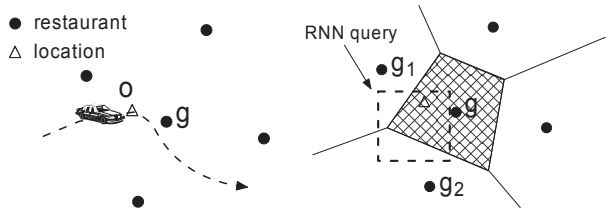
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

2PASS: Bandwidth-Optimized Location Cloaking for Anonymous Location-Based Services

Haibo Hu, and Jianliang Xu, *Senior Member, IEEE*

I. Introduction

Location-based services (LBS) are mobile content services that provide location-related information to users. However, in order to enjoy such services, the mobile user must explicitly expose his/her accurate location to the server. A typical example is a k-nearest-neighbor (kNN) query shown in Fig. 1(a). The user o sends out his/her accurate location and asks for the nearest restaurant. Upon receiving the kNN query, the server returns the name and address of the nearest restaurant (which is g), and other up-to-minute information, such as menu, table reservation status and customer reviews.



(a) Nearest Neighbor Query (b) Range Nearest Neighbor Query

Fig. 1. Location Privacy in kNN Queries

Mobile users see their location privacy compromised in exchange for services (e.g., finding the nearest restaurant). This issue has been receiving arising concerns from both the research community and the public [4], [23], [21], particularly when some locations (e.g., clinics, police stations) can lead to sensitive information such as medical conditions and legal affairs. To address this issue, an intuitive solution is to cache the whole dataset on the mobile device and resolve location-based queries locally. However, due to limited resources of the mobile device, this solution is neither scalable to large datasets nor robust to data updates [25]. Thus, researchers have recently been interested in developing *online* privacy-aware data access techniques [15], [12], [30], [3], [8], [26]. Along this line, *location cloaking* has been proposed to blur the user locations when they request services [15], [12], [23], [19]. The idea is to replace the accurate user location in the request with a well-shaped *cloaked region* (usually a circle or a rectangle), according to some privacy metric such as granularity [23], [27], [17], [22] (the area of this region must exceed a threshold) or K-anonymity [15], [12] (this region must contain at least K users). A kNN query with such a cloaked region is called a *k-range-nearest-neighbor* (kRNN) query [16], and the server returns to the user the kNNs of all points inside this region. Finally, the user refines the genuine kNNs from the kRNN query results, based on the accurate user location. In other words, to protect location privacy, the user requests a *superset* of kNN results from the server, thereby trading network bandwidth for location privacy. Fig. 1(b) shows a 1-range-nearest-neighbor (RNN) query (with the dashed-line box as the cloaked region) when location cloaking is applied to the NN query in Fig. 1(a). In this example, the server returns not only the genuine result g for the NN query, but also g_1 and g_2, because they are the NNs for some points in the cloaked region.

In effect, location cloaking achieves privacy at the cost of requesting non-result objects (e.g., g_1 and g_2) together with their contents (e.g., menu, customer reviews). Since the contents are usually web pages that include texts, images and even videos, their sizes could be significant. Moreover, the larger is the cloaked region, the more privacy is preserved, but the contents of more non-result objects are requested. Requesting these contents waste precious network bandwidth, consume device battery, and charge the user more than necessary. Therefore, an important issue is how to control location cloaking in order to minimize the number of non-result objects. This issue is especially critical in a client-server environment where the mobile user cannot rely on any trusted third party for location cloaking and service request. Whereas a number of location cloaking algorithms have been proposed for different privacy metrics (e.g., [15], [12], [23], [19], [9], [32]), their objective is always to minimize the size of the cloaked region, and thus only indirectly minimizing the bandwidth. Yet an integration of location cloaking with subsequent service request has not been explored in the literature. For example, in Fig. 1(b) if the user knows more restaurants are in the west than in the east of this user, the cloaked region could move more towards the east to avoid receiving too many objects during the kRNN search.

In this paper, we develop an innovative result-aware location cloaking approach for client-server environments, called *2PASS* (2-Phase Asynchronous Secure Search), based on a notion of *Voronoi cells*. Voronoi cells are a set of disjoint space partitions such that 1) each object corresponds to a Voronoi cell; and 2) each object is the nearest neighbor of any point in its Voronoi cell [5]. For example, the shaded region in Fig. 1(b) is the Voronoi cell of g; and the nearest restaurant is g for any point in this cell. If the user knows the Voronoi cells in advance, he/she can set the cloaked region to the Voronoi cell of the nearest neighbor object (that is, the user requests only this object). This is the best a user can get for an NN query — the cloaked region spans the entire cell, the result object is requested, and no non-result objects are requested —

Haibo Hu and Jianliang Xu are with the Department of Computer Science, Hong Kong Baptist University.
E-mail: {haibo,xujl}@comp.hkbu.edu.hk

Digital Object Indentifier 10.1109/TPDS.2010.26

a saving of 67% in bandwidth usage compared to the RNN approach in Fig. 1(b). 2PASS works under the granularity metric, which is the predominant privacy definition in client-server environments, for its simplicity and user-friendliness.[1] If a single cell still does not meet the privacy requirement, that is, the area of this cell is smaller than the threshold, then the cloaked region must span more cells, which means the user must request more objects. To minimize the number of non-result objects, we reduce this problem to the $\mathbb{k}$-minimum spanning tree ($\mathbb{k}$-MST) problem and provide an efficient and yet close-to-optimal algorithm to select objects to request.

Furthermore, due to limited storage capacity, the client may not have the complete Voronoi cell information cached locally. Thus, our 2PASS approach processes a kNN query in two phases. In the first phase, the client requests the Voronoi cell information relevant to the query, based on which it selects objects to request in the second phase. Since the delivery of such information also consumes network bandwidth, its structure must be kept concise. In this paper, we propose a WAG-tree index that encapsulates the Voronoi cell information. This index possesses the following features: 1) by requesting pieces of this index, the user does not suffer from any privacy loss, i.e., the server gets no additional information about the user location, except that he/she is in one of the Voronoi cells of requested objects; and 2) the index is lightweight, i.e., not only its size is small so as to save bandwidth, but also the user can easily compute the requested objects from it. Based on the WAG-tree index, we develop the client and server procedures for the 2PASS approach, which are also lightweight in terms of CPU cost.

To summarize, our contributions in this paper are as follows:

- To the best of our knowledge, this is the first study that explores the connection between location cloaking and bandwidth usage of requested services.
- Based on the notion of Voronoi cells, we develop an innovative 2PASS location cloaking approach that aims to minimize the bandwidth while still protecting user privacy. Through analysis and experiments, this approach is shown to significantly outperform existing approaches and remain robust against various threat models.
- 2PASS is applicable to the granularity privacy metric, and can be extended to other common types of location-based services. A case study application is developed to demonstrate the feasibility of 2PASS.
- We design a lightweight WAG-tree index that facilitates the access of the Voronoi cell information for 2PASS. We show that the bandwidth overhead incurred by this index is negligible compared with the bandwidth it saves from non-result objects.

The rest of the paper proceeds as follows. Section II reviews existing work on privacy-aware location-based services. Section III presents the proposed 2PASS approach for NN queries, followed by Section IV that extends this approach to kNN queries and service allocation queries in general. Further discussions on the privacy protection and security of the 2PASS approach are presented in Section V. The experimental results are shown in Section VI. Section VII describes a case study of the 2PASS approach, followed by the concluding remarks in Section VIII.

[1] K-anonymity, on the other hand, requires a trusted third party to monitor and collect location information of all users.

II. Related Work

Privacy awareness in mobile computing and location-based services has been extensively studied in the recent literature. The objective is to allow the mobile user to request services without compromising his/her privacy, especially location privacy. Various privacy protection techniques have been proposed. Based on the underlying methodologies, these techniques can be divided into three categories: *pseudonym*, *dummy*, and *cloaking*. Pseudonym decouples the mapping between the user identity and the location so that an untrusted server only receives the location without the user identity [24], [4]. However, such a technique is limited to those location-based services that do not require the user's identity. In particular, the lack of user identity makes the billing of these services impossible. Dummy generates fake user locations (called dummies) and mixes them together with the genuine user location into the request [20]. However, by monitoring long-term movement patterns of the user, the server may distinguish the genuine location from dummies. You *et al*. enhance this technique by generating consistent movement patterns for dummies in a long run [33]. Ghinita *et al.* proposed a novel framework that is based on Private Information Retrieval (PIR) [14]. The framework partitions the space into grid cells and then the user requests the content of the cell where he/she is located. Thanks to PIR, the user can hide which cell is requested while receiving the correct content. By setting the content of a cell to its range nearest neighbors, this framework can support NN queries. The framework guarantees the user to disclose nothing to the server, but at a high cost. To guarantee perfect privacy, this framework needs a large number of modulus bits to ensure PIR is computationally secure. This leads to significant overhead in terms of both computational and communication costs, compared with location cloaking techniques. For large datasets where the content size of an object can easily reach kilo- or even mega-bytes, PIR becomes unaffordable or as costly as requesting the entire dataset.

Cloaking has attracted intensive research as a solution to privacy protection. Gruteser and Grunwald were the first to propose spatio-temporal cloaking [15], where a trusted middleware generalizes (i.e., cloaks) the spatial and temporal extents of the user location so that this user satisfies K-anonymity, i.e., he/she is indistinguishable from at least $K-1$ other users with the same generalized location. More specifically, the middleware indexes all user locations using a quad-tree. Upon receiving a request, the middleware traverses the quad-tree until the smallest quadrant that contains the user of this request and other $K-1$ users is found. This quadrant is the cloaked region for this request. Gedik and Liu considered a personalized anonymization model and proposed "CliqueCloak," which constructs a clique graph to combine clients that can share the same cloaked region [12], [13]. Mokbel *et al*. proposed the Casper framework for location-based spatial

Abbreviation	Full Term
WAG	Weighted Adjacency Graph
MVWCC	Minimum Valid-Weight Connected Component
VWCC	Valid-Weight Connected Component
$\mathbb{k}$-MST	$\mathbb{k}$-Minimum Spanning Tree
$\mathbb{k}$-ST	$\mathbb{k}$-Spanning Tree
RNN	Range Nearest Neighbor

TABLE I
GLOSSARY OF ABBREVIATIONS

queries [23]. A grid-based cloaking algorithm was suggested on the anonymizer for both the K-anonymity and granularity metrics. Chow *et al*. studied location cloaking in a peer-to-peer environment [9]. The main idea is to let the client form a group from his/her peers via multi-hop communication. The cloaked region of any subsequent request is then a region that covers all peers in this group. Kalnis *et al*. identified *reciprocity*, a sufficient property for spatial K-anonymity [19]. They proposed two cloaking algorithms, namely, nearest neighbor cloaking (NNC) and Hilbert cloaking (HC, which satisfies reciprocity). In HC, all user locations are sorted by Hilbert space-filling curve ordering, and then users are grouped together in this order. Xu and Cai recently studied location cloaking based on historical trajectories for continuous location-based services [31] and mobile ad-hoc networks [32]. We also studied location cloaking for continuous LBS and proposed two cloaking algorithms that are aware of user mobility pattern and resist trace analysis attacks [30].

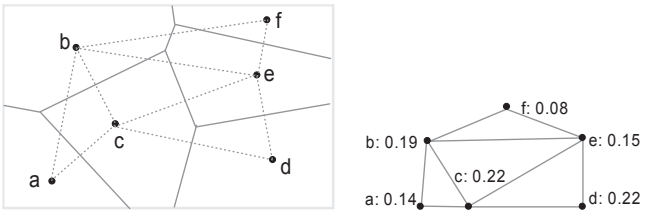
Our research is essentially a result-aware location cloaking approach. It is different from existing cloaking approaches in that the objective is to minimize the number of requested objects, as this determines the cloaking overhead such as bandwidth usage, device energy consumption, and operational charge from a user's perspective.

III. 2PASS FOR NN QUERY

In this section, we first introduce some preliminaries and the system model, followed by an overview of the 2PASS approach. Then we thoroughly study the client-side and server-side procedures in this approach. Finally, we propose the WAG-tree index which scales up the approach to large datasets. For the ease of presentation, we list in Table I the abbreviations used in this paper.

A. Voronoi Diagram and Weighted Adjacency Graph

Given a set of n objects, a *Voronoi diagram* divides the space into n partitions [5]. Each partition is called a Voronoi cell and corresponds to one object. The cell is in such a shape that the nearest neighbor of any point in this cell is the corresponding object. Fig. 2(a) shows an example of Voronoi diagram with 6 objects a through f. The solid lines show the borders of Voronoi cells, and the dotted lines connect the objects whose cells are adjacent. These latter lines divide the space into partitions of a special shape — triangles. As such, the set of these lines is called *Delaunay triangulation* of the space. It is noteworthy, however, if the space is bounded, these dotted lines might not form a closed Delaunay triangulation because two cells might share a border at somewhere beyond



(a) **Voronoi Diagram** (solid lines are cell borders, dotted lines are Delaunay triangulation) (b) **Weighted Adjacency Graph**

Fig. 2. Voronoi Diagram and WAG

the bounded space. For example, in the rectangular space of Fig. 2(a), there is no dotted line between objects a and d, because their Voronoi cells are not adjacent in this space, although they share a border outside the space.

We design a weighted undirected graph to store the Voronoi diagram and Delaunay triangulation. As shown in Fig. 2(b), each vertex in this graph denotes an object, and each edge denotes a line in the Delaunay triangulation. Each vertex i is also assigned a non-negative weight w_i (to be detailed in the next subsection). We call this graph a *weighted adjacency graph* (WAG) in the sequel. It is noteworthy that a WAG is a special weighted graph, because its vertices, instead of its edges are weighted.

B. System and Privacy Model

In this subsection, we describe the system model and privacy assumptions. As shown in Fig. 3, a mobile user wants to request a location-based service (e.g., finding the nearest restaurant) from the LBS server. The traditional location cloaking approaches protect location privacy as follows. Before requesting the service, the user should invoke location cloaking, which obtains for this user a cloaked region that satisfies the privacy metric (step ①). In this paper, we focus on the *granularity* privacy metric, so location cloaking generates a random cloaked region that encloses the user's genuine location and whose area is no less than a user-specified threshold τ. The user then attaches this region (e.g., [22.30N114.18E, 22.31N114.20E] by the longitude-latitude coordinates), instead of the accurate location, in the service request (step ②). Upon receiving this request, the server processes it and returns the resulted objects (step ③).

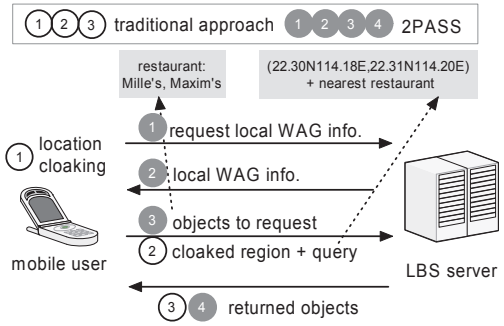


Fig. 3. System Model

Fig. 3 also shows how 2PASS differs from traditional cloaking approaches. In 2PASS, the cloaked region is not blindly generated without knowing the dataset; rather, 2PASS is aware of the spatial locations of the objects and directly requests

contents of result objects from the server instead of sending out an explicit cloaked region. To achieve this, 2PASS works in two phases. In the first phase (steps ❶❷), the client requests from the server a WAG of its neighborhood area, where the weight of a vertex is the area of the corresponding Voronoi cell. In the second phase (steps ❸❹), the client selects objects from this WAG (e.g., two restaurants Mille's and Maxim's) and requests them for their complete contents (e.g., map, customer reviews, and reservation status). Without additional information, the server can only know that the client is in the (implicit) cloaked region implied by these requested objects. In this sense, the client controls the objects to be returned and minimizes their number and hence the total bandwidth usage while still satisfying the privacy requirement.

In general, 2PASS follows a simple client-server architecture and is oblivious to the underlying type of service. The rest of this section starts with NN queries, followed by kNN queries in Section IV-A and other query types in Section IV-B kNN.

C. Overview of 2PASS

Based on the Voronoi cell information, 2PASS requests the objects (including the genuine NN together with other non-result objects) to satisfy the privacy requirement on the cloaked region, which is implied by these requested objects. 2PASS is unique in that the client controls what objects to request from the server so that their total number and thus the overall bandwidth are minimized. To minimize the object number while still meeting the privacy threshold τ, the criteria of object selection are a combination of the following: (1) the sum of the areas of Voronoi cells from the selected objects must exceed τ; (2) the genuine nearest neighbor o^* must be selected; and (3) these Voronoi cells must be connected, i.e., no cell is isolated from the rest of the cells. The last criterion guarantees the cloaked region is a single region, which is a common assumption in all existing location cloaking approaches [15], [12], [23], [19], [9]. Besides, the single-region assumption not only adapts to most location-based services which readily accept a single location as the input, but also alleviates some security problems. For example, a single region is more resilient than isolated regions against background or domain knowledge attacks. With the introduction of WAG, the above object selection is equivalent to finding a subgraph in the WAG that satisfies the following criteria: (1) the sum of the weights of vertices in the subgraph must exceed τ; (2) o^* must be in the subgraph; and (3) this subgraph must be a connected component. In the sequel, we call such a subgraph a "valid-weight connected component" (VWCC) of a query, and the objective of 2PASS is to find a VWCC with the minimum number of vertices. We formalize this problem as follows:

Problem 3.1: **Minimum Valid-Weight Connected Component (MVWCC) Problem**: Given a WAG $\mathcal{G}$, the privacy threshold τ, and the genuine NN object o^*, the problem is to find a VWCC that has the minimum number of vertices.

A naive solution to this problem is to grow all connected components starting from o^* and stop when a VWCC is found with no other VWCC having fewer vertices. To achieve this, we use a priority queue to sort all growing components by the numbers of vertices, breaking tie by the sum of vertex weights. Each time we pop up the top component in the queue and insert components grown from it by appending one adjacent vertex. The algorithm terminates when the first VWCC is popped up. To prevent components from being grown more than once, any component must be checked for duplicates in the queue before it is inserted.

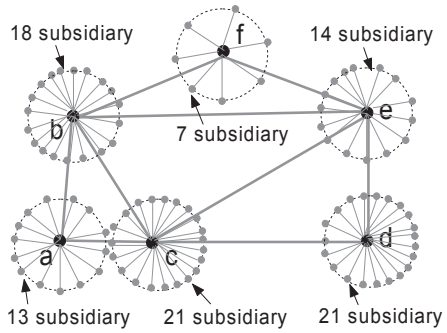


Fig. 4. Reduce MVWCC Problem to $\mathbb{k}$-MST

D. Approximate MVWCC Algorithm

Obviously, the naive solution is inefficient when τ is large, because in the worst case all components whose sums of weights are less than τ are grown. In this subsection, we present an efficient approximation algorithm with a constant bound of approximation ratio. A key observation is that MVWCC problem is very similar to the rooted $\mathbb{k}$-minimum spanning tree ($\mathbb{k}$-MST) problem. A $\mathbb{k}$-spanning tree ($\mathbb{k}$-ST) is a spanning tree with at least $\mathbb{k}$ vertices.

Problem 3.2: **$\mathbb{k}$-MST Problem:** Given a weighted undirected graph and a vertex r, the problem is to find a $\mathbb{k}$-spanning tree rooted at r that spans at least $\mathbb{k}$ vertices with the minimum sum of edge weights.

We show that the MVWCC problem can be reduced to a $\mathbb{k}$-MST problem in polynomial time. The key idea is to construct a new edge-weighted graph $\mathcal{G}'$ from the WAG $\mathcal{G}$. Initially, $\mathcal{G}'$ has the same sets of vertices and edges as $\mathcal{G}$, with each edge assigned a unit weight. Then for each vertex (called "primary vertex") in $\mathcal{G}'$, we add a number of auxiliary vertices (called "subsidiary"). Each subsidiary only connects to its primary vertex via an edge with a weight of 0. The number of such subsidiary vertices for each primary vertex i, denoted by p_i, is almost proportional to its weight w_i in $\mathcal{G}$: $p_i = w_i * \Delta - 1$, where Δ is a constant called scaling factor. Fig. 4 illustrates the $\mathcal{G}'$ constructed from the WAG $\mathcal{G}$ shown in Fig. 2(b).

On the other hand, we can also construct an MVWCC Γ in $\mathcal{G}$ from a $\mathbb{k}$-MST Γ' in $\mathcal{G}'$, where $\mathbb{k} = \tau * \Delta$: for each primary vertex in Γ', we add it to Γ. The following theorem proves that the resulted Γ is an MVWCC. In the sequel, we call $\mathbb{k}$-MST and MVWCC the "dual" of each other.

Theorem 3.1: Γ, the dual of a $\mathbb{k}$-MST Γ' in $\mathcal{G}'$, is an MVWCC in $\mathcal{G}$.

Proof: First, we prove that the sum of weights in Γ is no less than τ. According to the definition of $\mathbb{k}$-MST, the number of vertices in Γ' is at least $\mathbb{k} = \tau * \Delta$. In addition, since the weights of the edges that connect a primary vertex and its

subsidiaries are all 0, if this primary vertex is in Γ', all its subsidiaries must also be in Γ'.[2] Therefore,

$$\tau * \Delta = \Bbbk \leq \sum_i (p_i + 1) = \sum_i w_i * \Delta = \Delta \sum_i w_i.$$

That is, $\sum_i w_i \geq \tau$.

Second, we prove Γ is the VWCC with the minimum number of vertices. Suppose, by contradiction, that there were another VWCC Γ^* with fewer vertices, then its dual must have less weight than Γ' in $\mathcal{G}'$ because this dual must have fewer primary vertices in $\mathcal{G}'$ than Γ', and its sum of weights is exactly the number of primary vertices minus 1 (because all such edges have unit weights). Furthermore, this dual is a $\Bbbk$-ST because it has vertices no fewer than $k = \tau * \Delta$. Therefore, this dual is the $\Bbbk$-MST in $\mathcal{G}'$, which contradicts the assumption that Γ' is the $\Bbbk$-MST.

Finally, obviously, Γ is a connected component in $\mathcal{G}$. ■

Similarly, a VWCC can also be constructed from a $\Bbbk$-ST by adding all primary vertices in the $\Bbbk$-ST to this VWCC. The following corollary shows that $\Bbbk$-ST and VWCC are the dual of each other.

Corollary 3.2: Γ, the dual of a $\Bbbk$-ST Γ' in $\mathcal{G}'$, is a VWCC in $\mathcal{G}$.

Since the $\Bbbk$-MST problem is NP-complete [28], Theorem 3.1 essentially proves that the MVWCC problem is also NP-complete. Furthermore, the following theorem further proves that any algorithm with an approximation ratio to the $\Bbbk$-MST problem has the same approximation ratio to the MVWCC problem.

Theorem 3.3: The dual VWCC Γ of a $\Bbbk$-ST Γ' with α approximate ratio to the optimal $\Bbbk$-MST Γ'_* is also α-approximate to the MVWCC Γ_*.

Proof: Let W denote the weight of $\Bbbk$-ST in $\mathcal{G}'$ and P denote the number of vertices of VWCC in $\mathcal{G}$. By definition, $W_{\Gamma'} \leq \alpha W_{\Gamma'_*}$. We also have

$$P_{\Gamma_*} - 1 = W_{(\Gamma_*)'} \leq W_{\Gamma'_*}$$

On the other hand, $P_\Gamma - 1 = W_{\Gamma'}$. Therefore, $P_\Gamma - 1 \leq \alpha(P_{\Gamma_*} - 1) \longrightarrow P_\Gamma \leq \alpha P_{\Gamma_*}$. ■

The above theorem applies only when the scaling factor Δ is selected in such a way that $\forall i$, p_i, the number of subsidiaries, is exactly $\Delta * w_i - 1$. However, in practice, w_i, the weight of primary vertex i in WAG, is a continuous value, so even a large scaling factor Δ cannot guarantee $\Delta * w_i$ is an integer for all i. Furthermore, the time complexity of state-of-the-art $\Bbbk$-MST approximation algorithms [11], [6], [1] are polynomial to the number of vertices in $\mathcal{G}'$, which is proportional to Δ. So it is advisable to use a medium Δ and let the number of subsidiaries p_i rounded as $\lceil \Delta * w_i \rceil$-1. This is equivalent to rounding each w_i in the WAG $\mathcal{G}$ to w'_i where $w'_i = w_i \frac{\lceil \Delta * w_i \rceil}{\Delta * w_i}$. We call this modified WAG the "integral WAG" (denoted by $\overline{\mathcal{G}}$) while the original WAG the "fractional WAG". Let δ denote the maximum ratio of w'_i/w_i, i.e., $\forall i, w_i \leq w'_i \leq \delta w_i$. Then we have the following corollary on the approximate ratio of the integral MVWCC problem.

[2]If not all subsidiaries are in Γ', we can add them to Γ' with no additional costs.

Corollary 3.4: Let Γ' denote an α-approximate $\Bbbk$-ST to the $\Bbbk$-MST Γ'_* on the integral $\overline{\mathcal{G}}'$. Then on the fractional $\mathcal{G}$, the dual VWCC Γ of Γ' is $\alpha\delta$-approximate to the MVWCC Γ_*.

Proof: By definition, $W_{\Gamma'} \leq \alpha W_{\Gamma'_*}$. We also have

$$P_{\Gamma_*} - 1 = \sum_{i \in \Gamma_*} \Delta w_i - 1 \geq \frac{1}{\delta} \sum_{i \in \Gamma_*} \Delta w'_i - 1 \geq \frac{1}{\delta} W_{\Gamma'_*}.$$

On the other hand,

$$P_\Gamma - 1 = \sum_{i \in \Gamma} \Delta w_i - 1 \leq \sum_{i \in \Gamma} \Delta w'_i - 1 = W_{\Gamma'}.$$

Therefore, $P_\Gamma - 1 \leq \alpha\delta(P_{\Gamma_*} - 1) \longrightarrow P_\Gamma \leq \alpha\delta P_{\Gamma_*}$. ■

This corollary suggests that Δ does not need to be large: as long as it keeps δ from being too large, the approximate ratio $\alpha\delta$ will remain low. At runtime, the client can tune Δ according to its computational resources — a smaller Δ leads to a lower time complexity and a worse approximate ratio, and vice versa. In particular, when Δ is smaller than the inverse of the highest weight, the constructed $\mathcal{G}'$ contains no subsidiary and equivalently the WAG degenerates to an unweighted graph. Finally, the pseudo-code for the approximate MVWCC is listed in Algorithm 1. As for the $\Bbbk$-MST approximation algorithm, we adopt Garg's 3-approximation algorithm due to its constant approximate ratio ($\alpha = 3$) and ease of implementation [11]. The time complexity of Garg's algorithm is $O(N^2 \log N)$, where N is the number of vertices in the graph. Given n vertices and normalized weights in $\mathcal{G}$, graph $\mathcal{G}'$ has at most $N = \delta n$ vertices. Therefore, the time complexity for the approximate MVWCC algorithm is $O(\delta^2 n^2 \log^2 \delta n)$.

Algorithm 1 Approximate MVWCC

Input: $\mathcal{G}$: the WAG
o^*: the genuine NN
τ: the privacy threshold
Output: Γ: the selected VWCC
Procedure:
1: choose scaling factor Δ;
2: build integral $\overline{\mathcal{G}}'$;
3: Γ' = Garg_approximate_$\Bbbk$-MST($\overline{\mathcal{G}}'$,o^*);
4: build Γ from Γ';

E. WAG-Tree and 2PASS

In the first phase of 2PASS, the client requests for the WAG of its neighborhood. However, the definition of neighborhood is not clarified. If the object dataset is not huge, the client can request the WAG of the entire space and cache it to avoid re-request for subsequent queries. However, for a practical dataset with thousands or even millions of objects, it is impractical to request and cache the entire WAG due to the following reasons: (1) the WAG is huge in terms of memory footprint; (2) the cached WAG is vulnerable to even a slight change of the dataset, e.g., an object deleted/inserted/moved; (3) the computational cost of the approximate MVWCC algorithm on the entire WAG is high. As such, we propose to partition the entire WAG into *WAG snippets* of reasonable sizes so that the client receives only the snippet(s) surrounding the query location. In essence, a WAG snippet is the WAG of a subspace. For example, in Fig. 5 the four snippets are obtained

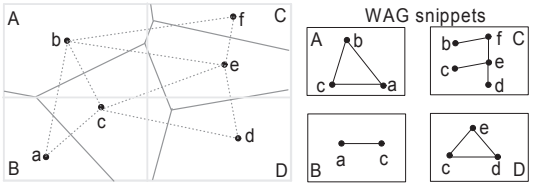


Fig. 5. WAG Snippet

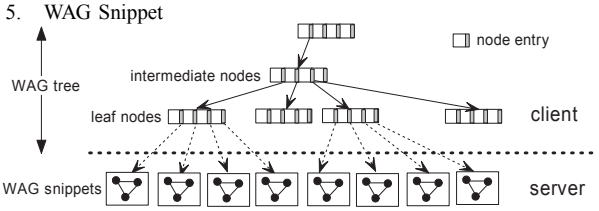


Fig. 6. Illustration of WAG-Tree

by partitioning the space in Fig. 2(a) into four sub-spaces (A, B, C, D) of equal widths and heights and computing their WAG's respectively.[3] It is noteworthy that an object that is outside of a sub-space can still appear in the WAG snippet of this sub-space, as long as the Voronoi cell of this object in the WAG of the entire space overlaps this sub-space, e.g., objects a and c in snippet A. The weight of an object in a WAG snippet is set to its Voronoi cell area that resides in this sub-space. WAG snippets can be *joined* to become the WAG of the union of these sub-spaces. The join is done by merging the vertices corresponding to the same object and assigning its new weight as the sum of the weights of these vertices.

In order for the client to know which snippet(s) to request in the first phase of the query, we build a hierarchical index called WAG-tree. Like a quad-tree, this index recursively partitions the space into quadrants until a certain criterion is met (discussed in Section III-F). Each entry in its leaf node points to a WAG snippet. Note that since the WAG-tree contains no WAG snippets, the size of this index is extremely small. Fig. 6 illustrates a WAG-tree and WAG snippets pointed by it.

Algorithm 2 Client NN Procedure for 2PASS

Input: q: the query point
τ: the privacy threshold
Output: o^*: the genuine NN
Procedure:
1: **if** WAG-tree is not available **then**
2: obtain WAG-tree from server;
3: $\mathcal{S}$ = {the snippet contains q};
4: **while** area($\mathcal{S}$) $< \tau$ **do**
5: move one level up the tree and add snippets to $\mathcal{S}$;
6: request $\mathcal{S}$ from the server;
7: join the received $\mathcal{S}$ into $\mathcal{G}$;
8: find o^* from $\mathcal{G}$;
9: Γ = Approximate_MVWCC($\mathcal{G}, o^*, \tau$);
10: request complete records of the objects appear in Γ from the server;
11: return o^* and its attributes to the user;

The whole 2PASS procedure is summarized in Algorithm 2. During the system initialization, the whole WAG-tree is sent to and cached on the client. Upon an NN query, the client looks up the WAG-tree and locates the snippet that contains the query point. If the area of this sub-space is still smaller than the user-specified privacy threshold τ, the client will locate the lowest-level ancestor node of this snippet whose sub-space area just exceeds τ, and request all snippets rooted at this node. In the sequel, we call all these snippets *host snippets*. The client then joins the received host snippets into a single WAG and applies the approximate MVWCC algorithm on it. The complete records of the objects that appear in the result VWCC are requested in the second phase. Note that throughout the procedure, the client does not send its location or the privacy threshold τ to the server.

[3]For clarity of presentation, the figure does not show the weights of the vertices.

F. WAG-Tree Construction and Maintenance

The construction of the WAG-tree follows a top-down recursive fashion. For each node, the algorithm maintains objects whose Voronoi cells in the whole space overlap this sub-space. Since each such object is the nearest neighbor of some point in this sub-space, it is essentially the range nearest neighbor (RNN) of this sub-space [16]. Obviously, any RNN of a child node must be an RNN of its parent, as the child sub-space is contained by the parent sub-space. As such, by initially setting the entire object dataset as the RNNs of the root node, we can recursively compute the RNNs of a child node among the RNNs of its parent. The recursive procedure terminates and then we build the WAG snippet with the current sub-space and RNNs, when a certain criterion is met. For example, in order to reduce I/O cost when accessing a WAG snippet, we can set the criterion as "the WAG snippet can fit into one page". We can also set the criterion to achieve other objectives, such as "the sub-space area is larger than a sufficiently large threshold" so that most client requests only one snippet per query. To build a WAG snippet, we adopt Fortune's algorithm for Voronoi diagram generation [5]. Additionally, each object has also a boolean flag $isBorder$. An object is a border object if its Voronoi cell touches the border of the sub-space. This flag is used to adapt the WAG-tree to kNN queries (see Section IV). Algorithm 3 shows the pseudo-code of the construction process. To initiate the construction, we simply call this algorithm with parameters $\mathcal{R} = \mathcal{D}$ (the entire object dataset), $T = root$, and a termination condition $\mathcal{F}$.

Algorithm 3 WAG-Tree Construction

Input: $\mathcal{R}$: the set of RNNs
T: the root node of tree
$\mathcal{F}$: the termination condition
Output: T: the WAG-tree rooted at T
Procedure:
1: **if** $\mathcal{F}(T)$ is true **then**
2: build WAG snippet using $\mathcal{R}$;
3: **else**
4: partition T into 4 quadrants t_1, t_2, t_3, t_4;
5: **for** each t_i **do**
6: compute the RNN set $\mathcal{R}_i$ from $\mathcal{R}$;
7: call *WAG-Tree-Construct*($\mathcal{R}_i, t_i, \mathcal{F}$);

As for maintenance, the key idea is to maintain the RNN set for each WAG-tree node and WAG snippet. When an object o is inserted, we traverse the WAG-tree bottom up starting from the leaf node n that contains o. We check all WAG snippets rooted at n for whether it should include o as its RNN and rebuild those that do. Then we traverse from n to n's parent node and repeat this process. It is noteworthy

that if an intermediate node does not include o as its RNN, none of snippets rooted at it need to be rebuilt. This serves as an efficient pruning method. The traversal keeps upwards to visit n's ancestors until the current ancestor node no longer includes o as its RNN. Algorithm 4 shows the pseudo-code for object insertion in a recursive fashion. To initiate the maintenance, we first rebuild wag_o, the WAG snippet where o is located, and then call this algorithm with parameters o, $T = n$, and $T' = wag\_o$. It is noteworthy that the third parameter T' serves as a switch for the algorithm: if it is null, the algorithm only rebuilds WAG snippets rooted at T (with recursive pruning); otherwise, after the rebuilding, the algorithm also traverses upwards to T's parent and continues the rebuilding.

Algorithm 4 WAG-Tree Maintenance on Object Insertion

Input: o: the object being inserted
T: the node of tree root
T': the child node from which T is traversed
Output: T: the updated WAG-tree rooted at T
Procedure:
1: **for** each child node E of T except T' **do**
2: recalculate $\mathcal{R}$, the RNN set of E;
3: **if** $\mathcal{R}$ is updated **then**
4: **if** E is a WAG snippet **then**
5: rebuild WAG snippet of E;
6: **else**
7: call *WAG-Tree-Maintain*$(o, E, null)$;
8: **if** T' is not null AND at least one $\mathcal{R}$ is updated **then**
9: call *WAG-Tree-Maintain*$(o, T.parent, T)$;

The procedure for o being removed is similar except that the pruning is based on whether o *was* an RNN of a node. If it was, the RNN set of this node is recomputed from the RNN sets of its child nodes. The correctness is guaranteed by the fact that an RNN of a parent node must be an RNN of at least one child node. It is noteworthy that, the WAG-tree index does not store any WAG snippet or any RNN set, so no object update will invalidate this index. As such, any object update is transparent to the client.

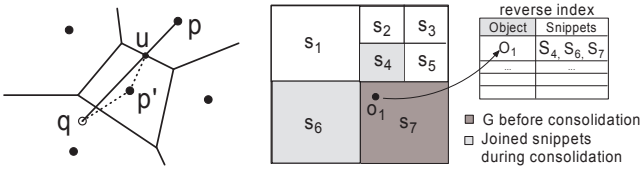
IV. Extensions to kNN and Other Query Types

In this section, we extend 2PASS beyond NN query. First, we will show how to evaluate kNN queries, followed by the generalization to service allocation queries. Finally, we extend 2PASS to achieve objectives other than minimizing the object number.

A. Extension to kNN Query

1) Server-Side Extension: Given a general kNN query, since WAG considers the first NN only, the rest kNNs ($k > 1$) of the query point q may not appear in the host WAG snippet(s) the client requests. As such, it is the server's responsibility to return all snippets (in addition to the requested ones) that may contain kNN results. The following theorem shows that a subgraph of the WAG of the whole space can serve as the boundary for such snippets.

Theorem 4.1: Any k-nearest-neighbor of q must be within $k-1$ hops away from the nearest neighbor of q in the WAG. The proof is straightforward if the following lemma is true.



(a) Illustration of Lemma 4.2 (b) Consolidation

Fig. 7. Extension to kNN

Lemma 4.2: In a Voronoi diagram, the line from q to its k-th ($k \geq 2$) nearest neighbor p, denoted as $\overline{qp}$, must cross only the Voronoi cells of the k nearest neighbors.

Proof: We prove this by mathematical induction. For $k = 2$, if the lemma is not true, then let u denote the point through which $\overline{qp}$ enters the Voronoi cell of p (see Fig. 7(a)), and let p' denote the object adjacent to p, i.e., the object whose Voronoi cell shares u on its border with p. By assumption, p' is neither the NN or the second NN, so p' must be farther to q than p. However, since $|\overline{qp}| = |\overline{qu}| + |\overline{up}| = |\overline{qu}| + |\overline{up'}| \geq |\overline{qp'}|$, p' is closer to q than p, which leads to a contradiction.

If for $k \leq i$, this lemma is true, then it must also be true for $k = i + 1$. Otherwise, $\overline{qp}$ must cross at least the Voronoi cell of one object which is not a j-th NN where $j \leq i$. Let p' denote this object. If p' is adjacent to p, then it is the same as when $k = 2$ and there will be a contradiction on whether p' is closer to q than p. However, even if p' is adjacent to $t \neq p$, the same contradiction appears sooner or later. If t is a j-th NN ($j \leq i$), then the contradiction appears already. So t cannot be a j-th NN. Using this reasoning, we can further deduce that for the object next to t, then next next to t, ..., none of them can be a j-th NN. Since the segment of $\overline{qp}$ is finite, finally the reasoning can reach the object whose next is p. Since this object cannot be a j-th NN either, we have a contradiction exactly the same as when $k = 2$. ■

This theorem shows that for kNN queries, it is sufficient to return the WAG snippets which contain objects that are within $k-1$ hops away from some object in the host snippet(s). To achieve this, we maintain a WAG $\mathcal{G}$ that is initially joined by all host snippet(s). Then we traverse objects in $\mathcal{G}$ in a breadth-first order and stop when all $k-1$ hops of objects from the host snippet(s) are traversed. More specifically, we use a first-in-first-out queue to store objects whose adjacent objects in $\mathcal{G}$ are yet to be traversed. Each object in the queue is also assigned a hop value. Initially, all objects in the host snippet(s) are enqueued with $hop = 0$. We repeatedly dequeue the top object and enqueue its adjacent objects that are not traversed yet with an increment of their hop values. The algorithm terminates when the hop value of the top object is k. During the traversal, when a border object is dequeued, since this object may also appear in other WAG snippets, a "consolidation" process is needed to join all these snippets into $\mathcal{G}$. To facilitate this process, we build a reverse index for WAG snippets that keeps track of, for each object, the snippets where it appears. Finally, the server sends all WAG snippets that are accessed and joined into $\mathcal{G}$ to the client. Fig. 7(b) illustrates this process. Snippets are denoted by s_i and initially the host snippet is s_7, i.e., $\mathcal{G} = \{s_7\}$. When border object o_1 is dequeued, the reversed index is accessed and snippets s_4 and s_6, which are not in $\mathcal{G}$, are loaded and joined.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

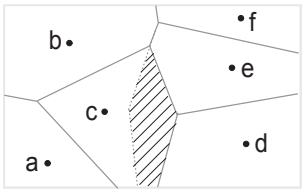


Fig. 8. Narrowing Down Client Location

2) Client-Side Extension: In the first phase of 2PASS, besides the host snippet(s), the client must also provide the value of k to the server. However, by knowing the genuine k, the server further knows that the second, third, ..., k-th nearest neighbors are among the requested objects. This may help the server narrow down the possible client location in the cloaked region. Fig. 8 shows an example where $k = 2$, a, b, c are the requested objects, and thus the cloaked region is their Voronoi cells. In the Voronoi cell of c, however, the shaded area cannot be the client location, because otherwise if the client queried from this area the second NN would be either d or e, which conflicts with the fact that only a, b, c are requested. In fact, this area can be obtained from the Voronoi diagram that excludes object c. Similarly, in Voronoi cells of a and b, there are also areas that cannot be the client location.

To prevent from this, the client should replace k with a larger k' value when sending it to the server. This is subsequently called *k-promotion*. There are several criteria for a good k-promotion algorithm. First, by knowing k', the server should not know the genuine k. Second, furthermore by knowing k', the probability of k being small values (such as $1, 2, 3, \cdots$) must not be negligible, otherwise the server can infer that the second, third, $\cdots$ nearest neighbors are very likely to be among the requested objects, based on which the server may be able to narrow down the possible client location. Last, k' should not be far from k so that not many unnecessary WAG snippets are requested due to k-promotion. As such, we devise a *randomized k-promotion* algorithm that promotes $k = i$ with $k' = j$ with exponentially decreasing probability as j increases. More specifically, the promotion probability $P(j|i) = 2^{(i-j-1)+M}$ ($j \geq i + M$), where M is a constant called *promotion degree*, denoting the minimum promotion. This is a valid probability function because for any i, the sum of probabilities of promoting it to every possible j is 1, i.e., $\sum_{j \geq i+M} P(j|i) = 1$.

Randomized k-promotion satisfies the aforementioned criteria. First, for any $k' = j$, the genuine k can be any integer from 1 to $j - M$. Second, when observing $k' = j$, the probability of $k = i$ is

$$P(i|j) = \frac{P(i,j)}{P(j)} = \frac{P(i)2^{(i-j-1)+M}}{P(j)} \geq \frac{P(i)2^{(i-j-1)+M}}{P(1)}.$$

Here $P(1) \equiv P(i = 1)$, and we assume $P(i)$ is ever-decreasing, that is, in practice the user prefers kNN queries with smaller k; by this assumption, $P(j) = \sum_{1 \leq i \leq j-M} P(i) * P(j|i) \leq P(1)$. As such, the higher is M, the more non-negligible of k being small values. Last, for any $k = i$, the expected value of j, $E(j|i)$ is

$$E(j|i) = \sum_{j \geq i+M}^{\infty} iP(i|j) = i + M + 1.$$

That is, on average, the algorithm promotes k by only $M + 1$. The second and third criteria also show that M is a tradeoff between the effectiveness and cost of the randomized k-promotion algorithm.

The second phase of 2PASS is almost the same as the NN case, except that all the genuine kNN objects, instead of just the genuine NN object o^*, must be in the VWCC. As such, in Algorithm 1 that computes the approximate MVWCC, a multi-root $\Bbbk$-MST approximation algorithm should be used instead of a single-root $\Bbbk$-MST algorithm.

B. Query Generalization

2PASS can be generalized to a wide range of queries or location-based services other than kNN. For example, it can be used to answer "service allocation" queries. In a service allocation problem, each data object (e.g., restaurant, gas station) has a corresponding service area, which is disjoint from those of the others. The query is to find the data object that services the query point. Indeed, NN is a special case of service allocation problem if we consider the Voronoi cell as the service area. Similar consideration can be applied to k-farthest-neighbor queries. All existing 2PASS techniques can thus be applied to service allocation queries except for the WAG construction algorithm, where no more Voronoi diagram needs to be computed, and instead the service area of each object is read directly from external input.

The 2PASS approach can be generalized to applications with objectives other than minimizing the number of requested objects. So far, we minimize this number because we assume that each object has the same data size. However, this approach can still work if the sizes are different and the objective is to minimize the total size of requested objects. The only change needed is the approximate MVWCC algorithm (Algorithm 1). Instead of assigning a unit weight to each edge (u, v) in the graph $\mathcal{G}'$, we should set the weight to the size of object u or v. To be fair with both objects, we set the weight of edge (u, v) to the size of u and the weight of (v, u) to the size of v, and thus makes $\mathcal{G}'$ a directed graph. In addition, the direction of edges that connect primary vertices and auxiliary vertices should be from the former to the latter with zero weight. Finally, a $\Bbbk$-MST approximation algorithm for a directed graph should be used instead of that for an undirected graph.

V. Security Discussions

In this section, we discuss the security aspect of the 2PASS solution. In particular, we identify several threat models from the server (i.e., the adversary) that might compromise the location privacy of the client. Then we show how 2PASS addresses these threats. The discussions in this section will be reinforced by the results of security test in Section VI-B. Throughout this section, we assume that the server knows not only the dataset but also the client-side 2PASS algorithm, including the approximate MVWCC algorithm and its embedded $\Bbbk$-MST algorithm. Other than that, we assume for simplicity that the server has no more background knowledge about the user. The objective of the server is to narrow down the client's location from the cloaked region, i.e., the Voronoi cells of all

requested objects. Meanwhile, we also assume that the client hides several secret values from the server, including: (1) the scaling factor Δ for the approximate MVWCC algorithm, and (2) the promotion degree M for the k-promotion algorithm. We argue that these values are safe from the adversary's speculation as they are completely random and on-the-fly from the client device.

A. Reverse Engineering

The first threat is "reverse-engineering" of the genuine NN object based on the set of requested objects and the approximate MVWCC algorithm. Since the server knows this algorithm and its output, i.e., the requested objects, it might reverse engineer the input, i.e., the genuine NN object, by enumerating every object o in the requested objects, executing the MVWCC algorithm with o as the root, and comparing the output with the set of requested objects. In this way, either the genuine NN can be found or at least those objects whose outputs are different from the set of requested objects can be excluded from being the genuine NN. However, we argue that the server lacks Δ as one of the dominating parameter to execute the same MVWCC algorithm as the client. Since Δ essentially decides the numbers of subsidiaries in the constructed graph $\mathcal{G}'$, the server cannot repeat the same result without knowing Δ.

B. Probability Attack

Following the first threat, the second threat is the "pseudo-randomness" of the genuine NN. Since the objective of 2PASS is to minimize the bandwidth of requested objects, it tends to choose objects with larger weights (e.g., larger Voronoi cells) in the MVWCC algorithm. This may let the server speculate that the query point is not uniformly distributed in the cloaked region, i.e., this distribution is pseudorandom instead of truly random. More specifically, the server may speculate that objects with larger weights are more likely to be "dummy objects" which are chosen merely for satisfying the privacy threshold, whereas objects with smaller weights are more likely to be the genuine NN. However, we argue that 2PASS has the following mechanisms to guard against such threats.

- The scaling factor Δ, which is decided at the client and hidden from the server, controls how weights are converted to the numbers of subsidiaries in $\mathcal{G}'$, so it affects the output of the MVWCC algorithm. Even if an object has a large weight, its corresponding number of subsidiaries in $\mathcal{G}'$ might still be zero if Δ is small enough. In particular, if Δ is smaller than the inverse of the highest weight in WAG $\mathcal{G}$, $\mathcal{G}'$ will contain no subsidiary at all. In this case, the output of requested objects will not depend on the weights anymore.
- 2PASS requires the cloaked region to be a single region. As such, the constituting Voronoi cells must be connected, so they must form a connected component in the WAG. This allows objects with smaller weights to appear in the result VWCC if they are on the path from the genuine object to objects of larger weights.
- Given a uniform distribution of query points, an object with a larger weight naturally has a higher probability of being the genuine NN. This fact undermines the confidence of the speculation that large-weighted objects are dummies.
- We adopt an approximate MVWCC algorithm, instead of an optimal algorithm. So objects with smaller weights are still possible to be included in the result VWCC due to the non-optimality.

C. k Exposure

The third threat is guessing k out of the received k' regarding kNN queries. Although the randomized k-promotion algorithm is used to prevent such guess, the server might still attempt to do so, even if the confidence of such guess is not high. For example, by the promotion algorithm, k is most probably (with a probability of 50%) promoted to $k + M$. As such, the server can simply guess $k = k' - M$. However, since M is unknown, the server must also guess M with various values, which dilutes the confidence of the guess of k.

Furthermore, we also argue that if the size of a WAG snippet is small compared with that of an object, the client can request a sufficiently large number of WAG snippets that contain all genuine kNNs in the first phase. In this way, the client no longer needs to send k'. To define "sufficiently large" with respect to k, we need to know the distance between the query point and the k-th nearest neighbor. The estimation of this distance has been studied in a global dataset [7] or in a local subset [29]. It is noteworthy that in this approach, overestimation is always preferred to underestimation, because failing to receive a WAG snippet that contains a genuine kNN will lead to a second request of more WAG snippets.

VI. Experimental Results

In this section, we evaluate the performance of $2PASS$ approach through extensive experiments. The object dataset we use is a real dataset that contains 123,593 postal addresses in New York, Philadelphia and Boston. For ease of presentation, the coordinates of these objects are normalized to a unit square. We compare $2PASS$ with the existing approach for privacy-aware kNN search, i.e., cloaked-region based kRNN [16], [23], [19], denoted by $Range$. While the cloaked region in $2PASS$ is implied by the Voronoi cells of the requested objects, the $Range$ approach first explicitly prepares a cloaked region that satisfies the granularity metric (i.e., a random square of area τ that encloses the genuine user location q), and then it issues a k-range-nearest-neighbor (kRNN) query with this region. In response, the server returns all kRNNs, i.e., the k nearest neighbors of all points in this region.

We implement a database server in Java on a desktop PC with Pentium D 3.0GHz CPU and 2GB RAM. We also implement a kNN query client in Microsoft .NET Compact Framework on a iPAQ hx2700 Pocket PC running Windows Mobile 5.0 with Marvell PXA270 624MHz processor, 64 MB RAM, and 320MB ROM. The client connects to the server through a 3G-equivalent network: 384kbps downlink speed and 128kbps uplink speed. The query load consists

Parameter	Symbol	Default Value
kNN query	k	1
object size	s	20KB
granularity threshold	τ or tau	0.0001
mean scaling factor	$\overline{\Delta}$ or Delta	$\lceil 1/w_{min} \rceil$
k-promotion degree	M	5
page size		4KB
max area of WAG snippet		0.0001

TABLE II
SIMULATION PARAMETER SETTINGS

of $1,000$ queries that are uniformly distributed in the unit square. We measure the following metrics: query response time, number of requested objects, client/server CPU time, and bandwidth consumption,[4] all of which are averaged over the $1,000$ queries. In addition, we also design several families of privacy attacks on $2PASS$ that aim to narrow down the client location by speculating the genuine NN object of this client (see Section VI-B for more details). We measure the success rates of these attacks as security indicators of $2PASS$. The parameter settings are summarized in Table II.

A. Index Construction and Overall Performance

Fig. 9 shows the comparison between $2PASS$ and $Range$ under default settings. For ease of comparison, we normalize the results of $2PASS$ to 1 and show the ratios of $Range$. To be fair, a WAG-tree index and an R-tree index are built for them respectively. In particular, the termination condition in WAG-tree construction is set as "the max area of a WAG snippet is 0.0001". As is shown in Fig. 9, the WAG-tree index requires about 30 minutes to construct, almost 20 times longer than the R-tree index. This is mainly due to the construction of all WAGs by computing their Voronoi diagrams. However, the resulted WAG-tree index is only 87KB, which is suitable to cache on the client side during system initialization. This index size is far smaller than 3.2MB of the R-tree index,[5] because WAG-tree is merely a partition of the space, not a partition of the objects. Even if we include all WAG snippets stored at the server, the entire storage cost of 2PASS is 14.8 MB only, about three times larger than the R-tree index. This cost is quite reasonable for such a large dataset.

As for query performance, the response time of $Range$ is about twice of $2PASS$, which is mainly due to the more number of objects it requests than $2PASS$. As a consequence, $Range$ also consumes about twice bandwidth than $2PASS$. To show how the object size (s) affects the bandwidth, we vary s exponentially from 1.25KB to 1280KB, the range of a typical web page size, and plot the bandwidth curve in Fig. 10. $Range$ outperforms $2PASS$ only when the object size is extremely small (s=1.25KB). Starting from s=5KB, $2PASS$ costs less than $Range$ because the bandwidth overhead for WAG snippets in the first phase is small and independent of s. The performance gain is more evident as s increases and this overhead is amortized by more bandwidth saved from object request. To confirm this, in the same figure, we also plot the

[4]For $2PASS$, the bandwidth consumption includes those incurred in both phases.

[5]Since the ratio of index size far exceeds 2.5, the maximum x-axis value, this ratio is not shown properly in Fig. 9.

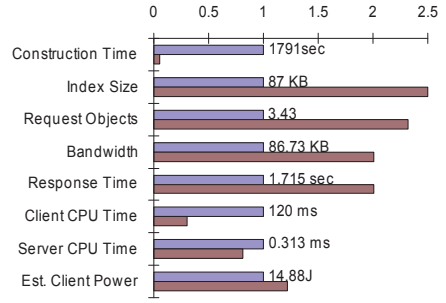


Fig. 9. Overall Performance

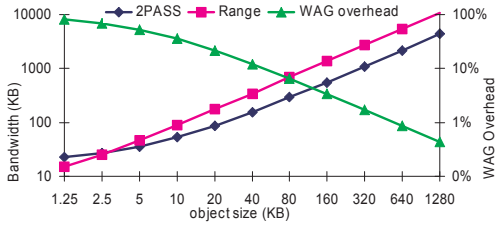
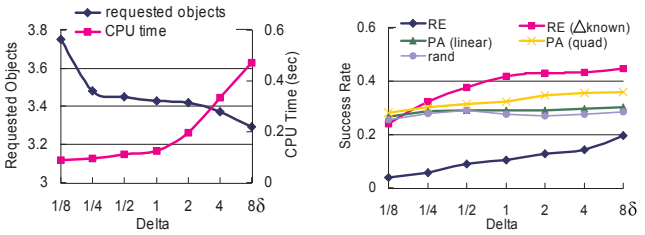


Fig. 10. Bandwidth v.s. Object Sizes

proportional bandwidth for WAG snippets. This overhead is less than 10% when the object size is larger than 40KB. The only metric in which $Range$ always outperforms $2PASS$ in Fig. 9 is the client CPU time, because the client simply issues a range query in $Range$ approach while 2PASS needs to execute an additional approximate MVWCC algorithm (which involves $\mathbb{k}$-MST algorithm) in $2PASS$. However, since we keep the graph that is fed into the $\mathbb{k}$-MST algorithm small, by setting the Δ value to only the inverse of the minimum vertex weight in the WAG, even the CPU time for $2PASS$ is just 120ms, which only accounts for less than 10% of the total response time. We also estimate the power consumption of CPU and bandwidth for both approaches as follows. The power consumed by CPU is estimated by its active power consumption (925 mW based on the Marvell/Intel data sheet [18]) multiplied by the CPU time. The power consumed by network bandwidth is estimated based on the 3G energy model derived in [2], where the energy spent to download/upload x KB data is $0.025x + 3.5 + 0.64t$ Joules. Here t is the total time (in seconds) the 3G module is in the high-power state, which is the transmission time plus a 12.5-second tail time. From Fig. 9, we observe that even though $2PASS$ consumes more energy for CPU computation, its total power consumption is about 20% less than $Range$, due to its less response time and bandwidth.

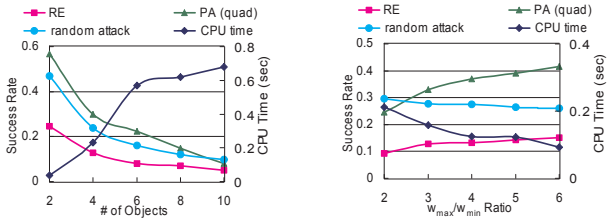
B. Security Test on Δ

In Sections III and V, we show that the scaling factor Δ controls how weights are converted into the numbers of subsidiaries in the approximate MVWCC algorithm. Therefore, it is not only a trade-off between the approximation ratio and computational cost, but also affects how secure the 2PASS is against privacy threats that aim to guess the genuine NN, i.e., the Voronoi cell where the client is located. To guarantee that Δ is random and secret, throughout the experiments we use a normal distribution for Δ and set its mean value, denoted as $\overline{\Delta}$, to $\lceil 1/w_{min} \rceil$, where w_{min} is the minimum weight in the WAG. This default setting (denoted as δ) ensures that on



(a) Client Performance (b) Security Test

Fig. 11. Effect of $\overline{\Delta}$ on Client Performance and Security



(a) v.s. Number of Objects (b) v.s. w_{max}/w_{min} Ratio

Fig. 12. Client CPU Time and Success Rates under $\overline{\Delta} = \delta$

average each primary vertex has at least one subsidiary. In this subsection, we vary $\overline{\Delta}$ with $1/8, 1/4, 1/2, 1, 2, 4, 8$ times of δ and measure the number of requested objects, the client CPU time, and the success rates of three families of privacy attacks, namely, reverse engineering attack (RE), probability attack (PA), and random attack ($rand$). The first corresponds to the attack model of Section V-A and attempts to guess the genuine NN by repeating the approximate MVWCC algorithm and comparing the result with the set of requested objects. However, since Δ is unknown to the adversary, this attack will use the largest $\overline{\Delta}$ the client may choose, i.e., 8δ, to make sure the weight conversion is accurate enough. If two or more objects have the same MVWCC results, this attack breaks tie by a random guess. For reference, we also show the RE attack result when Δ is known to the adversary. The second family of attacks correspond to the attack model of Section V-B and guess the genuine NN based on the probabilities (i.e., weights) of the requested objects, with a bias on lower-weighted objects. Specifically, the probability of guessing o_i is proportional to $w_i/bias_i$. In particular, PA (linear) has a linear bias on the weights, i.e., $bias_i = w_i$, while PA (quad) has a quadratic bias on the weights, i.e., $bias_i = w_i^2$. The last random attack, which serves as the benchmark, makes a random guess of the genuine NN purely based on the probabilities (i.e., weights) of the requested objects.

Fig. 11(a) shows the number of requested objects and CPU time at the client side. As expected, the former metric decreases while the latter increases as $\overline{\Delta}$ grows. In fact, the increase of CPU time is quite moderate until $\overline{\Delta} > 2\delta$, when it becomes sharp. On the other hand, the number of requested objects does not change much in the range of $[\frac{1}{2}\delta, 2\delta]$. Fig. 11(b) shows the success rates of the three families of privacy attacks. We observe that $rand$, PA (linear), and PA (quad) have similar success rates which approximate the inverse of the number of requested objects. This validates our justification in Section V-B that 2PASS tries to retain the uniformity of the genuine location (i.e., the query point) inside a cloaked region. Nonetheless, we do notice that by changing from in proportion to the weights ($rand$) to in equal proportion (PA-linear), and to in inverse proportion to the weights (PA-quad), the success rate increases, but within a 7% bound. This shows that the probability attacks can exploit the pseudo-randomness of the approximate MVWCC algorithm — the more accurate of the weight conversion (i.e., the higher the $\overline{\Delta}$), the more successful the attack is. However, the rate improvement tends to saturate when $\overline{\Delta}$ becomes high, which shows that the MVWCC algorithm has limited pseudo-randomness for the adversary to exploit. On the other hand, for low $\overline{\Delta}$ ($\leq \delta$), the rate improvement is hardly noticeable. We also observe that the reverse engineering attack has a much lower success rate. This is due to the unmatched Δ values used at the client and adversary sides. In many cases, the attack simply fails as the adversary cannot find any result that exactly matches the set of requested objects. Even when $\overline{\Delta} = 8\delta$, i.e., the mean Δ at the client equals to Δ used at the adversary, the success rate is still lower than that of random guess, because the client-side Δ may still deviate from $\overline{\Delta}$ due to its randomness. For comparison, if the exact Δ is known to the adversary, the success rate of RE improves by about 20%. This shows the effectiveness of having a secret Δ at the client. To summarize, the unknown and random Δ poses a great challenge for the adversary to guess the genuine NN from the set of requested objects, either by exploiting the pseudo-randomness or by reverse engineering. Furthermore, the result also shows the feasibility of a moderate $\overline{\Delta}$ (such as δ) that strikes a balance among the number of requested objects, CPU time, and the security.

To further verify this, in the second set of experiments we consistently use $\overline{\Delta} = \delta$ and measure the CPU time and success rates of RE, PA (quad) and $rand$ under kNN queries where k ranges from 1 to 5. To make a fair comparison, the adversary also uses $\Delta = \delta$ in the reverse engineering attack. Fig. 12(a) shows the CPU time and success rates decomposed by the number of requested objects, whereas Fig. 12(b) shows the results decomposed by the ratio of the maximum weight to the minimum weight of the requested objects. In Fig. 12(a), the CPU time increases as the number of requested objects increases, but it quickly becomes saturate as a larger k value can in turn reduce the vertices to explore in the approximate MVWCC algorithm (more on this in Section VI-E). On the other hand, all success rates decrease as the number of requested objects increases. Furthermore, the PA (quad) attack, which performs the best in Fig. 11(b) (except for the RE attack with known Δ), has its success rate drop faster than the other two attacks when the number of requested objects increases. This shows that it is increasingly difficult to exploit pseudo-randomness when more privacy is required, probably due to the connectivity requirement and the non-optimality of the MVWCC algorithm.

In Fig. 12(b), we analyze the performance with regard to various w_{max}/w_{min} ratios. As this ratio increases, the spatial distribution of the objects becomes skew, so does the distribution of their weights. The result shows that the CPU time drops as the distribution becomes skew, because fewer

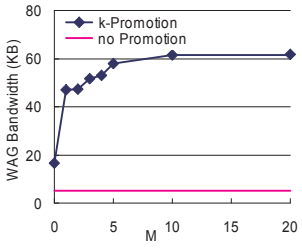


Fig. 13. WAG Bandwidth v.s. M

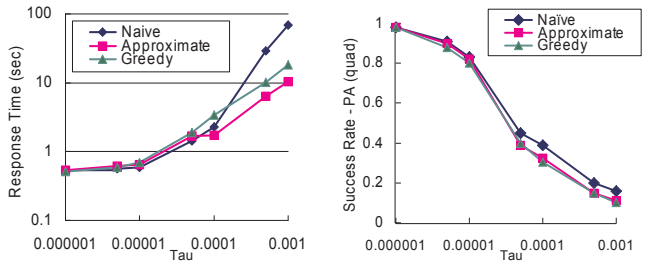
objects are needed to reach the τ threshold. On the other hand, the success rates increase except for that of the random attack. The reason is that as the weight distribution gets skew, the MVWCC algorithm will expose more pseudo-randomness, which can be exploited by PA and RE, but not the random attack. Nonetheless, even for the PA (quad) attack, which enjoys the most of the pseudo-randomness, the rate at the most skew setting (i.e., 6) is less than 10% higher than its overall success rate (ref. Fig. 11(b)). This figure gets as low as 5% for the RE attack. All these results confirm that 2PASS is efficient and secure with $\overline{\Delta} = \delta$, and it works best for high privacy requirement and for little or moderate skewness.

C. Choice of k-Promotion Degree M

In Section IV, we propose a randomized k-promotion algorithm to send a larger k' to the server. In this algorithm, parameter M denotes the minimum possible difference between k and k'. In this subsection, we evaluate impact of choice of M on the $2PASS$ performance. Since M only affects the performance in the first phase, we measure the bandwidth, which is allocated for WAG transmission, in Fig. 13. The bandwidth increases as M increases but it reaches saturation point at $M = 5$. Even when $M = 20$, the bandwidth is only two times higher than $M = 0$ and about one order of magnitude higher than the bandwidth without k-promotion. This is because even M (and thus k') increases, the traversal of objects within $k' - 1$ hops away does not load and join too many new WAG snippets. As such, the randomized k-promotion algorithm costs moderate bandwidth, even when a large M is set.

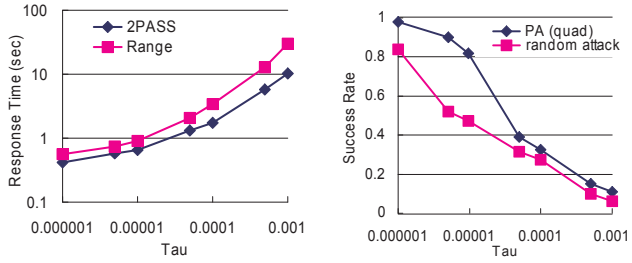
D. Evaluation of Approximate MVWCC Algorithm

In this subsection, we compare the performance of our proposed approximate MVWCC algorithm (Algorithm 1) with the naive algorithm (Section III-C) and a greedy algorithm that always adds to the VWCC result the adjacent vertex that has the maximum weight. Figs. 14(a) and 14(b) show the overall response time of these algorithms and the success rates of privacy attacks under various τ settings. As is shown in the figure, the response time of the naive algorithm is the lowest for small τ values, but the difference gap is negligible. On the other hand, the approximate algorithm always outperforms the greedy and naive algorithms for $\tau \geq 0.0001$, which is due to the fact that the response time of the naive algorithm is dominated by the high client CPU computation while the greedy algorithm requests more objects than the other two. As such, the approximate MVWCC algorithm strikes a good



(a) Response Time (b) Success Rate – PA (quad)

Fig. 14. Approximate MVWCC v.s. Greedy and Naive MVWCC



(a) Response Time (b) Success Rate on 2PASS

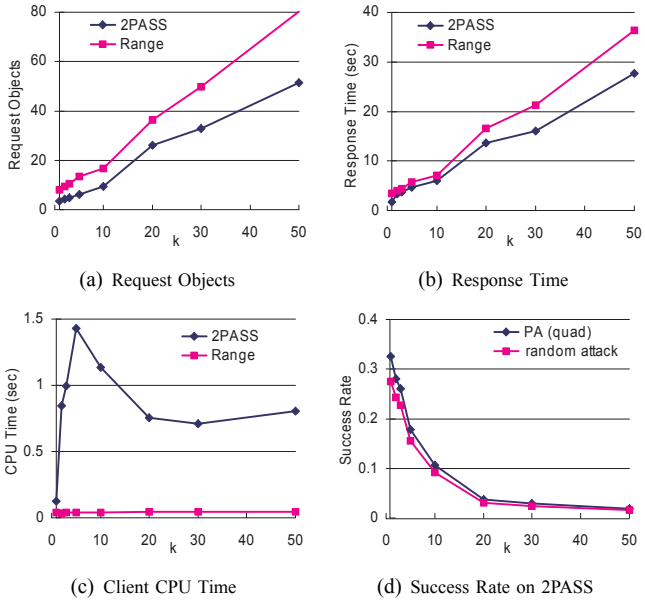
Fig. 15. Comparison under Various Privacy Threshold τ Settings

balance between the approximation ratio and computational cost. Fig. 14(b) confirms this conclusion from another perspective —- the success rate of the PA (quad) attack, where the approximate algorithm also sits in between that of the naive and that of the greedy algorithm. The naive algorithm has the highest success rate because it requests the fewest objects and exhibits the most pseudo-randomness due to its optimality. The approximate MVWCC algorithm, on the other hand, has a lower success rate as a side effect.

E. Scalability

In this subsection, we compare $2PASS$ and $Range$ approaches under various privacy threshold τ and query parameter k settings. Fig. 15(a) shows the response time when τ ranges from 0.000001 to 0.001. In all settings, $2PASS$ outperforms $Range$ by 20% to 70%. It is also noteworthy that the performance gain of $2PASS$ increases as τ increases, which is mainly due to the number of requested objects: the number for $2PASS$ increases linearly as τ increases, whereas the requested objects of $Range$ are the range nearest neighbors, whose number increases dramatically with the area of the query range τ. Fig. 15(b) shows the success rates of random and PA (quad) attacks on $2PASS$. In all settings, the rate of the latter is at most 60% more than the rate of the former, which consistently shows $2PASS$ has limited pseudo-randomness to exploit.

Figs. 16(a) through 16(d) show the comparison when k (for kNN) ranges from 1 to 50. Fig. 16(a) shows that the number of requested objects of $2PASS$ is almost equivalent to k while that of $Range$ increases dramatically as k increases. This is because the first phase of $2PASS$ already loads WAGs that contain the index information for enough (more than k) objects, so the requested objects in the second phase can be as few as k if the privacy threshold is met. However, for $Range$ approach, the number of requested objects increases sharply



(a) Request Objects (b) Response Time

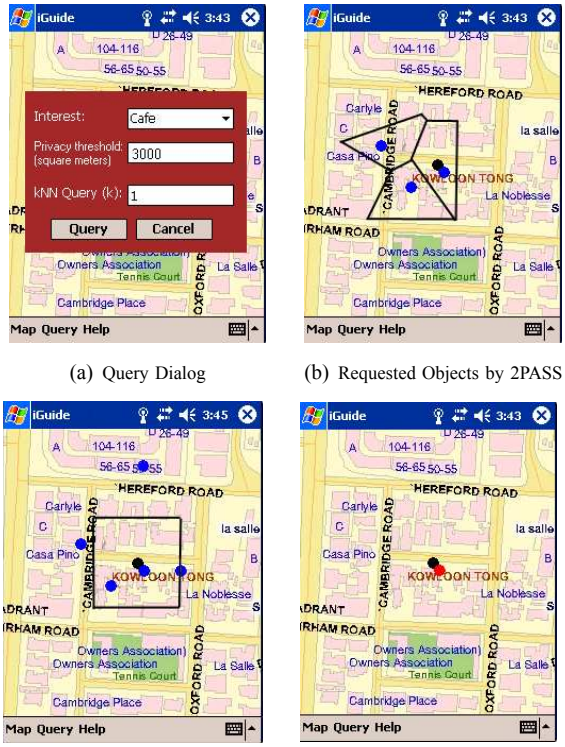
(c) Client CPU Time (d) Success Rate on 2PASS

Fig. 16. Comparison under Various k Values

because the kNNs of all points in the query range can be far more than k. Fig. 16(b) is the direct consequence of Fig. 16(a), where $2PASS$ is consistently better than $Range$ in all k settings in terms of response time. Fig. 16(c) shows the CPU time at the client side where $Range$ has an extremely low cost because all range query processing is at the server side. On the other hand, $2PASS$ requires a non-negligible CPU time for the approximate MVWCC algorithm. However, in general, the CPU cost decreases as k increases, because given τ, fewer vertices need to be explored and added to the result VWCC to count the total weight to τ. It is noteworthy, however, that there is a sharp increase from $k = 1$ to $k = 5$, which we believe is due to the multi-root $\mathbb{k}$-MST algorithm that repeatedly tries every root to find the best one to span. However, even at its peak when $k = 5$, the CPU time is still less than 1.5 seconds, about one-third of the total response time. Fig. 16(d) shows the success rates of random and PA (quad) attacks on $2PASS$. As k increases, the difference between these two rates drops. This is because the larger k, the more irregular of the VWCC returned by the approximate algorithm, and hence the less pseudo-randomness $2PASS$ exhibits. As such, we can conclude that $2PASS$ is a robust and consistently better approach for secure nearest neighbor search under various privacy threshold τ and query parameter k settings.

VII. Case Study: iGuide

We implement the 2PASS client in our existing *iGuide* prototype system which provides location-based information (such as ATM, cafe, and restaurant) to tourists [10]. *iGuide* is a GPS-aware system, so the user is able to issue kNN queries from his/her current position with a GPS-equipped PDA. Fig. 17(a) shows the query dialog that allows the user to specify the dataset, the granularity threshold τ, and the query parameter k. Fig. 17(b) shows the query point (black dot), requested objects (blue dots), and the corresponding Voronoi cells, i.e., cloaked region (in bold lines). As a comparison, we also show the requested objects by range nearest neighbor ($Range$) in Fig. 17(c), where the bold lines constitute the query range (i.e., the cloaked region). The number of requested objects in $Range$ is 5, larger than 3 in $2PASS$. The underlying reason is illustrated in the same set of figures: the cloaked regions in both figures must be no smaller than τ, however, since $2PASS$ is aware of the Voronoi cell of each object, by applying the MVWCC algorithm it reduces the number of requested objects. As the final note, in Fig. 17(d), whichever approach ($2PASS$ or $Range$) is applied, the user is shown only the genuine nearest neighbor (red dot) from his/her position.



(a) Query Dialog (b) Requested Objects by 2PASS

(c) Requested Objects by Range (d) Result Displayed to User

Fig. 17. 2PASS in iGuide System

VIII. Conclusion

In this paper, we investigate the problem of privacy-aware data access in location-based services. We propose an approach called 2PASS that allows the client to control what objects to request in order to minimize their number while not compromising location privacy of the user. The core component of 2PASS is a lightweight WAG-tree index from which the client can compute out the objects to request from the server. Efficient client and server procedures of 2PASS are discussed in detail. Although we use kNN queries as the running type of service to introduce 2PASS, we show that this approach can be generalized to many other location-based services with various objectives.

As for future work, we plan to extend 2PASS beyond the client-server model. Specifically, we are interested in a client-anonymizer-server model where the trusted anonymizer performs location cloaking for the client. 2PASS can therefore be deployed at the anonymizier. By redefining the weights in the WAG, 2PASS can be extended to support other privacy metrics such as K-anonymity.

We also plan to deploy our iGuide system in a larger geographical area and conduct a large-scale user experience survey on the usability of our 2PASS-based system. We also plan to study the behavior of 2PASS when the client issues a series of requests within a short period. There are two contradicting implications. On one hand, a client-side cache may add more indeterminacy to the client behavior and thus makes the server speculation even more difficult. On the other hand, the server might get extra information if it compares the requested objects of two or more consecutive queries. For example, at time $t = 0$ the client requests, among other objects, object o, but at time $t = 1$ second, the client requests neither o nor any object whose Voronoi cell is reachable from the cell of o within this second. As such, the server can confirm that o is a dummy object in the $t = 0$ request and can be removed from speculation. Therefore, the client needs a more consistent object selection algorithm than the approximate MVWCC to ensure no shrinking of cloaked region during continuous service requests.

REFERENCES

[1] S. Arora and G. Karakostas. A 2+ϵ approximation algorithm for the k-mst problem. In *Proc. 11th SODA*, 2000.

[2] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proceedings of the 9th ACM/USENIX conference on Internet Measurement Conference*, pages 280–293, 2009.

[3] B. Bamba, L. Liu, P. Pesti, and T. Wang. Supporting anonymous location queries in mobile environments with privacygrid. In *The 17th International World Wide Web Conference (WWW'08)*, 2008.

[4] A. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[5] M. Berg, M. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 1997.

[6] A. Blum, R. Ravi, and S. Vempala. A constant-factor approximation for the k-mst problem. In *Proc. 28th STOC*, 1996.

[7] C. Bohm. A cost model for query processing in high dimensional data spaces. *TODS*, 25(2):129–178, 2000.

[8] Ying Cai and Toby Xu. Design, analysis, and implementation of a large-scale real-time location-based information sharing system. In *Proceedings of ACM Mobisys*, 2008.

[9] C.-Y. Chow, M. F. Mokbel, and X. Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based services. In *Proc. of ACM GIS*, pages 171–178, 2006.

[10] J. Du, J. Xu, X. Tang, and H. Hu. iPDA: Supporting privacy-preserving location-based mobile services. In *Proc. 8th MDM (demo)*, 2007.

[11] N. Garg. A 3-approximation for the minimum-tree spanning k vertices. In *Proc. 37th FOCS*, 1996.

[12] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *Proc. of ICDCS*, pages 620–629, 2005.

[13] 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.

[14] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. Tan. Private queries in location based services: Anonymizers are not necessary. In *Proc. of SIGMOD*, 2008.

[15] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of MobiSys*, pages 31–42, 2003.

[16] H. Hu and D. Lee. Range nearest neighbor query. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):78–91, 2006.

[17] Haibo Hu, Jianliang Xu, and Dik Lun Lee. PAM: An efficient and privacy-aware monitoring framework for continuously moving objects. *IEEE Transactions on Knowledge and Data Engineering, to appear*, 2009.

[18] Intel. Data sheet of intel pxa270 processor. http://www.phytec.com/pdf/datasheets/PXA270_DS.pdf, 2005.

[19] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference in anonymous spatial queries. *TKDE*, 19(12):1719–1733, 2007.

[20] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *Proc. 2nd ICPS*, pages 88–97, 2005.

[21] Ling Liu. Protecting location privacy in mobile computing systems: Architecture and algorithms (tutorial). In *ACM Mobicom 2007, Montreal, Quebec, Canada*, 2007.

[22] S. Mascetti, C. Bettini, D. Freni, X.S. Wang, and S. Jajodia. Privacy-aware proximity based services. In *Proceedings of the 10th International Conference on Mobile Data Management (MDM)*, 2009.

[23] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: Query processing for location services without compromising privacy. In *Proc. of VLDB*, pages 763–774, 2006.

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

[25] Himanshu Pagey, Kien Hua, and Chow-Sing Lin. Caching as privacy enhancing mechanism in location based services. In *Proceedings of the 10th International Conference on Mobile Data Management (MDM)*, 2009.

[26] L. Pareschi, D. Riboni, and C. Bettini. Protecting users' anonymity in pervasive computing environments. In *Proceedings of the Sixth IEEE International Conference on Pervasive Computing and Communications*, 2008.

[27] Nayot Poolsappasit and Indrakshi Ray. Towards a scalable model for location privacy. In *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*, 2008.

[28] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning trees short or small. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[29] C. Xia, W. Hsu, and M. L. Lee. Erknn: Efficient reverse knearest neighbors retrieval with local knndistance estimation. In *Proc. CIKM*, 2005.

[30] J. Xu, X. Tang, H. Hu, and J. Du. Privacy-conscious location-based queries in mobile environments. *IEEE Transactions on Parallel and Distributed Systems*, to appear.

[31] T. Xu and Y. Cai. Exploring historical location data for anonymity preservation in location-based services. In *IEEE Infocom, Phoenix, Arizona*, 2008.

[32] Toby Xu and Ying Cai. Location cloaking for safety protection of ad hoc networks. In *Proceedings of IEEE Infocom*, 2009.

[33] T. You, W. Peng, and W. Lee. Protect moving trajectories with dummies. In *The International Workshop on Privacy-Aware Location-based Mobile Services*, 2007.

Haibo Hu is an Assistant Professor in the Department of Computer Science, Hong Kong Baptist University. Prior to this, he held several research and teaching posts at HKUST and HKBU. He received his PhD degree in Computer Science from the Hong Kong University of Science and Technology in 2005. His research interests include mobile and wireless data management, location-based services, and privacy-aware computing. He has published 20 research papers in international conferences, journals and book chapters. He is also the recipient of many awards, including ACM Best PhD Paper Award and Microsoft Imagine Cup.

Jianliang Xu is an associate professor in the Department of Computer Science, Hong Kong Baptist University. He received the BEng degree in computer science and engineering from Zhejiang University, Hangzhou, China, in 1998 and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2002. He was a visiting scholar in the Department of Computer Science and Engineering, Pennsylvania State University, University Park. His research interests include data management, mobile/pervasive computing, wireless sensor networks, and distributed systems. He has published more than 70 technical papers in these areas, most of which appeared in prestigious journals and conference proceedings. He has served as a vice chairman of ACM Hong Kong Chapter. He is a senior member of the IEEE.