above. We define the *inferred privacy region* Ψ as the set of all possible locations q_c. Intuitively, the *privacy value* is quantified as the average distance of a location in Ψ from the user's actual location q:

$$\Gamma(q, \Psi) = \frac{\int_{z \in \Psi} dist(z, q)\ dz}{\int_{z \in \Psi}\ dz} \tag{3}$$

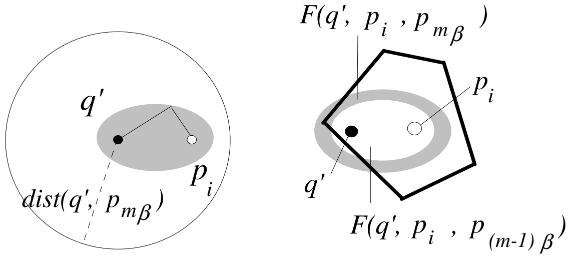
While region Ψ can be inferred by both the user and the adversary, only the user can derive the privacy value $\Gamma(q, \Psi)$.

Since Ψ does not have closed-form expression in general, its derivation is non-trivial. The Monte Carlo method can be used for approximating Ψ, by randomly generating candidate locations for q_c and checking them against inequalities 1 and 2.

Exact Privacy Region Derivation Fortunately, we have discovered a closed-form expression for Ψ for the case $k = 1$. For each retrieved point p_i, we can derive $Vor(p_i)$, its Voronoi cell [21] with respect to all retrieved points. Observe that p_i is the NN of any location q_c inside $Vor(p_i)$. Furthermore, the possible location of q_c is constrained by the termination condition of the algorithm. Figure 5a depicts the final supply space as the circle with radius $dist(q', p_{m\beta})$ and center at the anchor q'. Termination occurs when the supply space covers the demand space:

$$dist(q_c, q') + dist(q_c, p_i) \leq dist(q', p_{m\beta}) \tag{4}$$

The set of locations satisfying this inequality can be expressed as an elliptical region $F(q', p_i, p_{m\beta})$ (shown in gray in Figure 5a) with foci q' and p_i, where any point on the border has its sum of distances to the foci being equal to $dist(q', p_{m\beta})$.



(a) supply space constraint (b) privacy region

Fig. 5. Inferring a Privacy Region

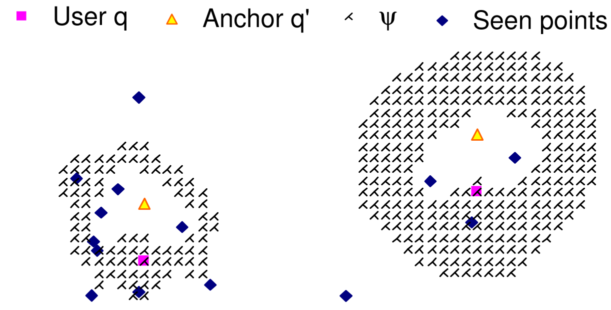
Similarly, we obtain $F(q', p_i, p_{(m-1)\beta})$, the elliptical region with $p_{m\beta}$ replaced by $p_{(m-1)\beta}$. Since the algorithm did not terminate at point $p_{(m-1)\beta}$, we exclude $F(q', p_i, p_{(m-1)\beta})$ from the possible region, as shown in Figure 5b.

Combining the above with the Voronoi cell $Vor(p_i)$, the inferred privacy region is given by:

$$\Psi = \bigcup_{i=1}^{m\beta} Vor(p_i) \cap (F(q', p_i, p_{m\beta}) - F(q', p_i, p_{(m-1)\beta}))$$

Visualization of Ψ We apply the above derivation on a dataset in order to visualize the inferred privacy region Ψ. Figure 6 depicts the actual user location q, the anchor q', the retrieved points, and the region Ψ.

Figure 6a illustrates the inferred privacy region Ψ for the case $\beta = 4$. Observe that Ψ is approximately a ring centered at q', with the distance $dist(q, q')$ to q'. As we will see in the experimental section, the privacy value $\Gamma(q, \Psi)$ obtained from Equation 3 is at least the anchor distance $dist(q, q')$.



(a) packet capacity $\beta = 4$ (b) coarser granularity

Fig. 6. Inferred Privacy Regions, $k = 1$

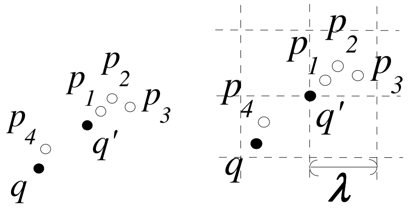
It is worth noticing that even when $dist(q, q')$ is fixed, the privacy value can be improved by performing the search at a lower data density, as illustrated in Figure 6b. The next section develops a search technique that supports user-specified granularities.

IV. Granular Search

We develop a server-based granular search technique that is capable of retrieving data points from the server with a user-specified granularity. This technique enables communication cost reduction and location privacy improvement while providing strict guarantees on the accuracies of the query results. Section IV-A describes granular search for the case k=1; its implementation is covered in Section IV-B. Section IV-C extends granular search to arbitrary values of k.

A. Basic Granular NN Search

Recall that the client-side algorithm requests POIs from the server in ascending order of their distance to anchor q'. For the example in Figure 7a, the server returns points in the order: p_1, p_2, p_3, p_4. Although p_4 is the actual NN of q, it cannot be obtained early by the client.



(a) set of points (b) grid cells

Fig. 7. Granular Search

The communication cost can be reduced by returning only a sample of the reported POIs. A threshold ϵ is then introduced for controlling the result accuracy:

Definition 1: Given a point set P, a distance threshold ϵ, and a location q, a point $p \in P$ is said to be an ϵ-relaxed kNN of q when $dist(q, p) \leq \epsilon + \min^k_{p' \in P} dist(q, p')$, where the last term represents the k^{th} NN distance to q in P.

The idea behind granular search is to impose a grid (with cell extent λ) on the domain space, as shown in Figure 7b. The server then iteratively retrieves incremental nearest neighbors of anchor q', except that it disregards points in a grid cell from which a point has already been reported. To ensure that the query result is an ϵ-relaxed NN of q, it suffices to set the cell extent λ to $\epsilon/\sqrt{2}$:

Lemma 2: Consider a regular grid with cell extent of λ. Let $p_\star$ be the actual NN of q and p' be the retrieved NN of q. It holds that $dist(q, p') \leq dist(q, p_\star) + \sqrt{2} \cdot \lambda$.

Proof: In case $p_\star$ has been retrieved, the inequality holds trivially (by setting p' to $p_\star$).

Otherwise, $p_\star$ has not been retrieved. Thus, a point p'' in the cell of $p_\star$ must have been retrieved. The maximum possible distance between p and p'' is the diagonal length of the cell, i.e., $\sqrt{2} \cdot \lambda$. From the triangular inequality, we obtain $dist(q, p'') \leq dist(q, p_\star) + dist(p_\star, p'') \leq dist(q, p_\star) + \sqrt{2} \cdot \lambda$. Since p' is the retrieved NN of q, we have $dist(q, p') \leq dist(q, p'')$, completing the proof. ■

Continuing with the example in Figure 7b, the server first sends point p_1 to the client. Since p_2 and p_3 fall in the cell of p_1, they are disregarded. Finally, p_4 is reported to the client. In this example, the communication cost drops from 4 POIs to 2 POIs.

B. Implementation of Granular Search

We proceed to consider the implementation of the above method. If the error bound ϵ is given in advance, then it is possible to pre-select a data point from each (non-empty) cell and index those points by another (small) R-tree, which is then used at query time. This pre-computation approach becomes impractical when different users use different values for ϵ and may choose these values at run time.

In the context of data streams, efficient main-memory data structures for maintaining relaxed results for NN queries with fixed error bounds have been proposed [22]. We are unable to use these because (i) we deal with large, disk-based point sets, and (ii) they require the error bound to be known in advance.

Algorithm 2 shows our granular incremental NN algorithm, which takes the user-specified error bound ϵ as input. A conceptual grid with cell extent λ ($= \epsilon/\sqrt{2}$) is imposed on the returned points during runtime. The algorithm also takes an R-tree R (of the data points) and an anchor q' as arguments. The notation $mindist(q', e)$ ($maxdist(q', e)$) represents the minimum (maximum) possible distance between q' and an R-tree entry e [1], [2]. Next, $\mathcal{C}_\lambda(p)$ denotes the cell containing point p.

The algorithm applies INN search [2] around anchor q', with two modifications: (i) a set V is employed (Line 3) for tracking the grid cells of the reported points (Line 12), and (ii) only qualifying entries that are not covered by the union of cells in V are further processed (Line 9).

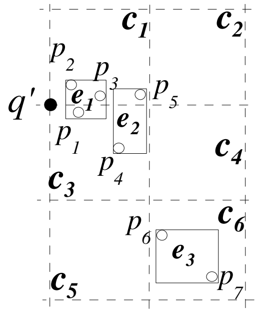
Algorithm 2 Granular Incremental NN

```
    algorithm GranularINN(R-Tree $R$, Point $q'$, Value $\epsilon$)
 1: $\lambda \leftarrow \epsilon/\sqrt{2}$;
 2: $H \leftarrow$ new min-heap ($mindist$ to $q'$ as key);
 3: $V \leftarrow$ new set;                          ▷ cells of reported points
 4: for all entries $e \in R.root$ do
 5:     insert $\langle e, mindist(q', e)\rangle$ into $H$;
 6: while $H$ is not empty do
 7:     deheap $\langle e, mindist(q', e)\rangle$ from $H$;
 8:     remove each cell $c$ from $V$ satisfying
                $maxdist(q', c) < mindist(q', e)$;
 9:     if $e$ is not covered by the union of cells in $V$ then
10:         if $e$ is a point $p$ then
11:             report $p$ to the client;
12:             $V \leftarrow V \cup \{\mathcal{C}_\lambda(p)\}$;
13:         else
14:             read the child node $CN'$ pointed to by $e$;
15:             for all entries $e' \in CN'$ do
16:                 insert $\langle e', mindist(q', e')\rangle$ into $H$;
```

Memory Usage Optimization The memory usage of the algorithm can be reduced, as some cells $c \in V$ can be removed early without affecting the correctness. The basic idea is that if c does not intersect any entry in heap H, it can be removed safely from V. However, this checking is computationally expensive, requiring search of all of H.

We thus propose a lazy approach for eliminating unnecessary cells from V. The min-heap has the property that any entry/point discovered later will be at least the last deheaped distance $mindist(q', e)$ away from q' [2]. Therefore, if a cell c has its maximum distance $maxdist(q', c)$ smaller than the deheaped distance $mindist(q', e)$, then c cannot intersect with points/entries found in the future. The condition is easy to check as $mindist(q', e)$ is already known. In the above algorithm, this checking is implemented in Line 8 in order to eliminate unnecessary cells from V and reduce the memory usage.

Example Figure 8b illustrates the use of granular NN search on the example in Figure 8a. The root node of R-tree contains the three entries e_1, e_2, e_3, each of which points to a leaf node. Each cell is marked by a bold label c_i. The algorithm first examines the root the R-tree and inserts entries e_1, e_2, e_3 into heap H. Next, e_1 is deheaped and its child entries p_1, p_2, p_3 are inserted into H. Then, p_1 is found and reported, and its corresponding cell c_3 is added to V. Next, p_2 is found and reported, and its cell c_1 is inserted into V. When point p_3 is deheaped, it is discarded because it falls in a cell (i.e., c_1) in V. Similarly, entry e_2 is discarded, as it is covered by the union of cells c_1 and c_3 in V. After that, e_3 is deheaped and its child entries p_6, p_7 are inserted into H. Cell c_1 (and c_3) is removed from V, as it cannot intersect any point or entry encountered in the future. The algorithm continues until H becomes empty or it is terminated by the client.



(a) locations of points

Operation	Heap H	Cell list V
Expand root	e_1, e_2, e_3	$\varnothing$
Expand e_1	p_1, p_2, p_3, e_2, e_3	$\varnothing$
Report p_1	p_2, p_3, e_2, e_3	c_3
Report p_2	p_3, e_2, e_3	c_3, c_1
Discard p_3	e_2, e_3	c_3, c_1
Discard e_2	e_3	c_3, c_1
Expand e_3	p_6, p_7	$\varnothing$
Report p_6	p_7	c_6
Discard p_7	$\varnothing$	c_6

(b) execution steps

Fig. 8. Granular INN Example

C. Granular kNN Search

We observe that Lemma 2 is applicable to kNN search as well. The basic idea is to keep k points in each cell and discard any other points in the cell. To accomplish this, Algorithm 2 is modified as follows. First, each cell $c \in V$ is associated with a counter $cnt(c)$. Second, in Line 12, we check whether the cell $\mathcal{C}_\lambda(p)$ already exists in V. If so, we increment its counter; otherwise, we insert the cell (with counter value 1) into V. Third, in Line 9, we only consider the cells with a counter value of k.

V. Parameter Selection

This section presents detailed guidelines for the user to set the parameters in our algorithms: (i) the error bound ϵ, and (ii) the anchor q'.

It is natural to set the error bound $\epsilon = v_{max} \cdot \Delta t_{max}$, according to the maximum speed v_{max} of the user and the maximum travel time delay Δt_{max} acceptable by the user.

For instance, a typical value for Δt_{max} may be 5 minutes and the value of v_{max} depends on the user's transportation method (e.g., walking, cycling, driving).

We proceed to choose an appropriate anchor q' in two steps. The first step is to decide on an anchor distance $dist(q, q')$. In a second step, the anchor q' is set to a random location at distance $dist(q, q')$ from q.

A simple guideline for the first step is based on the privacy requirement of the user—the privacy value Γ is at least $dist(q, q')$, as we will see in the experimental section.

An alternative guideline is based on the budget for the communication cost. Assume that N data points (on the server) are distributed uniformly in the square 2D space with extent U. Recall from Section IV that, the extent of a grid cell equals $\lambda = \epsilon/\sqrt{2}$ and at most k points are returned from each grid cell. Thus, the maximum number of points received by the client is at most $N_\epsilon = \min\{N, 2k \cdot (U/\epsilon)^2\}$. The kNN distance $R_{k\text{NN}}$ is then estimated as [23]:

$$R_{k\text{NN}} = U \cdot \sqrt{\frac{k}{\pi \cdot N_\epsilon}} \tag{5}$$

As shown in Figure 3b, the demand space (supply space) is a circular region centered at query point q (anchor q'). Assume that the client applies the algorithm in Section III-B with the default termination condition. Let m be the number of packets retrieved from the server and β be packet capacity. In this case, the final demand space covers k points and its radius is estimated as $R_{k\text{NN}}$. The final supply space contains $m\beta$ points and its radius is estimated as $R_{k\text{NN}} + dist(q, q')$. Under a uniform data distribution, both the demand space and the supply space have the same density of points, and thus we have:

$$dist(q, q') = \frac{U}{\sqrt{\pi \cdot N_\epsilon}} \cdot (\sqrt{m\beta} - \sqrt{k}) \tag{6}$$

VI. Empirical Evaluation

In this section, we denote our solution by GST (Granular SpaceTwist) to emphasize that both the client-side algorithm and the server-side granular search algorithm are used. We compare with the transformation-based approach in Section VI-A and the spatial cloaking approach in Section VI-B. Section VI-C studies the performance of GST with respect to various parameters.

The studies utilize (randomly generated) uniform (UI) datasets and two real datasets: SC[1] (Schools) with 172,188 points, and TG[2] (Tiger Census Blocks) with 556,696 points. The coordinates of points in each dataset are normalized to the square 2D space with extent 10,000 meters. At the server, each dataset is indexed by an R-tree with a 1K byte page size. Table I summarizes the parameters (with default values in bold) used in the experiments. In each experiment, we use a workload with 100 uniformly random generated query points and measure the average value of the following performance metrics:

- Communication cost, in numbers of TCP/IP packets[3].
- Measured result error, defined as the result kNN distance minus the actual kNN distance.
- Privacy value of the inferred privacy region (Equation 3).

TABLE I

Parameter Values

Parameter	Values
Error bound (meter), ϵ	0, 50, 100, **200**, 500, 1000
Anchor distance (meter), $dist(q, q')$	50, 100, **200**, 500, 1000
Number of required results, k	**1**, 2, 4, 8, 16
Data size (million), N	0.1, 0.2, **0.5**, 1, 2

A. Comparing with Transformation-Based Matching

We compare with the following solutions: (i) SHB [11], which finds k nearest neighbors along a Hilbert curve, (ii) DHB [11], which performs search along two orthogonal Hilbert curves. According to [11], the level of the Hilbert curve used is fixed to 12 for both SHB and DHB. The previous theoretical transformation-based study [10] does not cover implementation details and is not tested here.

Table II compares the result error of SHB, DHB, and GST for different values of k. For uniform data (UI), the Hilbert transformation approach (SHB and DHB) is quite accurate. Since DHB employs two Hilbert curves, it is more accurate than SHB. The accuracy of GST remains acceptable, being much better than the specified error bound (ϵ = 200).

TABLE II

Result Error Versus k

k	*UI, N=0.5M*			*SC*			*TG*		
	SHB	**DHB**	**GST**	**SHB**	**DHB**	**GST**	**SHB**	**DHB**	**GST**
1	7.1	2.2	51.3	1269.3	753.7	2.5	1013.9	405.8	16.1
2	9.3	4.0	49.0	1634.3	736.2	2.6	1154.6	548.7	16.7
4	13.2	6.0	47.6	1878.5	810.9	2.6	1182.3	596.5	17.0
8	19.0	7.3	42.0	2075.6	864.5	2.6	1196.2	599.7	16.3
16	27.0	10.3	36.3	2039.6	985.7	2.6	1199.6	603.2	14.5

For the real-world datasets (SC and TG), both SHB and DHB compute results with poor accuracy because the Hilbert curve does not completely preserve spatial proximity. In contrast, GST benefits from the skew in the data to achieve the best accuracies. This is so because data points in the same grid cell (as illustrated in Figure 7b) are likely to fall in the same cluster and the distances between them are significantly lower than the worst case distance bound (i.e., the diagonal grid cell distance). GST is more accurate on SC than on TG because SC is more skewed.

Regarding the communication cost, DHB has to transfer $2 \cdot k$ Hilbert values (which fit in a single packet for k from 1 to 16) from the server to the client, across all data distributions. As we will see in Section VI-C, GST has comparable communication cost for k from 1 to 16. On the SC (TG) dataset,

[1] U.S. Board on Geographic Names, `http://geonames.usgs.gov/index.html`

[2] R-tree portal, `http://www.rtreeportal.org/`

[3] The packet capacity β is set to (576-40)/8=67, since a 2D data point takes 8 bytes, a packet has a 40-byte header, and the typical value of a Maximum Transmission Unit (MTU) over a network is 576 bytes.

GST additionally incurs as few as 1 (3) packet(s) for much more accurate results than those obtained by DHB. On the UI dataset, the total communication cost of GST is bounded by 5 packets.

In summary, GST is robust and achieves stable result errors across different data distributions. Since the accuracies of SHB and DHB are unacceptable for the real-world data, we exclude them from subsequent experiments.

B. Comparing with Spatial Cloaking

Having confirmed the high result accuracy of GST, we continue to compare GST with the spatial cloaking approach. In keeping with the simple client-server architecture (assumed in our problem setting), we focus on client-based cloaking techniques and disregard techniques that require trusted third-party components or peer-to-peer functionality. For comparison purposes, we implement a prototype cloaking technique, called CLK, that generates the cloaked region as a (randomly generated) square region that contains the exact user location q. The region has an extent of $2 \cdot dist(q, q')$, making its span comparable to the inferred privacy region of GST. The query processing algorithm of [4] is applied on the server to evaluate the cloaked query.

Table IIIa shows the communication cost as a function of $dist(q, q')$, for the two real datasets. As we will see in the next subsection, the value $dist(q, q')$ roughly reflects the privacy value. At high $dist(q, q')$ values, GST incurs much lower communication costs than CLK does. In other words, GST affords high privacy at low communication cost.

TABLE III
IMPACT ON COMMUNICATION COST

	SC		*TG*	
$dist(q, q')$	**CLK**	**GST**	**CLK**	**GST**
50	1.3	1.0	1.9	1.0
100	2.0	1.0	4.6	1.0
200	6.2	1.0	15.0	1.0
500	33.5	1.1	72.8	1.3
1000	107.0	1.4	282.0	2.6

(a) vs. $dist(q, q')$, on real datasets

N	*UI*	
(million)	**CLK**	**GST**
0.1	3.0	1.0
0.2	5.1	1.0
0.5	12.2	1.0
1	23.9	1.0
2	47.5	1.0

(b) vs. N, on UI datasets

Table IIIb compares the communication cost with respect to the data size N using the synthetic UI datasets. Due to the granular search, the cost of GST is independent of N, while the cost of CLK is proportional to the data size.

In terms of result accuracy, CLK always provides exact results (i.e., $\epsilon = 0$). The result accuracy of GST is guaranteed by a user-specified error bound ϵ. In addition, experimental results in Section VI-C suggest that the measured result error of GST is significantly lower than ϵ and acceptably low for a wide range of ϵ values.

From the above experiments, we conclude that CLK does not scale well with the data size and the extent of the cloaked region. Thus, we focus on GST in the sequel.

C. Performance Study of GST

We proceed to investigate the scalability of GST with respect to different parameters, using the two real datasets as well as a synthetic UI dataset of 500,000 points.

Figure 9 depicts the performance of GST when varying the error bound ϵ. As a reference for comparison, the curve for the anchor distance $dist(q, q')$ is included in Figure 9c. As ϵ increases, each grid cell has a larger extent, and fewer points are retrieved, which yields a lower communication cost. But the result error and the privacy value also increase. Since real datasets are skewed, the average error is much smaller than the specified error bound ϵ. At $\epsilon = 0$, granular search is not applied, and exact results are reported. Even for this case, the communication cost and privacy value are both acceptable.

Observe that GST indeed achieves both low communication cost and low result error for a broad range of ϵ values (between 50 and 500). At $\epsilon = 50$, the communication cost is only slightly more than two packets. For the other end (i.e., $\epsilon = 500$), the measured error is acceptably low and stays within 25% of the bound ϵ.

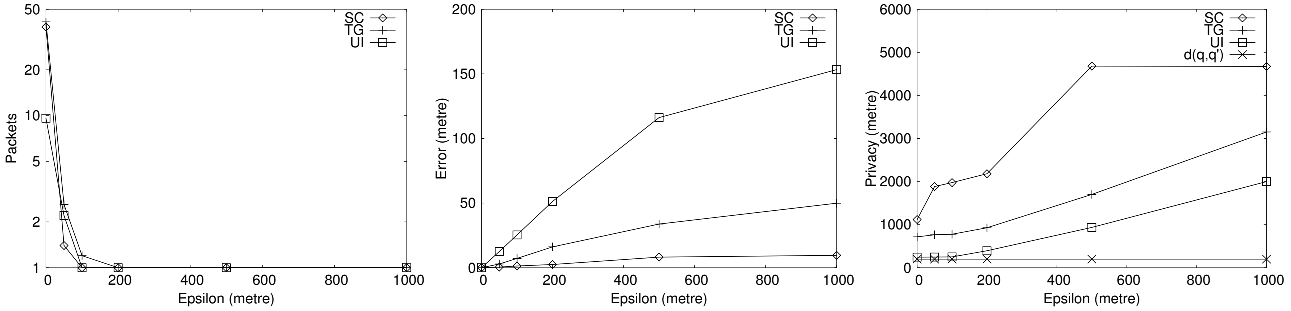
Figure 10 shows the performance of GST as a function of the anchor distance $dist(q, q')$. The communication cost and result error increase when $dist(q, q')$ increases. However, even for large $dist(q, q')$, the communication cost and result error are quite low. Note also that the location privacy afforded the GST is very good, as the privacy value is several times greater than the anchor distance $dist(q, q')$. It is worth noticing that the more the skew, the lower the result error and the higher the privacy value become.

Figure 11 shows the performance of GST with respect to the number of required results k. The communication cost is directly proportional to k, but it remains low for typical values of k. The result error is fairly insensitive to k, but benefits from skew in the data. When k increases, the kNN distance (the middle term in Equation 2) increases much faster than the final supply space radius (the rightmost term). Thus, the inferred privacy region Ψ becomes smaller and the privacy value decreases. Nevertheless, for high values of k, the privacy value is still much larger than the specified anchor distance.

Finally, we study the performance of GST with respect to the dataset size N using synthetic UI datasets. Figure 12 plots the results. Since the error bound ϵ is fixed, the communication cost, result error, and privacy are insensitive to N. Thus, GST scales well with the dataset size.

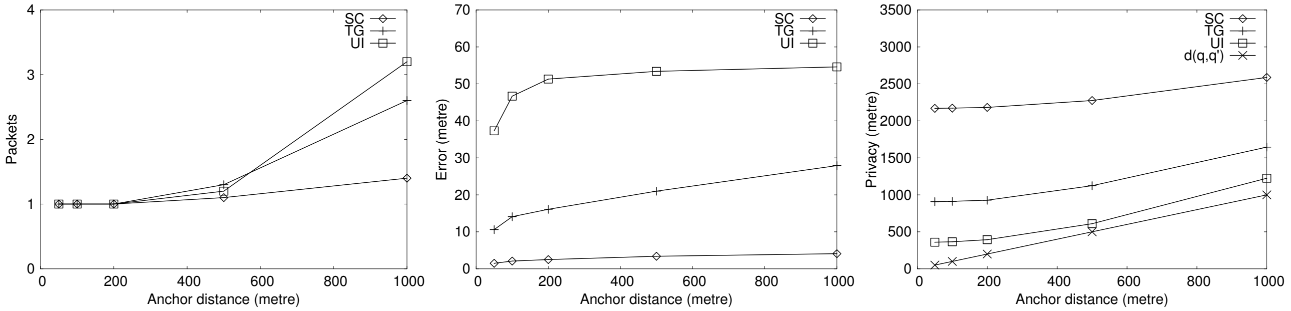
VII. DISCUSSION OF OUR PRIVACY MODEL

Comparison with K-anonymity Location K-anonymity [3], [13] is an oft-used model for specifying the location privacy required by a user. A K-anonymous region (say, Q') is safe in the sense that the user's location q cannot be distinguished from those of $K - 1$ other users in Q', even if the adversary is able to somehow determine the exact locations of all users. However, current works on location K-anonymity demand additional system resources from trusted third parties (e.g., location anonymizer [3], [4], peer users [8], [17], [18]). In addition, the privacy offered to a user may be questionable — Q' is very small when the other $K - 1$ locations in it are very close to the user's location.



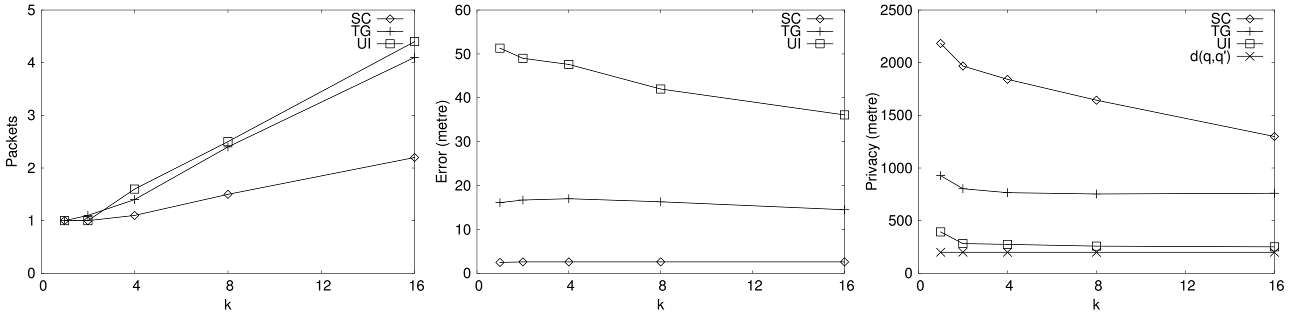
(a) communication cost (b) result error (c) privacy value

Fig. 9. Performance Vs. Error Bound ϵ, $dist(q, q') = 200$, $k = 1$



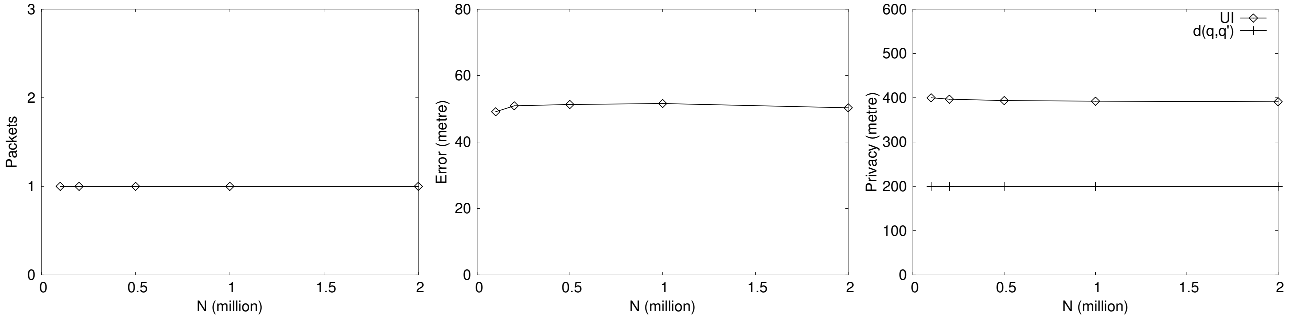
(a) communication cost (b) result error (c) privacy value

Fig. 10. Performance Vs. Anchor Distance $dist(q, q')$, $\epsilon = 200$, $k = 1$



(a) communication cost (b) result error (c) privacy value

Fig. 11. Performance Vs. Number of Required Results k, $dist(q, q') = 200$, $\epsilon = 200$



(a) communication cost (b) result error (c) privacy value

Fig. 12. Performance Vs. Data Size N, $dist(q, q') = 200$, $\epsilon = 200$, $k = 1$, on UI datasets

Our privacy model does not require the knowledge of all users' locations. It is applicable in simple client-server architectures, with no need for any third-party services. Our privacy model is not as strong as K-anonymity in the sense that when the adversary knows all user locations, it may be possible to determine the actual query user location from the inferred privacy region Ψ (derived in Section III-C).

Hence, K-anonymity solutions and our solution work in different settings and offer different kinds of privacy guarantees. We thus advocate our solution as a practical, readily deployable solution for client-server architectures.

Shape of Privacy Region The inferred privacy region Ψ may have an irregular shape, as shown in Figure 6a: within region Ψ, there may be a "clump" in the neighborhood of the actual user location q. This may give hints to the adversary for performing an informed guess of the user's location. This problem can be alleviated by using a large packet size β. This way, the precise termination point of our technique is concealed among β points in the final packet. Another possibility is to perform the search with a coarser granularity, as in Figure 6b. Additional studies of these aspects are in order.

Extension for Advanced Constraints and Preferences Our definition of privacy value (see Equation 3) is built on the basic assumption that each location in the inferred region Ψ has the same probability of being the actual user location. It is an interesting topic to enrich our privacy model with complex features: (i) spatial domain constraints (e.g., excluding low density regions such as forests and lakes from the space) and (ii) user preferences (e.g., a user requires low privacy at work and high privacy when visiting a clinic).

VIII. Conclusion and Research Directions

This paper concerns the efficient support for location privacy protection for location-based service users. Existing location privacy solutions either incur high server load, require specialized server implementations, or produce results without practical guarantees on accuracy bounds of query results.

Motivated by this, the paper proposes a novel and effective framework, called SpaceTwist, that consists of a client-side processing algorithm and a server-side granular search technique that supports user-defined (relaxed) query accuracies. SpaceTwist offers systematic support for managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. Empirical studies with real-world datasets demonstrate that SpaceTwist is capable of providing high degrees of location privacy as well as very accurate results at low communication cost.

Several promising research directions exist. First, it is relevant to extend the cost model (in Section V) to cover real data distributions, as the current model assumes uniform data and may not accurately reflect the distributions found in real-world data. Second, our proposal considers snapshot k nearest neighbor queries. It is of interest to extend them to support also continuous queries [20]. Third, it is possible to apply SpaceTwist to queries over data in road networks, as its correctness (by Lemma 1) only requires the triangular inequality to hold, which is the case for road network distance.

Acknowledgments

This research was conducted as part of the Streamspin project, which is part of Center for Software-Defined Radio, funded in part by the Ministry of Science, Technology and Innovation and Ministry of Economic and Business Affairs, Denmark.

References

[1] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest Neighbor Queries," in *SIGMOD*, pp. 71–79, 1995.

[2] G. R. Hjaltason and H. Samet, "Distance Browsing in Spatial Databases," *TODS*, 24(2): 265–318, 1999.

[3] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The New Casper: Query Processing for Location Services without Compromising Privacy," in *VLDB*, pp. 763–774, 2006.

[4] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing Location-Based Identity Inference in Anonymous Spatial Queries," *IEEE TKDE*, to appear.

[5] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," in *USENIX MobiSys*, pp. 31–42, 2003.

[6] B. Gedik and L. Liu, "Location Privacy in Mobile Systems: A Personalized Anonymization Model," in *ICDCS*, pp. 620–629, 2005.

[7] H. Kido, Y. Yanagisawa, and T. Satoh, "An Anonymous Communication Technique using Dummies for Location-based Services," in *IEEE International Conference on Pervasive Services*, p. 1248, 2005.

[8] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Services," in *GIS*, pp. 171–178, 2006.

[9] M. Duckham and L. Kulik, "Simulation of Obfuscation and Negotiation for Location Privacy," in *COSIT*, pp. 31–48, 2005.

[10] P. Indyk and D. Woodruff, "Polylogarithmic Private Approximations and Efficient Matching," in *Theory of Cryptography Conference*, pp. 245–264, 2006.

[11] A. Khoshgozaran and C. Shahabi, "Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy," in *SSTD*, pp. 239–257, 2007.

[12] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar, "Preserving User Location Privacy in Mobile Data Management Infrastructures," in *Privacy Enhancing Technologies*, pp. 393–412, 2006.

[13] L. Sweeney, "k-Anonymity: A Model for Protecting Privacy," *International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5): 557–570, 2002.

[14] J. Xu, J. Du, X. Tang, and H. Hu, "Privacy-Preserving Location-based Queries in Mobile Environments," *Technical Report, Hong Kong Baptist University*, 2006.

[15] C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. di Vimercati, and P. Samarati, "Location Privacy Protection Through Obfuscation-Based Techniques," in *DBSec*, pp. 47–60, 2007.

[16] M. Duckham and L. Kulik, "A Formal Model of Obfuscation and Negotiation for Location Privacy," in *PERVASIVE*, pp. 152–170, 2005.

[17] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "PRIVÉ: Anonymous Location-Based Queries in Distributed Mobile Systems," in *WWW*, pp. 371–380, 2007.

[18] G. Ghinita and P. Kalnis and S. Skiadopoulos, "MobiHide: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries," in *SSTD*, pp. 758–769, 2007.

[19] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," in *SIGMOD*, pp. 47–57, 1984.

[20] C.-Y. Chow and M. F. Mokbel, "Enabling Private Continuous Queries For Revealed User Locations," in *SSTD*, pp. 258–275, 2007.

[21] A. Okabe, B. Boots, K. Sugihara, and S. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd ed. Wiley, 2000.

[22] N. Koudas, B. C. Ooi, K.-L. Tan, and R. Zhang, "Approximate NN Queries on Streams with Guaranteed Error/Performance Bounds," in *VLDB*, pp. 804–815, 2004.

[23] S. Berchtold, C. Böhm, D. A. Keim, F. Krebs, and H.-P. Kriegel, "On Optimizing Nearest Neighbor Queries in High-Dimensional Data Spaces," in *ICDT*, pp. 435–449, 2001.