

## *p*-Sensitivity: A Semantic Privacy-Protection Model for Location-based Services

Zhen Xiao<sup>1</sup>

<sup>1</sup>School of Information  
Renmin University of China  
{xiaozhen,xfmeng}@ruc.edu.cn

Jianliang Xu<sup>2</sup>

Xiaofeng Meng<sup>1</sup>  
<sup>2</sup>Dept of Computer Science  
Hong Kong Baptist University  
xujl@comp.hkbu.edu.hk

### Abstract

Several methods have been proposed to support location-based services without revealing mobile users' privacy information. There are two types of privacy concerns in location-based services: location privacy and query privacy. Existing work, based on location  $k$ -anonymity, mainly focused on location privacy and are insufficient to protect query privacy. In particular, due to lack of semantics, location  $k$ -anonymity suffers from query homogeneity attack. In this paper, we introduce  $p$ -sensitivity, a novel privacy-protection model that considers query diversity and semantic information in anonymizing user locations. We propose a PE-Tree for implementing the  $p$ -sensitivity model. Search algorithms and heuristics are developed to efficiently find the optimal  $p$ -sensitivity anonymization in the tree. Preliminary experiments show that  $p$ -sensitivity provides high-quality services without compromising users' query privacy.

### 1. Introduction

With the advances in wireless communication and mobile positioning technologies, location-based services (LBSs) are becoming increasingly popular for mobile users. While users benefit much from the useful and convenient information provided by LBS, the threat of revealing the user's private information has become a severe issue [3]. In general, there are two types of privacy concerns: 1) *location privacy* (the location information is sensitive, e.g., being in a clinic or pub), and 2) *query privacy* (the query content is sensitive, e.g., querying about political information) [4, 18]. Existing studies mainly focused on location privacy and employed *location  $k$ -anonymity* for privacy protection, which requires each user to be indistinguishable from at least  $k-1$  other users [7, 8, 9, 10, 15]. A widely used technique based on location  $k$ -anonymity is *cloaking*. Specifically, each user location is enlarged into a cloaking region large enough such that each cloaking region covers at least  $k-1$  other users and each user location is covered by at least  $k-1$  cloaking regions. In this way, a user is not identifiable from other  $k-1$  users with respect to location information.

In contrast to existing work that focused on location privacy, in this paper we study protection of query privacy for LBS applications. In many LBS applications, mobile users do not mind to reveal their exact location information. However, they would like to hide the fact that they have issued queries that contain sensitive contents as such information may reveal their personal interest (e.g., searching the nearest clinic when the user is in an insensitive public place). To protect query privacy, a simple approach is anonymization which removes the user id of the query. This approach, however, is vulnerable to attacks with external knowledge. For example, a user's location can be learned using latest network-based positioning technologies (e.g., triangulation [6]). By joining user location data and anonymized queries, an adversary can easily infer the true issuer of a sensitive query.

Existing location  $k$ -anonymity technique can be used to improve protection of query privacy [4, 18]. Nevertheless, the protection provided by location  $k$ -anonymity is not sufficient. Consider a scenario where each query location is enlarged in accordance with  $k$ -anonymity. That is, each query location is covered by at least  $k$  queries (hereafter called *anonymity set*). Thus, even though the adversary knows the exact location of a user, he is not able to link the user to a specific query (rather  $k$  queries). However, one main weakness of  $k$ -anonymity is that it considers only spatial proximity in forming anonymity sets, but not query semantics at all. In an extreme case, if all queries in the anonymity set contain about the same content, the query privacy is still revealed. This situation is not uncommon. For example, when friends meet after office hours and discuss to visit some club, they may all issue location-based queries containing the keyword of club. Since these query locations are spatially proximate, they are very likely to be anonymized together in the same anonymity set. As a result, although the adversary cannot infer which user issued which query, he can know all users queried about clubs. For another example, several speciality clinics are located in a small area of the downtown, and people are easy to lose their way after leaving the highway exit. The users may often issue a location-based query to find out the way to some speciality clinic near the highway exit. These queries are then likely to be anonymized with each other. Furthermore, even if the  $k$  queries in an anonymity set are not of

the same kind (e.g., satisfying  $l$ -diversity in [15]), it is still not acceptable to some users if they all contain sensitive information (e.g., some queries ask about clubs and some others ask about clinics). In a word, due to lack of semantics, location  $k$ -anonymity can just prevent the association between users and requests, but not the association between users and (sensitive) query contents, and hence suffers from the aforementioned attacks.

To address this issue, we propose a new  $p$ -sensitivity model that considers query diversity and semantics in anonymizing user locations. We identify two main attacks against query privacy, namely *location linking attack* and *query homogeneity attack*. To protect against these two attacks, the key concept of our model is to prevent the association between users and query contents in such a way that even if the user location is revealed, the adversary would not be able to infer that the user must have issued a sensitive query with high confidence. We propose a Partition-Enumeration-Tree (PE-Tree) based method to implement the  $p$ -sensitivity model. Search algorithms and heuristics are developed to efficiently find the *optimal*  $p$ -sensitivity anonymization in the tree. Some preliminary experimental results show that  $p$ -sensitivity provides high-quality services without compromising users' query privacy.

The contributions we make in this paper can be summarized as follows:

- To the best of our knowledge, we are the first to identify the query privacy problem for LBS applications in which mobile users are concerned with disclosure of any sensitive queries.
- We propose a novel  $p$ -sensitivity model that considers query diversity and semantics to protect the query privacy.
- We present a PE-Tree to model all possible anonymizations of user queries, based on which search algorithms and heuristics are developed to find the optimal anonymization.
- Some preliminary experiments are conducted to evaluate the performance of our proposed algorithms and heuristics.

The rest of the paper is organized as follows. Section 2 reviews related work on privacy protection in data publication and LBS applications. Some preliminaries, including system architecture, adversary attacks, and the  $p$ -sensitivity model, are presented in Section 3. Section 4 presents our anonymization algorithm based on a PE-Tree. Section 5 provides some preliminary experimental results of our proposed techniques. Finally, Section 6 concludes the paper.

## 2. Related work

In the research of privacy protection for LBS, most existing work just considered location privacy and employed location  $k$ -anonymity model. Although location  $k$ -anonymity

model can be used to preserve query privacy, it has some limitations which are originated from  $k$ -anonymity in privacy protection for data publication. First, a  $k$ -anonymous table may allow an adversary to derive sensitive information of an individual with 100% confidence. Even though an adversary cannot find which tuple among the anonymity set the user corresponds to, if all the tuples have the same sensitive value (e.g., same disease), the privacy of all the corresponding users is disclosed. This homogeneity attack is solved by  $l$ -diversity, which requires not only anonymity but also diversity – all tuples that share same quasi-identifiers should have diverse values for their sensitive attributes [15]. However,  $k$ -anonymity and  $l$ -diversity both do not take into account the semantics of sensitive values thus may cause privacy breach. When considering query privacy in LBS, for the same reason of lack of query semantics, location  $k$ -anonymity can just prevent the association between users and requests, but ignores the association between users and sensitive query contents.

Several studies have attempted to distinguish location privacy and query privacy. For data publication, [13, 17] analyzed two types of information disclosures: 1) *identity disclosure*, which occurs when an individual is linked to a particular record in the released table, and 2) *attribute disclosure*, which occurs when new information about some individuals is revealed, i.e., the released data makes it possible to infer the characteristics of an individual more accurately than it would be possible before the data release. Identity disclosure often leads to attribute disclosure. Once there is identity disclosure, an individual is re-identified and the corresponding sensitive values are revealed. Attribute disclosure, on the other hand, can occur with or without identity disclosure. An observer of a released table may incorrectly perceive that an individual's sensitive attribute takes a particular value, and behave accordingly based on the perception. This can harm the individual, even if the perception is incorrect. [19] proposed personalized anonymity, in which individual's privacy requirement is expressed as some semantic information based on the domain of sensitive attribute values. For example, Andy gets gastric ulcer; however he may not want anyone to think that "Andy must have some stomach problem." Thus, associating Andy with dyspepsia or gastritis (both are stomach problems) is not allowed.

For LBS applications, [4, 18] distinguished user's privacy requirements with respect to location and query content separately. However, [18] still used  $k$ -anonymity to satisfy query privacy requirements, and [4] assumed that the content of a query can be cloaked into a cloaking region, which is not acceptable in our model. [2] discussed the threats based on two types of users' private information: 1) the disclosure of the association between the user's identity and the service information, and 2) the disclosure of the association between the user's identity and the location information. [7] analyzed two attacks: 1) *restricted space identification* (when location  $l$  exclusively belongs to user  $S$ , e.g., by geo-code postal addresses or maps correlations,  $S$  could be correlated with the request from  $l$ ); 2) *observa-*

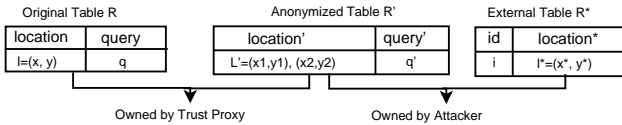
tion identification (when user  $S$  is observed in a location  $l$  thus correlated with the requests from  $l$ ). These two attacks both suppose that the adversary has the knowledge of the correlation between users and locations, and uses this knowledge to correlate users with request. This assumption is shared in our problem formulation. Different from our work, the above studies merely presented the concept but did not develop any solutions. Moreover, they did not consider diversity and semantics of query contents.

Several techniques have been proposed in the literature of LBS with regard to cloaking efficiency and effectiveness, including Quad-Tree [7], Clique Cloaking [9], Hilbert Curve [10], nnASR [10], Grid [14], and mobility-aware cloaking [20]. All these techniques aim to generate an appropriate anonymization of user requests with respect to privacy protection level and/or service quality. In contrast, our algorithm is the first one to get the optimal anonymization with minimal anonymization cost.

### 3. Preliminaries

#### 3.1. System architecture

We consider a centralized LBS system consisting of mobile users, a trusted anonymizing proxy, and an LBS server [4, 7, 9, 14, 18]. Upon a user query, the mobile client first sends the LBS request to the anonymizing proxy through an authenticated and encrypted connection. Upon receiving a batch of user requests, the anonymizing proxy performs location cloaking and forwards the anonymized requests to the LBS server. Finally, the LBS server evaluates the anonymized requests and returns the query results back to the mobile users through the anonymizing proxy, which performs result refinement.



**Figure 1. Original Table, Anonymized Table and External Table.**

Figure 1 shows the formats of user requests during remote communication as well as the external data. For simplicity, we use three tables to describe different data. Table  $R$  stores the original requests from mobile users received by the anonymizing proxy. Each tuple in  $R$  is an LBS request from a mobile user, expressed as:  $r = (id, l, q)$ , where  $id$  is the user’s identifier,  $l = (x, y)$  is the user’s current location, and  $q$  is the query content. These three parameters have different implications. First,  $id$  uniquely identifies a user. It cannot be revealed to any third-party and, hence, should be removed before being forwarded to the LBS server. Second,  $l$  could be a quasi-identifier ( $QI$ ) attribute, which cannot directly identify an user but may reveal a user’s association with requests by joining with external data (e.g., some background knowledge such as yellow pages and location data

obtained by network-based positioning). Thus,  $l$  should be cloaked (enlarged) in the request sent to the LBS server. Third,  $q$  is a sensitive attribute, which may be confidential to an individual (subject to her/his preference) but must be sent to the LBS server in order to answer the request.

Following the above analysis, the trusted proxy should compute an anonymized table  $R'$  such that (1) it contains all the attributes of  $R$  except  $id$ ; (2) it has an anonymized tuple for each tuple in  $R$ ; (3) it preserves as much information of  $R$  as possible; (4) it does not cause any privacy breach based on the user’s privacy requirements (to be discussed below). Each tuple  $r'$  in  $R'$  is an anonymized request corresponding to a tuple  $r$  in  $R$ , defined as:  $r' = (L', q')$ , which contains the user’s cloaking region  $L'$  and the query content  $q'$ , and  $r'.L' \ni r.l$  and  $r.q' = r.q$ .

Table  $R^*$  contains the external data that the attacker can get. Each tuple  $u^*$  in  $R^*$  identifies a user’s location, defined as  $u^* = (id^*, l^*)$ , where  $l^*$  is user  $id^*$ ’s precise location observed by the attacker.

To facilitate our study, we further make several system assumptions as follows:

- *Complete Location Knowledge:* We suppose that in the worst case the attacker owns the accurate location of each user. This is practically possible by employing real-time network-based location positioning.
- *Complete Activity Knowledge:* We assume that the attacker has the background knowledge about all users’ activities. That is,  $R^*$  includes  $l^*$  if and only if this user has sent an LBS request.
- *Equal Background Knowledge:* Without further information, we assume that the attacker does not have extra background knowledge for any particular user.
- *Online Processing:* The trusted proxy adopts online processing, by which the proxy accepts at most one request from a mobile user at one time. That is, the proxy does not accept new requests from the same user until the last one is processed. Thus, a user could appear at most once in each of the three tables.

#### 3.2. Adversary attacks

To protect query privacy, first we define the query semantics. For simplicity, we do not construct a taxonomy for query contents as in [19] because not so many requests arrive in the anonymizing proxy within a short time period under the nature of online processing. Instead, we simply assume that each query can be classified into two types according to its content: (1) *Insensitive Query* ( $Q_i$ ), e.g., queries about traffic; (2) *Sensitive Query* ( $Q_s$ ), e.g., queries about bar, clinic, and political information.

Following our system assumption, the attacker may obtain the tables  $R'$  and  $R^*$  (as shown in Figure 1) and attempt to establish their relationship. There could be two possible attacks: *location linking attack* and *query homogeneity attack*. Below we use an example to illustrate each of these two attacks. We assume that there are six users  $u_1$  through  $u_6$ , as shown in Figure 2.

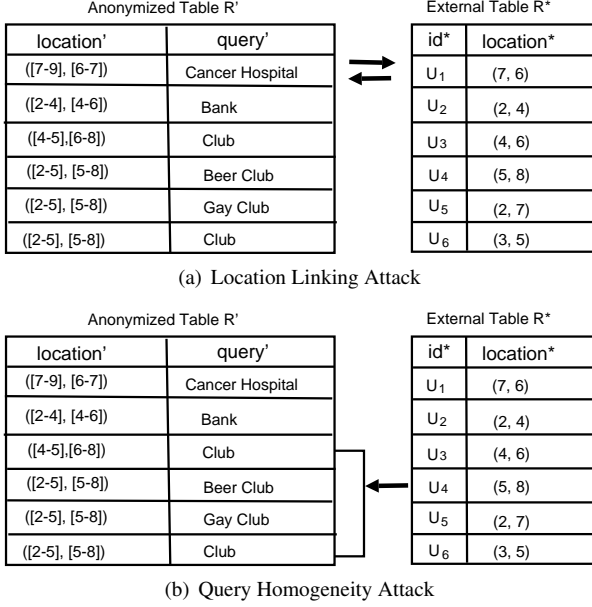


Figure 2. Adversary Attacks.

**Location Linking Attack:** Figure 2(a) gives an example of location linking attack. In the external table  $R^*$ , the public location of user  $u_1$  is  $l_1^* = (7, 6)$ . In the anonymized table  $R'$ , a query is sent with the cloaking region of  $L_1' = ([7, 9], [6, 7])$ . When  $L_1'$  is joined with  $R^*$ , the attacker observes that  $L_1'$  covers only the location of one user  $u_1$  and, hence, infers that the query “Cancer Hospital” must have been sent by  $u_1$ . Similarly, given  $l_1^*$ , after joining it with  $R'$ , the attacker observes that  $l_1^*$  is covered only by the cloaking region of one request and, hence, infers that  $u_1$  must have sent the query “Cancer Hospital”. In a word, the attacker can use location information to link a sensitive location-based query to the issuer or link a user to the issued sensitive location-based query.

**Query Homogeneity Attack:** Figure 2(b) shows an example of query homogeneity attack. In the external table  $R^*$ , user  $u_4$  has the public location of  $l_4^* = (5, 8)$ . When  $l_4^*$  is joined with  $R'$ , the attacker can observe that  $l_4^*$  is covered by the cloaking regions of four requests, each of which covers more than one location in  $R^*$ . Then the attacker can only know that  $u_4$  has sent one of the four queries but cannot deduce which one. However, all the four queries are about some information about “club”. Thus, the attacker concludes that  $u_4$  must have queried about “club”, which is sensitive with respect to  $u_4$ 's preference. In a word, the attacker can infer that a user has issued a sensitive query with high confidence by query homogeneity attack.

### 3.3. $p$ -Sensitivity model

To protect against location linking attack, each request could be de-linked from its issuer by confusing the attacker

with more than one user appearing in the cloaking region of the request; each user could be de-linked from his/her query by confusing the attacker with more than one request with cloaking regions covering the user's location. We first define the concept of anonymity set for each user and each query as follows:

**Definition 1 (User Anonymity Set).** For a given anonymized request  $r' = (L', q')$  in  $R'$ , the *User Anonymity Set*  $S_u$  of  $r'$  contains all users  $u^*$  in  $R^*$  whose location  $l^*$  is covered by  $L'$ . Formally,  $r'.S_u = \{u^* \mid u^* = (id^*, l^*) \in R^*, l^* \in L'\}$ .

**Definition 2 (Request Anonymity Set).** For a given user  $u^*, u^* = (id^*, l^*)$  in  $R^*$ , the *Request Anonymity Set*  $S_r$  of  $u^*$  contains all requests  $r'$  in  $R'$  of which the cloaking region  $L'$  covers  $l^*$ . Formally,  $u^*.S_r = \{r' \mid r' = (L', q') \in R', l^* \in L'\}$ .

For a given anonymized request  $r'$ , we denote by  $P(r' \rightarrow u^*)$  the probability that a user  $u^*$  in  $r'.S_u$  being the true issuer of  $r'$ . For a given user  $u^*$ , we denote by  $P(u^* \rightarrow r')$  the probability of a request  $r'$  in  $u^*.S_r$  being sent by  $u^*$ . Based on the assumption of equal background knowledge, each request being sent by any user in its  $S_u$  happens with equal probability and each user has the same chance of sending any request in the user's  $S_r$ . Thus, in order to defend against location linking attack, it is required that  $P(r' \rightarrow u^*)$  is always less than or equal to a given threshold  $p_{uor}$  and  $P(u^* \rightarrow r')$  is always less than or equal to a given threshold  $p_{rou}$ :

$$P(r' \rightarrow u^*) = \frac{1}{|r'.S_u|} \leq p_{uor}; \quad (1)$$

$$P(u^* \rightarrow r') = \frac{1}{|u^*.S_r|} \leq p_{rou}. \quad (2)$$

To protect against query homogeneity attack, each user could be de-linked from sensitive queries by confusing the attacker with some insensitive requests in the user's  $S_r$ . For a given user  $u^*$ , we denote by  $P(u^* \rightarrow Q_s)$  the probability that  $u^*$  has sent a *sensitive query*. Hence, it is required that  $P(u^* \rightarrow Q_s)$  is always less than a given threshold  $p$ . It can be formalized as:

$$P(u^* \rightarrow Q_s) = \frac{|\{u^*.S_r \mid u^*.S_r \in Q_s\}|}{|u^*.S_r|} < p, \quad (3)$$

where  $|\{u^*.S_r \mid u^*.S_r \in Q_s\}|$  counts the number of sensitive queries in the query anonymity set of user  $u^*$ .

Expressions (1) and (2) ensure that any request will be linked with at least  $\frac{1}{p_{uor}}$  users and any user will be linked with at least  $\frac{1}{p_{rou}}$  requests. For simplicity, we use the same  $k$  to represent  $\frac{1}{p_{uor}}$  and  $\frac{1}{p_{rou}}$  in the rest of the paper. For any user that has sent a sensitive query, Expression (3) ensures that less than  $p$  of the queries are sensitive among all the requests that correspond to this user, i.e., the probability of any user sending a sensitive query is less than  $p$ . Finally, we wrap up the  $p$ -sensitivity model as:

**Definition 3** ( $p$ -Sensitivity).  $p$ -sensitivity is satisfied if and only if 1) for each user  $u^*$ ,  $P(u^* \rightarrow r') \leq \frac{1}{k}$ ,  $P(u^* \rightarrow Q_s) < p$  and, 2) for each request  $r'$ ,  $P(r' \rightarrow u^*) \leq \frac{1}{k}$ .

## 4. Anonymization algorithm

Our anonymization algorithm aims to satisfy  $p$ -sensitivity while minimizing the anonymization cost (i.e., the sum of the costs of cloaking regions).<sup>1</sup> Our algorithm is divided into three steps. In step one, we would like to find all possible sets of cloaking regions, where each cloaking region satisfies  $k$ -anonymity (recall that  $p$ -sensitivity implies  $k$ -anonymity). Here we use a widely-used property of  $k$ -sharing region to achieve  $k$ -anonymity. The main idea of  $k$ -sharing is that all users in the same anonymity set will share the same cloaking region, which could resist outlier attack [4]. Then the main problem of this step becomes to partition the whole user set into groups such that each group contains a minimum of  $k$  users. We adopt the common practice in existing  $k$ -anonymization literature, which requires no overlap between user groups. In other words, each user could exist in just one group. Thus, in step one, we find all possible partitions, each of which consists of a set of disjoint, non-overlapping cloaking regions that satisfy  $k$ -anonymity. In step two, we prune the partitions found in step one that do not satisfy  $p$ -sensitivity. Finally, we choose the optimal partition with the minimal cost among all the partitions retained in step two.

### 4.1. PE-Tree

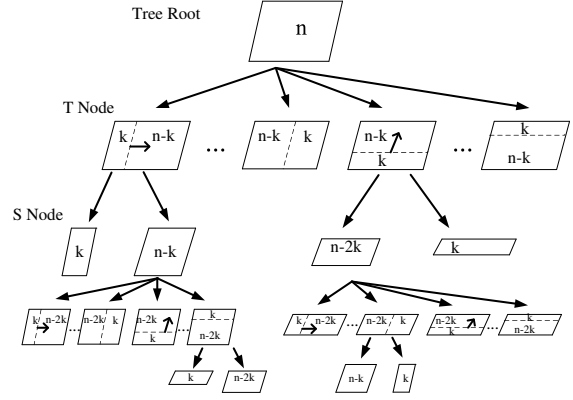
In the first step, it is easy to partition the user set into groups according to their locations. However, the number of possible partitions is exponential with respect to the number of users. Thus, we propose to construct a Partition-Enumeration-Tree (*PE-Tree*), in which all possible partitions of the user set are enumerated through tree expansion. We will later develop efficient searching and pruning heuristics over the PE-Tree to quickly identify the optimal partition.

Figure 3 shows the data structure of a PE-Tree for  $n$  requests from different users, where each user is represented by a (location) point. We give some formal definitions of a PE-Tree.

**Tree Definition 1** (PE-Tree). A PE-Tree has a single root. Each non-leaf node could have more than one child. Each node consists of one or more node entries. The tree enumerates all partitions of a given user set through tree expansion.

**Tree Definition 2** (Tree Node Entry). Each node entry contains a point set and the pointers to its children. The point set contains from  $k$  to  $n$  points in accordance with  $k$ -anonymity requirement.

<sup>1</sup>In the rest of this paper, cost is referred to as anonymization cost unless explicitly mentioned.



**Figure 3. PE-Tree Structure.**

**Tree Definition 3** (Tree Node). We define two types of tree nodes: Transformation node (denoted by ‘T’ node) and Split node (denoted by ‘S’ node). As shown in Figure 3, the root contains the whole point set and is an S node. A T node is generated by an S node through transformation, which partitions the points in the entry of the S node into two subsets/entries along  $x$  or  $y$  dimension. Different partition lines generate different transformations (the arrow shown in some T nodes means the trend of moving the partition line). Each S node can have many T children (transformations). Except the root node, an S node represents the point subset corresponding to one entry of its parent (a T node). Each T node has two S children.

**Tree Definition 4** (Leaf Node). A leaf node is an S node that cannot further split due to either of the following two reasons: 1) the node contains less than  $2k$  points (otherwise at least one subset does not satisfy  $k$ -anonymity); 2) the node contains more than  $2k$  points but all its T children are invalid.

**Tree Definition 5** (Partition). A partition consists of a set of leaf nodes that cover all the points.

An S node is valid if the point set satisfies our privacy requirement, that is, containing more than  $k$  points and the percentage of sensitive queries is less than a threshold of  $p$ . A valid partition consists of a set of valid leaf nodes that cover all the points.

The optimal partition is the valid partition with the minimum cost. For a T node, its cost is the sum of the costs of its two S children. For a non-leaf S node with many T children, its cost is the minimum cost of its children. For a leaf S node, its cost is the sum of the cost of each point contained in it. While all the points share the same cloaking region in a leaf node (i.e., the minimum bounding rectangle (*MBR*) of the point set), the cost of a leaf node is the product of the point counts and the area of their *MBR*. For a partition, the cost is the sum of the cost of each leaf node contained in it. In this paper, we assume the cost of an *MBR* does not increase with decreasing the number of users in the region. This property implies that after splitting one node into two,

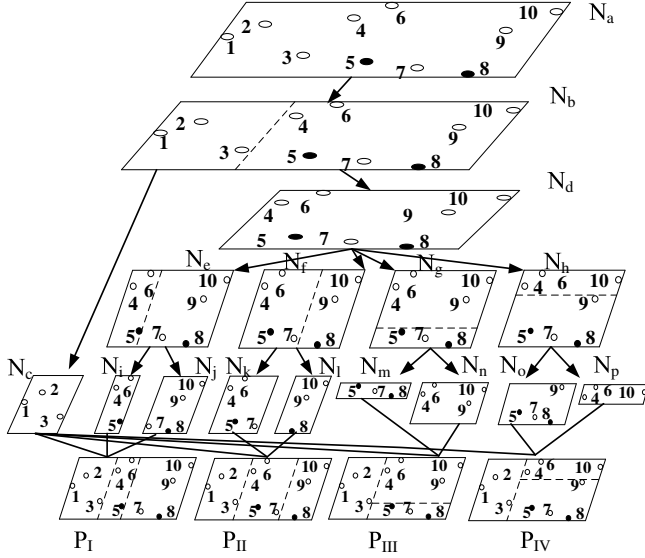


Figure 4. An Example of PE-Tree.

each will have a lower cost than the parent. Without loss of generality, we use the area as the cost metric for an *MBR*.

Figure 4 illustrates an example of PE-Tree for a set of 10 users, which are numbered by the order of their  $x$  coordinates (denoted by 1 to 10). In the interest of space, here we just show one subtree of the root and omit other subtrees. We assume  $k = 3$ . The root  $N_a$  contains the whole set of 10 users. Node  $N_b$  is one of the transformation child of  $N_a$ , generated by partitioning  $N_a$  at the position between users 3 and 4 along the  $x$  dimension. Nodes  $N_c$  and  $N_d$  are the split children of  $N_b$ . All the nodes  $N_e$  through  $N_g$  are transformation children of node  $N_b$ , generated by further partitioning  $N_b$  at different positions along the  $x$  and  $y$  dimensions respectively. All the nodes  $N_i$  through  $N_p$  are the split children of one node in  $N_e$  through  $N_g$ .  $N_c$  and  $N_i$  through  $N_p$  become leaf nodes because each of them contains less than  $2k$  users. For all the points from 1 to 10, the black ones represent requests with sensitive queries while the white ones represent requests with insensitive queries. Given  $p = 1/2$ , nodes  $N_m$  and  $N_o$  become invalid because the percentage of sensitive queries in them are  $2/3$  and  $1/2$  (no less than  $1/2$ ). As a result, there are four possible partitions, represented from  $P_I$  to  $P_{IV}$ , each consists of three leaf nodes.  $P_{III}$  and  $P_{IV}$  are invalid partitions because node  $N_m$  and  $N_o$  are invalid while  $P_I$  and  $P_{II}$  are both valid. This example will be used throughout the next subsections without explicit reference.

## 4.2. Pruning heuristics

Intuitively, we construct the PE-Tree first, then search for the optimal partition by fully traversing the tree using some standard traversal strategies such as depth-first search. For each partition, the cost is computed and compared with the optimal cost identified so far. If the cost is lower than the

optimal cost found so far, the current partition and its cost are recorded as optimal. This process is not stopped until we have examined all the partitions. Since this partition enumeration is systematic and complete, once the algorithm terminates, we are guaranteed to have identified an optimal partition.

To expedite the above process, our algorithm employs some pruning heuristics while constructing the PE-Tree based on two lemmas:

**Lemma 1.** If an S node is invalid, then its T parent is invalid.

**Lemma 2.** If any node is invalid, then the whole subtree of this node could be pruned.

Thus, when we generate an invalid S node, the other children of its T parent can be pruned from generation. For example, when  $N_m$  is computed as invalid,  $N_g$  is immediately marked as invalid and its second child  $N_n$  is not necessary to be generated. This way the PE-Tree size and the search space are significantly reduced.

To prune more nodes, we try to compute a lower cost bound for each node (denoted as  $N_{LCB}$ ), which is the lowest cost that all possible leaf node sets in the subtree of the node could achieve. This  $N_{LCB}$  participates in the comparison with the optimal cost searched so far. If  $N_{LCB}$  of a node is greater than the optimal cost got so far, then we can prune this node without generating its subtree. We compute  $N_{LCB}$  as follows. For each user point in the node, the area of the minimum rectangle containing itself and  $k-1$  other points is computed as its lower cost bound (denoted as  $P_{LCB}$ ). Any other possible cloaking region of this point will definitely have a larger area than  $P_{LCB}$ . Thus, the summation of  $P_{LCB}$  over all the points contained in the node can serve as  $N_{LCB}$  for this node. To compute  $N_{LCB}$ , there are two alternative methods. In the first method, we compute  $P_{LCB}$  for each user point of a given node when examining this node. For example, when examining node  $N_k$ , the  $P_{LCB}$  of user  $u_7$  is the area of the *MBR*{4, 5, 7} (cf. the  $k$ -anonymity set *MBR*{4, 5, 6, 7}). For the second method, we pre-compute all the rectangles that contain  $k$  user points in the root node. Each point maintains a list of its relevant rectangles. When examining a node, for each user point in this node, we choose the minimum  $P_{LCB}$  from all the relevant rectangles that are contained in the boundary of this node. For example, user  $u_5$  has a list of rectangles *MBR*{4, 5, 6}, *MBR*{5, 6, 7}, *MBR*{8, 7, 5}, *MBR*{7, 5, 3}, and *MBR*{5, 3, 1}. The  $P_{LCB}$  of user  $u_5$  is the area of *MBR*{5, 6, 7} when examining node  $N_k$ .

## 4.3. Algorithm optimizations

We now discuss several optimization techniques which further improve the efficiency of finding the optimal solution.

*Redundant Node Search:* We have an observation that in the PE-Tree, many nodes of the same point set are repeatedly generated in different branches. To improve the

redundancy, we use a hash table to index all the nodes with the key of the *MBR* boundary. This is to (automatically) identify all redundant nodes. When generating a node for the first time, we create the node in the hash table. When we have to generate it again, it is found in the hash table and can be used directly. For example, in Figure 4, node  $N_l$  with point set  $\{8, 9, 10\}$  will be generated many times in different subtrees of  $N_a$  by partitioning along the  $x$  dimension; it is created only once for the subtree of  $N_b$ .

*Optimal Cost Bound:* After searching a node for the first time, we store the cost of this node as its  $N_{OCB}$  in the hash table. When this (redundant) node is searched again later, we can use  $N_{OCB}$  as the filter instead of  $N_{LCB}$  because  $N_{OCB}$  is always larger than  $N_{LCB}$  and can prune more nodes. In fact,  $N_{LCB}$  of each node is just used for the first time and  $N_{OCB}$  is used in later steps. For example, the  $P_{LCB}$  of user  $u_5$  in node  $N_d$  is the area of  $MBR\{5, 7, 8\}$  when examining  $N_d$  for the first time. When examining  $N_d$  again, because node  $N_g$  and  $N_h$  are both invalid and node  $N_f$  has less cost than  $N_e$ , we could know that the actual cost of  $u_5$  is the area of  $MBR\{4, 5, 6, 7\}$  in  $N_f$ .

## 5. Performance evaluation

Our experiments evaluate the performance of our proposed anonymization algorithm and compare it with the state-of-the-art Mondrian algorithm [12]. For Mondrian, we use the same greedy multidimensional algorithm in [12], except that we find the optimal solution with minimum anonymization cost by its median-partitioning strategy rather than finding an approximate solution.

We use the well-known Brinkhoff data generator [1] to simulate movement of mobile users on a  $[1..100, 1..100]$  space. Each user randomly generates sensitive and insensitive queries, with a probability of 40% being sensitive queries. The number of users in each anonymization process is set at 200. The users' location coordinates in each dimension fall into 100 distinct values. By default, we set  $p$ , users' sensitivity requirement in location cloaking, at 0.4.

The algorithms under investigation are implemented in Java. All experiments run on a dual-processor Inter 1.83 GHz desktop with 1 GB main memory. We report the average result over 20 simulation runs.

The first set of experiments evaluate the effectiveness of the anonymization algorithms in terms of the anonymization cost. Figure 5(a) shows the result when  $k$  is varied from 5 to 25. Our algorithm (denoted by  $p$ -sensitivity) clearly outperforms Mondrian for all  $k$  values tested. When  $k$  is increased, the anonymization cost is higher as the cloaking region for each user becomes larger to satisfy the increased privacy requirement. We also evaluate the scalability of our algorithm in terms of the anonymization time. Figure 5(b) shows that Mondrian runs within 1 second for all cases as it does not find the optimal anonymization among all the partitions. Although our algorithm is slower than Mondrian, it is still within 40 seconds for the worst case (i.e., when  $k=5$ ), and improves quickly with increasing  $k$  (drops

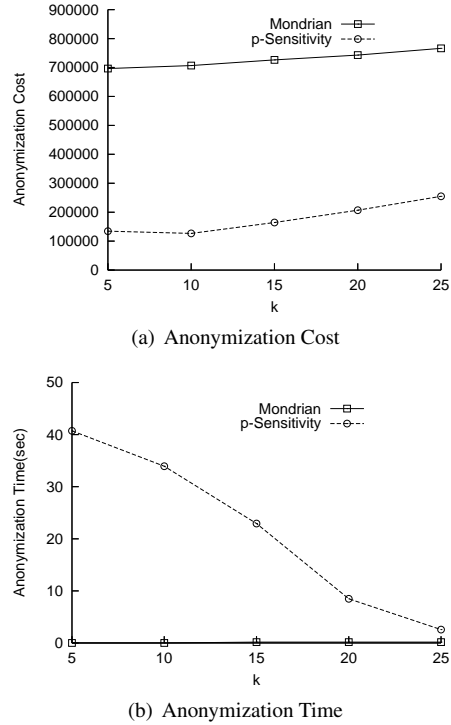
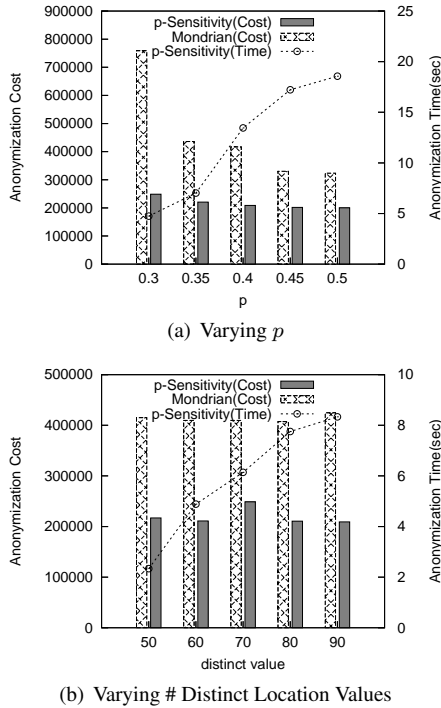


Figure 5. Performance as a Function of  $k$ .

to about 2 seconds when  $k=25$ ). Such an overhead, we believe, is reasonably acceptable compared to the data transfer latency with a (slow) mobile network.

The performance of our algorithm is also influenced by some other parameters such as  $p$  and distinct location values. For 200 users,  $k = 25$ , and 100 distinct location values in each dimension, Figure 6(a) shows that when increasing  $p$ , the anonymization cost decreases while the anonymization time increases. The reason is as follows. With a larger  $p$ , more valid nodes will be searched, which will significantly increase the anonymization time. Meanwhile, more search space will provide more chances to find a solution with smaller anonymization cost. Compared with Mondrian, our anonymization cost increases a little with decreasing of  $p$ , whereas Mondrian increases a lot. This is because we always find the optimal solution while Mondrian does not.

For 200 users,  $k = 25$ , and  $p = 0.5$ , Figure 6(b) shows the impact of distinct location values on the performance of our algorithm. As we aim to provide non-overlapping cloaking regions for all users, for each iteration, all the points that have the same value in the current partition dimension should be processed as a whole and included in the same cloaking region (strict multidimensional partitioning in Mondrian [12]). With more distinct values in each dimension, we could have more partitions for a given node and, hence, the anonymization time will greatly increase. However, no matter how many distinct values we have, the user locations are in the same range. Thus, the anonymization cost would not have much fluctuation with varying distinct location values, which is verified by both the result of



**Figure 6. Performance Impact of  $p$  and Distinct Location Values.**

Mondrian and our algorithm. We can also see that Mondrian gets a much higher anonymization cost than ours under a wide range of system parameters.

## 6. Conclusion

In this paper, we propose a new  $p$ -sensitivity model that considers query diversity and semantics for protecting query privacy in LBS applications. We present a PE-Tree based method to implement the  $p$ -sensitivity model. Search algorithms and heuristics are developed to efficiently find the optimal  $p$ -sensitivity anonymization in the tree. Preliminary experiments are conducted to demonstrate that  $p$ -sensitivity provides high-quality services without compromising users' query privacy. Moreover, the results show that our algorithm achieves a much lower anonymization cost than the existing algorithm.

## Acknowledgments

This research was partially supported by the grants from the Natural Science Foundation of China under grant number 60573091; Program for New Century Excellent Talents in University(NCET). Jianliang Xu's work was partially supported by the Research Grants Council of Hong Kong under Projects HKBU211206 and FRG/07-08/II-23.

## References

- [1] T. Brinkhoff. A Framework for Generating Network-Based Moving Objects. In *GeoInformatica*, 6(2):153-180, 2002.
- [2] C. Bettini, L. Pareschi and S. Jajodia. Anonymity and Diversity in LBS: A Preliminary Investigation. In *PerCom 2007*, Work in progress session.
- [3] A. R. Beresford and F. Stajano. Location Privacy in Pervasive Computing. In *IEEE Pervasive Computing*, 2(1):46-55, 2003.
- [4] C. Chow and M. F. Mokbel. Enabling Private Continuous Queries for Revealed User Locations. In *SSTD 2007*: 258-275.
- [5] R. Cheng, Y. Zhang, E. Bertino and S. Prabhakar. Preserving User Location Privacy in Mobile Data Management Infrastructures. In *PET*, 2006.
- [6] E. Hendricks, editor. *Wireless Location Technology: The Ultimate Challenge to Privacy*. In *Privacy Times*, 2001.
- [7] M. Gruteser and D. Grunwald. Anonymous Usage of Location Based Services through Spatial and Temporal Cloaking. In *ACM/USENIX MobiSys*, 2003.
- [8] G. Ghinita, P. Kalnis and S. Skiadopoulos. PRIVE: Anonymous Location-based Queries in Distributed Mobile Systems. In *WWW*, 2007.
- [9] B. Gedik and L. Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *ICDCS*, 2005.
- [10] P. Kalnis, G. Ghinita, K. Mouratidis and D. Papadias. Preventing Location-Based Identity Inference in Anonymous Spatial Queries. In *TKDE*, 2007.
- [11] A. Khoshgozaran and C. Shahabi. Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy, In *SSTD 2007*, Boston, MA, 239-257.
- [12] K. LeFevre, D. DeWitt and R. Ramakrishnan. Mondrian Multidimensional  $k$ -Anonymity. In *ICDE*, 2006.
- [13] N. Li, T. Li and S. Venkatasubramanian.  $t$ -Closeness: Privacy Beyond  $k$ -Anonymity and  $l$ -Diversity. In *ICDE*, 2007.
- [14] M. F. Mokbel, C. Chow and W. G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *VLDB*, 2006.
- [15] A. Machanavajjhala, J. Gehrke, and D. Kifer.  $l$ -Diversity: Privacy Beyond  $k$ -Anonymity. In *ICDE*, 2006.
- [16] L. Sweeney.  $k$ -anonymity: A Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5): 557-570, 2002.
- [17] T. M. Truta and B. Vinay. Privacy Protection:  $p$ -Sensitive  $k$ -Anonymity Property. In *ICDE Workshop PDM*, 2006.
- [18] Z. Xiao, X. Meng and J. Xu. Quality-Aware Privacy Protection for Location-based Services. In *DASFAA*, 2007.
- [19] X. Xiao and Y. Tao. Personalized Privacy Preservation. In *SIGMOD*, 2006.
- [20] J. Xu, J. Du, X. Tang and H. Hu. Privacy-Conscious Location-Based Queries in Mobile Environments. Under journal submission, 2007.