Location Privacy Preserving without Exact Locations in Mobile Services

Xiao PAN\(^1\), Xiaofeng MENG\(^2\)

\(^1\) Shijiazhuang Tiedao University, Hebei 050043, China
\(^2\) Renmin University of China, Beijing 100872

Abstract Privacy preservation has recently received considerable attention in location-based services. A large number of location cloaking algorithms have been proposed for protecting the location privacy of mobile users. However, most of existing cloaking approaches assume that mobile users are trusted. And exact locations are required to protect location privacy, which are just the information mobile users want to hide. In this paper, we propose a \(p\)-anti-conspiring privacy model to anonymize over semi-honest users. Furthermore, two \(k^*\)NNG-based cloaking algorithms, \(v^*\)NNCA and \(e^*\)NNCA, are proposed to protect the location privacy without exact locations. The efficiency and effectiveness of the proposed algorithms are validated by a series of carefully designed experiments. The experimental results show that the price paid for location privacy protection without exact locations is small.

Keywords Location privacy, Semi-honest users, Privacy preservation, Location-based services.

1 Introduction

With advances in wireless communication and mobile positioning technologies, location-based services (LBS) have been gaining increasingly popularity. Typical LBS applications include road navigation (e.g., "What is the shortest path from my current position to the nearest hospital"), point-of-interest query (e.g., "Where is the nearest ATM"), receiving alerts (e.g., notification of a sale on gas or warning of a traffic jam), and so on [38]. While users enjoy various valuable services, they have to expose their exact locations and the identities to the service provider (SP). However, the SP is untrusted, who can exploit these information to breach mobile users privacy (e.g., travel habits, health and religion). Dobson and Fisher discussed the possibilities for misuse of location information in [10]. Meanwhile, according to a new report from market research firm Nielsen, most users in the United States are concerned about privacy when it comes to sharing their location [34]. A lack in privacy-ensuring system might hinder the growth of LBS.

The simplest way for protecting privacy is to replace the identity with a pseudonym before sending the query to the SP. However, this is not enough. Location information included in the query can be used as a quasi-identifier (QI) [41] to re-identify the user. Suppose that the query was issued from Alice’s home; it can then be linked to Alice with some background knowledge (e.g., telephone directory). We consider the location privacy is under threat when an adversary can obtain unauthorized access to raw location data and sensitive information due to location disclosing [29].

In recent years, many research efforts have been put into investigating how to preserve the location privacy of mobile users. They can be divided into two directions: 1) how to define the privacy protection model and protect users’ locations efficiently [18, 31, 40, 43, 44]; 2) how to efficiently answer privacy-aware location-based queries (e.g., nearest neighbor and range queries) [13, 21, 31]. In this paper, we investigate a new privacy protection model and two location cloaking algorithms, in line with the first research direction.
Location k-anonymity is the most prominent location privacy protection model proposed so far [18]. A mobile user is considered as location k-anonymous if and only if the location information sent to the SP is indistinguishable from those of at least $k - 1$ other users. To achieve location k-anonymity, cloaking [18, 22, 31] is a popular methodology. Its main idea is to reduce the spatial and temporal resolution of the user’s location, such that exact user locations are extended to cloaked regions and each region covers at least $k$ users. For example, Fig. 1 shows a location 3-anonymity example. Locations of A, B, and C are all extended to their minimum bounding rectangle (MBR) $CR$. Thus the adversary cannot be sure where is the exact location of each mobile user.

Though location k-anonymity model provides a good protection of user’s location privacy, cloaking reduces the resolution of the location information. Thus, the quality of service (QoS) is degraded. In order to maximize QoS, each mobile user exposes its exact location for cloaking in most cloaking algorithms. And the MBR or the minimum covering circle that encloses the genuine positions of $k$ users is usually used as the cloaked region. Here a very strong assumption is hidden that mobile users are trusted. However, in practice, there is no trusted party in the service space. Everyone could be malicious, and sell information to others or a third party.

For example, several malicious users meet at an agreed place and issue several LBS queries. Since their locations are spatially proximate, they are very likely to be anonymized together in a same cloaking set. If the malicious users corrupt with each other, they could infer the other user’s location. For the same example in Fig. 1, assume that A and B know each other, and share each other’s location. As the cloaking algorithm is public, A and B know that the cloaked region is the MBR of covering users. Then, A and B get a corrupted region $CR_{AB}$, which is the MBR covering A and B (blank region on the right side of the dash line). Therefore, user C location is inferred in the shadow region. If the shadow area is less than the privacy requirement of C, the location privacy is disclosed. In the worst case, exact location is exposed when the user locates at a corner of the cloaked region.

Meanwhile, there is an awkward situation in the most existing cloaking algorithms. For location privacy protection, each user exposes its exact location to an anonymizer [31] or a peer [19] for location cloaking. However, as no party is credible in the service space, not everyone wants to expose its location for others cloaking. The exact locations using in most existing cloaking algorithms are just the protection targets that users want to hide. In this sense, most existing cloaking algorithms have limited applications [15]. Therefore, we aim to propose a new cloaking algorithm over semi-honest users without exact locations.

There are two challenges for location privacy protection over semi-honest users without exact locations. First, on one hand, the cloaking algorithm should partition users into different cloaking sets as their spatial proximity. On the other hand, no exact location can be employed. Second, cloaked regions should be generated through users cooperating. But users don’t trust each other, and don’t want to provide exact locations. Meanwhile, we should ensure that the user’s location cannot be inferred by the other users through intermediate results.

[15] first answers the above two questions. It employs a proximity graph to find cloaking sets, and a progressive bounding algorithm to generate cloaked regions. However, the drawbacks of [15] are as follows. First, mobile users in [15] suffer from the threat of location privacy disclosure. In the progressive bounding algorithm, the host user asks other users in the same cloaking set whether agree with the hypothetic bound iteratively. Between two consecutive iterations, if a user P disagrees the hypothetic bound in the first iteration but agrees in the next iteration, the difference between two hypothetic bounds is just the location where P must locate. In the worst case, the difference between two hypothetic bounds is little, such that the exact location of the user is disclosed. Second, the $k'$-mutual nearest neighbor consistency [11] is ignored in [15]. [11] has verified that $k'$-mutual
nearest neighbor consistency is a key quality measure of data clustering, which requires that users in a cluster are $k^*$ mutual nearest neighbors. For example, Fig. 2 is an example of a proximity graph. The nodes represent mobile users and the weight on the edge is the relative distance between the two users. \{u_1, u_2, u_3\} and \{u_4, u_5, u_6\} are two cloaking sets generated by [15]. Though $u_2$ and $u_3$ are $u_1$’s 2 nearest neighbors (2NN), $u_1$ is neither one of 2NN for $u_2$ nor $u_3$ (see detail illustration in Section 4.1). The mutual nearest neighbor consistency is violated. Thus, the average cloaked region size is large, which leads to a poor QoS.

In order to remedy the above privacy disclosure problem, we propose a $p$-anti-conspiring privacy model. It requires that the probability for malicious users inferring each mobile user’s location cannot exceed the user privacy requirement $p$. In order to improve the QoS, two cloaking algorithms, $v^k$NNCA and $e^k$NNCA, are proposed, which follow $k^*$-mutual nearest neighbor consistency. For the above example, cloaking sets \{u_2,u_3,u_4\} and \{u_1,u_5,u_6\} are generated by our proposed algorithms, which satisfy $k^*$-mutual nearest neighbor consistency.

Specifically, mobile users carry mobile devices, which are able to measure the closeness from its peers, by measuring either the received signal strength (RSS) from its peers, or the time difference of arrival (TDOA) of beacon signals from its peers using omnidirectional antennas [15]. Thus, each mobile user could save a $k^*$ nearest neighbors ($k^*$NN) list locally, and sends the $k^*$NN information to a certificated server, e.g. base, periodically. Then the certificated server maintains a $k^*$ nearest neighbor graph ($k^*$NNG). In order to ensure the cloaking set with the property of $k^*$-mutual nearest neighbor consistency, the cloaking problem is changed to find independent cliques in the $k^*$NNG, such that the cloaking set is found from cliques. Afterwards, each user in a cloaking set contributes some space to the cloaked region according to the $p$-anti-conspiring privacy model. Based on this basic idea, users in a cloaking set generate a cloaked region collaboratively without privacy disclosure.

We conduct a series of experiments to evaluate the performance of the proposed algorithms using both location data generated from a well-known road network simulator [37] and adapted from two real datasets, the Athens trucks [4] and the Geolife dataset [17]. Experimental results show that $v^k$NNCA and $e^k$NNCA are efficient in terms of the cloaking time and the cloaking success rate. The average cloaking time is only 0.16ms and the cloaking success rate is about 94% for most cases. Meanwhile, the anonymization cost and the scarified cost are acceptable. The width (height) of the cloaked region is about 200m with real dataset.

The contributions of this paper can be summarized as follows:

- We propose two cloaking algorithms $v^k$NNCA and $e^k$NNCA based on $k^*$NNG, which maintain the $k^*$-mutual nearest neighbor consistency such that the average anonymization cost is reduced;
- We propose a $p$-anti-conspiring privacy model. Under this model, semi-honest users in a cloaking set could generate the cloaked region collaboratively without privacy disclosure;
- We employ a decentralized hybrid architecture with limited cooperation, which exploits the hybrid nature of current networks and the capabilities of mobile devices;
- A series of experiments is conducted to evaluate the performance of our proposed algorithms. The experimental results validate the efficiency and effectiveness of our proposed algorithms.

The rest of the paper is organized as follows. We review the related work in Section 2. The system architecture and the problem under investigation are formally defined in Section 3. The new cloaking algorithms for finding cloaking set and generating cloaked regions are proposed in Section 4 and Section 5 respectively. Section 6 presents the performance evaluation results of our proposed algorithms. Finally, the paper is concluded in Section 7.

## 2 Related work

In this section, we first survey the system architectures for location privacy preserving in LBS in Section 2.1. Then, we give an overview of the existing techniques in online location privacy protection with and without exact locations in Section 2.2 and Section 2.3, respectively. Finally, we review the existing methods for off-line trajectory privacy preserving in Section 2.4.

### 2.1 System architectures

In terms of the system architecture for protecting location privacy, existing approaches can be classified into three categories: non-cooperative architecture, centralized architecture and decentralized architecture. In the non-cooperative architecture [9, 46], mobile users communicate with the SP directly. Mobile users maintain the location privacy based on their own knowledge. The SP answers the query and sends
the results to users directly. This architecture is easy to deploy. But as each mobile user only has its own knowledge and cannot employ other users location information, the privacy is easy to be disclosed.

The most widely used architecture is the centralized architecture [22–24, 31, 43]. The main difference between this architecture and the non-cooperative architecture is that a trusted third party, termed location anonymizer, is employed. The location anonymizer is placed between mobile users and the SP, which is responsible for location anonymization and query result refinement. As the location anonymizer has complete knowledge of the locations and queries of all users, it can provide much safer privacy protection. On the other hand, the centralized location anonymizer would become a single point of failure and a potential scalability bottleneck.

In order to overcome the drawbacks of the centralized architecture, the decentralized architecture is proposed [19, 20]. In this architecture, peer users collaborate with each other to keep their customized privacy information and maintain a distributed data structure to store the location information. A selected user in a cloaking set will act as the query sender and is responsible for the query result refinement returned by the SP. This architecture could be further subdivided into distributed architecture [19] and mobile peer-to-peer architecture [20]. For the former one, mobile users communicate with each other through a fixed communication infrastructure, e.g., base stations; for the latter one, mobile users communicate with their peers through multi-hop routing.

Among three system architectures mentioned above, users in the non-cooperative architecture employ their own locations to protect the privacy, thus users needn’t trust each other. However, in the centralized and decentralized architecture, most of work assume that mobile users are trusted. Thus, users’ exact locations are either exposed to the location anonymizer in centralized architecture or to trusted peers in decentralized architecture. Then, the cloaking process is performed on the exact locations. In this paper, we employ a decentralized hybrid architecture, combing the distributed architecture with mobile ad-hoc networks.

2.2 Online location privacy protection with exact locations
In the early work for location privacy protection, mobile users are supposed to trust each other. In order to maximize QoS, genuine locations are sent to an anonymizer or a peer for anonymization. [18] is the first cloaking algorithm, which follows location $k$-anonymity model. A Quad-tree based cloaking algorithm is used to generate cloaked regions in [18]. However, the privacy level $k$ must be set the same for all users in the system, which does not accommodate personalized privacy requirements. Furthermore, a very large cloaked region might be generated by the Quad-tree algorithm, which would result in a poor QoS. The cloaking algorithm employed in Casper [31] bears some similarity to [18] that a variant Quad-tree is used to compute the cloaked region. Yet different from [18], each user can specify the smallest area of a cloaked region and the smallest privacy level $k$.

The first work proposing personalized privacy requirements is [22]. [22] transforms the location cloaking problem to find cliques on an undirected graph. Our work also finds cloaking sets in cliques, but differs from [22] on the following aspects. First, the graph in [22] is constructed based on the exact locations of users. However, the graph in our paper is just a weighted proximity graph. Users proximity information is employed instead of exact locations. Second, [22] finds cliques for each user greedily, which results in exponential cost. However, we employ the well known Welsh-Powell algorithm to find the independent sets of the complementary graph, such that the graph is partitioned into different partitions. Then the cloaking set is found from each partition. Followed by [22], Xiao et al. [44] proposed to find cloaking sets from a directed graph. Similar with [22], exact user locations are employed to construct the graph.

[19] proposed a hilbASR cloaking algorithm, which is the first work making use of Hilbert curve to find the proximity relation between users in services. Then it was followed by many other works [20, 25]. However, due to drawbacks of the Hilbert curve, a large cloaking region is generated when the requests locate around the corner of the curve, which results in a bad QoS.

2.3 Online location privacy protection without exact locations

[12] is the first work that proposed to un-trust any involved party in the service space, neither any of the involved peers nor the SP. The proposed approach combines $k$-anonymity with obfuscation. Each peer obfuscates her position by substituting the precise location with a local cloaked region, and anonymizes her request by manipulating the local cloaked region to a global cloaked region. Most importantly, [12] proposed an anonymous algorithm, which selects the query requester in the global cloaked region with a near-uniform randomness. This method is only applied for mobile peer-to-peer architecture.

[15] first proposed to formalize the proximity informa-
tion of mobile users as a proximity graph. Then, it decomposes the cloaking problem into two subproblems: proximity minimum $k$-clustering and secure bounding. Our work also employs a proximity graph to facilitate finding cloaking sets, but differs from [15] in several aspects. First, as mentioned in the Introduction section, mobile users in [15] suffer from the threat of location privacy disclosure. However, we resolve this problem by proposing a $p$-anti-conspiring privacy model. Second, [15] doesn’t consider the mutual nearest neighbor consistency for a cloaking set, thus the anonymization cost is high. However, our work finds cloaking sets in the cliques of a $k^*$NNG, which refines the anonymization cost of cloaking sets.

Recently, some work suggested using encryption for location privacy protection. The basic encryption mechanisms can be divided into space transformation [26] and PIR protocols [21, 27]. Their main ideas are both that the query is encrypted so that the service provider answers the queries without knowing what kind of information is being retrieved. Then the user de-encrypts the result candidates and refines them. Both of them provide strong privacy guarantees. Therefore, mobile users in these kinds of work needn’t trust each other. However, encryption incur significant computational overhead, which is not an ideal solution for large number of moving objects with frequent location updates.

2.4 Off-line trajectory privacy protection in data publication

Spatial and temporal information included in trajectories have great benefits for many real-life applications, such as road network optimization, intelligent transportation, user behavior analysis, business analysis, and so on. However, publishing trajectory data to a third-party directly may cause serious individual privacy disclosure. Therefore, more and more work [1,5,28,32,42,45] pays more attention on trajectory privacy protection. The purpose of trajectory anonymization is to prevent user trajectories from being discovered when user locations are published.

In terms of the techniques used for protecting trajectory privacy, existing approaches can be classified into three categories [14]: dummy, generation and suppression. The main idea of dummy [28] is that a true trajectory is sent with several dummies, such that the true trajectory is indistinguishable from the others. Trajectory generation [1,32,45] is to reduce the spatial and temporal resolutions of location samples on a trajectory. The location samples are extended to cloaked regions so that the trajectory privacy is protected. To protect trajectory privacy, some location samples (e.g. sensitive locations) are not been published, that is the main idea of suppression based trajectory privacy protection method [5,42].

The biggest difference between trajectory privacy protection with our work is that the former one is an off-line process, but ours is online. Besides, exact locations are required by existing trajectory privacy protection methods. However, the exact location is just the target we want to protect. Therefore, existing methods for anonymizing trajectories are not applicable to our problem.

3 Preliminary

In this section, we formally define the problem under study. Section 3.1 describes the system architecture. Relevant definitions, including the privacy model, the cloaking set, and the anonymization cost etc., are given in Section 3.2.

3.1 System architecture

We employ a decentralized hybrid architecture, which combines the distributed architecture with mobile ad-hoc networks. It includes mobile users, a certification server (CServer) and LBS service providers, as shown in Fig. 3. Mobile users can communicate via wireless and cellular protocols to access LBS services, as the distributed architecture [19]. Meanwhile, mobile users establish ad-hoc (WiFi) point-to-point connections with other mobile peers in the network. As a result, there are several wireless mobile ad-hoc networks, represented by the dashed rectangles.

Mobile users in LBS carry devices with positioning technology (e.g., GPS) and a complete set of wireless communication interfaces (e.g., Bluetooth, WiFi, and HSDPA), typ-
ically Smartphones and PDAs. [15] observes that a mobile device is able to measure the closeness from its peers, by measuring either the received signal strength (RSS) from its peers, or the time difference of arrival (TDOA) of beacon signals from its peers using omnidirectional antennas. Fig. 4 shows a WiFi RSS of neighboring peers from a laptop computer [15]. According to these WiFi RSS neighboring peers, each mobile user could save a $k^*$ nearest neighbors locally.

From the Fig. 4, the mobile user could save 3NN list. Before issuing any LBS query, a mobile user first registers on the CServer with their identities (e.g., IP address, phone number) and receives a query identity (QID) from the CServer. The CServer (e.g., base) is trusted. When a mobile user issues an LBS query, the user sends the query content (e.g., "finding the nearest gas station") and its $k^*$ nearest neighbors, namely proximity information, to the CServer. The CServer maintains a proximity graph which is based on the proximity information. A vertex in the proximity graph stands for a mobile user, and an edge means that the two users are WiFi neighbors. The weight on the edge indicates the relative distance between two users, as Fig. 2 shows. Then, the CServer finds cloaking sets for mobile users from this graph. Note that, although the CServer participates in the anonymization process, which is different from the one in [19], the CServer DOES NOT know the exact location of each user. What the CServer knows is just the proximity information.

An LBS SP is untrusted. The SP answers location-based queries with cloaked regions rather than exact locations [31]. Instead of returning an exact answer, the SP returns a candidate answer list. Mobile users would locally refine the candidate list by their own locations.

The location cloaking for mobile users is conducted in two phases. In the first phase, user cloaking sets are identified through the proximity information in the proximity graph by the CServer. In the second phase, users in a same cloaking set cooperate with each other to compute the cloaked region, without exposing any exact location. This region would be served as the cloaked region for all users in the same cloaking set.

The complete workflow of a service request from a mobile user is as follows.

1. Mobile users register on the CServer and send $k^*$NN to the CServer periodically. An LBS query without location information is sent to the CServer through an authenticated and encrypted connection.
2. The CServer maintains a proximity graph. Upon receiving the query, the first phase of the anonymization algorithm is invoked to find cloaking sets in accordance with users privacy requirements. Then, the CServer sends a QID list to users in a same cloaking set correspondingly.
3. Mobile users of a group corporate with each other through peer-to-peer communication. A cloaked region is generated through the second phase of cloaking algorithm. A random user is selected from this group and sends the cloaked region to the CServer.
4. The CServer forwards the modified query consisting of the cloaked region and the query content to the SP.
5. Finally, the CServer will relay the result set returned from the SP to mobile users. And the mobile user refines the query result with its own exact location.

Privacy analysis: We discuss the location privacy guarantees in this system architecture. There are three parties in the service space: mobile users, CServer, and SP. Neither of them is trusted. First, each user sends the proximity information to the CServer instead of exact locations, thus the location is protected from the other mobile users and CServer. Second, the cloaking technique protects location privacy by reducing the spatial and temporal resolution. User’s exact location is hidden in a cloaked region. Thus, the location privacy is also protected from SP. In a word, no party in the service space knows the exact location of the user, except the user itself.

3.2 Definitions

As our privacy model is proposed over semi-honest mobile users, we give the definition of semi-honest users first.

Definition 1. (semi-honest users [2]) A mobile user is semi-honest if it follows the protocol properly with the exception that it keeps a record of all its intermediate computations, such that other mobile users’ privacy information is inferred.
Let’s take the users A, B, and C in Fig. 1 as the example. A and B both follow the anonymization algorithm and provide genuine locations for cloaking. However, both of them employ the intermediate computations (e.g., CR_{AB}, CR, etc.) to breach user C privacy. Thus, these three users are all semi-honest users.

We observe that for a semi-honest user u in service space, u has two kinds of background knowledge: the internal knowledge, denoted by InK_u, and the external knowledge, denoted by ExK_u. InK_u is u’s own knowledge, for example, u’s exact location. ExK_u is the knowledge given by other parties, including the intermediate computations received from the CServer, the knowledge given by other users who corrupt with u, and so on. With the internal and external background knowledge, we define p-anti-conspiring privacy model as follows.

**Definition 2. (p-anti-conspiring privacy model)** Assume p is the maximum probability that user u can tolerate to expose her privacy information. For any user u_j in the service space, if

\[ P_u(u_j(InK_u + ExK_u)) \leq p, \]

user u is said to be p-anti-conspiring, where \( P_u(u_j(InK_u + ExK_u)) \) is the probability that user u_j infers u’s privacy information with u_j’s whole background knowledge (namely, InK_u + ExK_u).

In this paper, a user’s privacy refers in particular to location privacy without special specification. Thus, we only focus on p-anti-conspiring location privacy model. Specifically, under p-anti-conspiring location privacy model, the probability for malicious users inferring each mobile user’s location cannot exceed the user privacy requirement p, even malicious users with their whole background knowledge.

We still use cloaking set {A, B, C} in Fig. 1 as the example to illustrate the above definitions. Recall that A and B know each other, thus they share each other’s location. A knows its own location (InK_A), B’s exact location (ExK_A), and the published cloaked region CR (ExK_A). Then A gets a corrupted region CR_{AB}. Therefore, user C location is inferred in the shadow region CR − CR_{AB}. Then

\[ P_A(C(InK_A + ExK_A)) = \frac{1}{\text{Area}(CR - CR_{AB})}. \]

If Area(CR − CR_{AB}) is too small, such that \( P_A(C(InK_A + ExK_A)) > p_C \), C doesn’t follow p-anti-conspiring privacy model, thus the location privacy of C is disclosed.

For each mobile user in service space, its privacy profiles are specified by two parameters: anonymity level (k) and constrained privacy level (p). Thus, a users set is a cloaking set if the conditions in Definition 3 are satisfied.

**Definition 3. (cloaking set)** Let CS be a users set with a region CR. CS is a cloaking set and CR is the cloaked region if and only if for any user u ∈ CS,

- the anonymity level \( k \leq |CS| \), where \(|CS|\) is the size of CS;
- user u follows p-anti-conspiring privacy model;
- user u’s location is covered by CR and the probability of u locating in CR is uniform;
- Area(CR) ≤ maxArea, where Area(CR) is the area of CR, and maxArea is the tolerable worst QoS specified by system.

The first two conditions ensure CS is a cloaking set and the last two conditions ensure CR is the cloaked region. Specifically, the first condition protects the user identity by following the location k-anonymity model; the second condition prevents the user location from been disclosed when other users in a same cloaking set corrupt with each other; the third condition hides the user exact location in a region; and the fourth condition ensures the quality of services.

Similarly to the existing work [6, 16, 23], we employ the average area of the cloaked region for each query as a measure for anonymization cost.

**Definition 4. (anonymization cost)** Consider a series of cloaking sets CU. For any cloaking set \( CS \in CU \), its anonymization cost is the product of the MBR area of CS and the number of users in CS:

\[ \text{cost}(CS) = \text{Area}(MBR(CS)) \times |CS|. \]

The total anonymization cost of CU is:

\[ \text{Total}_\text{Cost}(CU) = \sum_{CS \in CU} \text{cost}(CS). \]

The average anonymization cost of CU is:

\[ \text{Average}_\text{Cost}(CU) = \frac{\text{Total}_\text{Cost}(CU)}{\sum_{U \in CU} |U|}. \]

Let’s consider the six mobile users in Fig. 5, represented by dotted points, as an example. Suppose that \( \{u_1, u_2, u_3\} \) are two cloaking set. And the rectangles with dash lines in Fig. 5) are the cloaked regions. As Definition 4, the average anonymization cost is 2.5(=2.5\times3\times2\times6\times2\times2\times2\times2\times2\times2).
The essence of most location privacy protecting work [19, 22, 23, 43] is to partition mobile users into different groups as the above example shows, thus users in a same group constitute a cloaking set. The goal of the partition method is to minimize the total anonymization cost. Obviously, it is a typical combinational optimization problem. Therefore, the location anonymization problem in mobile services could be formalized as follows.

**Definition 5. (Mobile users partition)** Let V be the mobile users set waiting to be anonymized. V is partitioned into different groups (e.g. CS₁, CS₂, ..., CSₙ), such that
- CS₁ ∪ CS₂ ∪ ... ∪ CSₙ = V;
- for any i and j, CSᵢ ∩ CSⱼ = ∅;
- for any i, the size of CSᵢ is not less than the anonymity level k, that is |CSᵢ| ≥ k;
- the total anonymization cost ∑ᵢ=1^n cost(CSᵢ) is minimized.

The second condition ensures that cloaking sets are independent with each other. In other words, the cloaking region has k-sharing property [8].

### 4 kNNG-based cloaking set finding algorithms

Recall that our algorithm is conducted in two phases: finding cloaking sets and generating cloaked regions. We first give the algorithms for identifying cloaking set through kNNG in this section, and will elaborate the process of generating cloaked regions in Sections 5.

#### 4.1 Preliminary

As elaborated in Section 3, each mobile user sends kNN to the CServer. Thus, the CServer could maintain a weighted proximity graph (WPG) [15] to save mobile users proximity information. Continuing using Fig. 2 as an example, u₁ to u₆ are six mobile users. From this graph, we can see that u₁ and u₂, u₁ and u₅ are neighbors in service space. Moreover, u₂ is much nearer to u₁ than u₅. The graph in Fig. 2 is connected. However, in most cases, the WPG is a non-connected graph, composed of multiple connected components.

[15] proves that location k-anonymity on WPG is equivalent to k-clustering. Therefore, Definition 5 could be amended as follows.

**Definition 6. (Weighted graph partition)** Given a weighted proximity graph G = (V, E), where V is the set of vertices (users), E is the set of edges, and the weight on the edge represents the relative distance between two users. The weighted graph partitioning problem consists in dividing G into n disjoint partitions, G = {P₁(V₁, E₁), P₂(V₂, E₂), ..., Pₙ(Vₙ, Eₙ)}, where
- V₁ ∪ V₂ ∪ ... ∪ Vₙ = V, E₁ ∪ E₂ ∪ ... ∪ Eₙ ⊆ E, and partition Pᵢ (1 ≤ i ≤ n) is connected;
- For any i and j (1 ≤ i ≤ n, 1 ≤ j ≤ n), Vᵢ ∩ Vⱼ = ∅ and Eᵢ ∩ Eⱼ = ∅;
- For any i (1 ≤ i ≤ n), each partition size is not less than the anonymity level k, namely |Vᵢ| ≥ k;
- Cost(G) is minimized. The cost of G could be the sum of subgraph diameters for partitions, or the sum of edge weights for partitions, and so on.

Each partition, in the above definition, is a cluster in the graph. The simplest way of the weighted graph partition is the kNN algorithm. Specifically, for each mobile user, it clusters with its k-1 nearest neighbors that have not been clustered yet in the WPG. Continuing with the example in Fig. 2, u₁ is scanned first. 2NN(u₁) = {u₂, u₃}, thus {u₁, u₂, u₃} is anonymized together and removed from the WPG. Then u₄ is scanned, 2NN(u₄) = {u₅, u₆} now. Then {u₄, u₅, u₆} is anonymized together. The final partitions are shown in Fig. 6(a). The sum of edges weight for partitions in Fig. 6(a) is 15. From Fig. 2, we observe that there exists a better partitions set: {u₁, u₅, u₆} and {u₂, u₃, u₄}, whose edges weight sum is 11, as shown in Fig. 6(b). The reason is that the mutual nearest neighbors consistency is ignored in Fig. 6(a). Specifically, 2NN(u₁) = {u₂, u₅}. However, from the original WPG, 2NN(u₂) = {u₁, u₄} and 2NN(u₅) = {u₂, u₄}. Though u₂ and u₅ are u₁’s 2NN, u₁ is neither one of 2NN for u₂ nor for u₅. However, u₂, u₃ and u₄ are mutual two nearest neighbors. When u₁ is cloaked with u₂ and u₃, they are removed from the WPG.

---

2) Without exact locations, the anonymization cost is measured by one kind of graph cost approximately. In this paper, we employ the sum of edge weights as the graph partition cost.
As a result, \( u_4 \) cannot find its mutual 2NN \( u_2 \) and \( u_3 \), and has to cloack with much farther users \( u_5 \) and \( u_6 \). [11] proposed and verified \( k^* \)-mutual nearest neighbor consistency is a key quality measure of data clustering. Inspired by [11], we employ the \( k^* \)-mutual nearest neighbor consistency to improve the quality of partitions.

**Definition 7.** \((k^*\text{-mutual nearest neighbor consistency})\)

Given a users set \( P \), \( P \subseteq V \), where \( V \) is the mobile users set. For \( \forall v_i, v_j \in P \), if \( v_i \in k^*\text{NN}(v_j) \) and \( v_j \in k^*\text{NN}(v_i) \), then we say \( P \) satisfies \( k^* \)-mutual nearest neighbor consistency.

In the above example, \( \{u_2, u_3, u_4\} \) satisfies 2-mutual nearest neighbor consistency.

We employ \( k^*\text{NNG} \) to get clusters with the property of \( k^* \)-mutual nearest neighbor consistency.

**Definition 8.** \((k^*\text{ nearest neighbor graph}, k^*\text{ NNG})\) A \( k^*\text{NNG} \) is a weighted graph \( G(V, E) \), where

- For any vertex \( v \in V \), \( v \) is a mobile user who registers on the CServer;
- For any edge \((v, u) \in E \), \( u \in k^*\text{NN}(v) \) and \( v \in k^*\text{NN}(u) \);
- The weight \( w \) of \((v, u) \) is the relative distance between \( v \) and \( u \).

The second condition indicates that there is an edge between two vertices if and only if the two users are mutual \( k^* \)NN. From Fig. 2, we can get the 2NN list for each user, as shown in Fig. 7(a). And the corresponding 2NN is shown in Fig. 7(b). As the CServer receives \( k^*\text{NNG} \) from each mobile user, \( k^*\text{NNG} \) could be maintained conveniently.

Comparing WPG and \( k^*\text{NNG} \), we can see that \( k^*\text{NNG} \) is also a WPG. Only the condition, whether there exists an edge between two vertices, is more constraint in \( k^*\text{NNG} \) than in WPG. Therefore, location \( k^* \)-anonymity on \( k^*\text{NNG} \) is also a weighted graph partition problem. As we known, graph partition problems falls under the category of NP-hard problems [35]. Therefore, we aim to propose heuristics algorithms.

4.2 \( vk^*\text{NNCA} \) and \( ek^*\text{NNCA} \)

4.2.1 \( k^*\text{NNG} \) partitions

As Definition 6, a cloaking set is a connected subgraph in the WPG. Now the problem changes to find independent connected subgraphs in the \( k^*\text{NNG} \). Obviously, users in a clique of the \( k^*\text{NNG} \) follow the \( k^* \)-mutual nearest neighbor consistency, and the clique is a connected subgraph. Moreover, if the clique size is not less than the anonymity level \( k \), users in the clique could constitute a candidate cloaking set \(^3\).

Therefore, we change the problem to find independent cliques in an un-directed graph. Unfortunately, finding cliques in an un-directed graph incurs exponential computational cost as we known. However, fortunately, there are many research fruits in finding independent sets. As we known, graph coloring is a way of coloring the vertices of a graph such that no two adjacent vertices share the same color [36]. A \( l \)-coloring is the same as a partition of the vertex set into \( l \) independent sets. Therefore, we first compute the complement graph \( \bar{G} \) for graph \( G \). Then, we employ the well known graph coloring algorithm, Welsh-Powell algorithm [39], to find the maximal independent sets on \( \bar{G} \), which are maximal cliques on \( G \).

Algorithm 1 shows the procedure of Welsh-Powell algorithm. In Line 1, the complement graph \( \bar{G} \) is computed. Then sort the vertices in \( \bar{G} \) with non-ascending order of the degree, and store them in a stack \( vset \) in Line 2. From Line 3 to Line 6, for each vertex in \( vset \), the vertex is colored with \( c_i \), which is the minimum integer not being used by its colored neighbors. Repeat above process until all the vertices have been colored. From Line 7 to Line 9, put users with the same color number into the group \( uset_{ci} \). Finally, delete the edges in \( uset_{ci} \) from graph \( G \) (Line 10). Thus, \( uset_{ci} \) is a maximal clique in \( G \).

Fig. 8 is a \( k^*\text{NNG} \) where \( k^* = \text{4} \). Taking this graph as a

\(^3\) For stating conveniently, we assume \( k^*\text{NNG} \) is a connected graph in this section. For a disconnected \( k^*\text{NNG} \), the proposed algorithms are applied on each connected component.
Algorithm 1 Welsh-Powell algorithm

Input: vertices set $V$, edges set $E$
Output: cliques set $uset$

1: compute the complement graph $\bar{G}$;
2: $vset$=sort the vertices $V$ in $\bar{G}$ with non-ascending order of the degree;
3: for each node $v_n$ in $vset$ do
4: if $v_n$ is uncolored then
5: record all the neighbors colors of $v$;
6: give a color $c_1$ to $v$ which is the minimum integer not being used;
7: for each used color $c_i$ do
8: if vertex $v_n$ in $vset$ is colored as $c_i$ then
9: put $v_n$ into $uset_{c_i}$;
10: delete the edges in $uset_{c_i}$ from graph $G$;

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{4NNG.png}
\caption{4NNG}
\end{figure}

running example, we get maximal cliques set \{$\{A, B, I\}$, $\{C, H\}$, $\{D, E, F, G\}$} after running through Algorithm 1.

Not every clique got from Algorithm 1 is a cloaking set. Therefore, we classify the maximal cliques into positive candidates and negative candidates.

Definition 9. (positive candidates and negative candidates)
Let $\text{CliqueS}$ be the set of cliques. For any clique $C \in \text{CliqueS}$,
- if $|C| \geq k$, $C$ is a positive candidate;
- otherwise, $C$ is a negative candidate.

Continuing with above example, for simplicity, assume all users specify $k=4$, cliques $\{A, B, I\}$ and $\{C, H\}$ are both negative candidates. Clique $\{D, E, F, G\}$ is a positive candidate.

4.2.2 Two combing strategies
If the anonymity level $k$ is large, most cliques generated from Algorithm 1 tend to be negative candidates. Recall that a cloaking set is just required to be a connected subgraph instead of a clique. The condition, that users in a cloaking set are required to be a clique in $k^\prime$NNG, is too strict. In order to guarantee more users anonymized successfully, we propose two combining strategies to relax the constraint condition, such that negative candidates are transformed to positive ones. One strategy is to combine negative candidates from scanning edges, and the other is from scanning vertices.

Algorithm 2 Edge-based combing algorithm

Input: negative candidates set $NegSet$, positive candidates set $PosSet$, $k^\prime$NNG $G$, anonymity level $k$
Output: negative candidates set $NegSet$, positive candidates set $PosSet$

1: edgeset=sort the edges of $G$ by edge weight with non-decreasing order;
2: while edgeset is NOT NULL do
3: $(v, u)$=pop up the top edge from edgeset;
4: $c_v$=the candidate set which $v$ is in;
5: $c_u$= the candidate set which $u$ is in;
6: if $c_v \neq c_u$ and ($c_v$ is a negative candidate or $c_u$ is a negative candidate) then
7: if $c_v$ is a negative candidate then
8: negc $\leftarrow c_v$;
9: $fc$ $\leftarrow c_u$;
10: else
11: if $c_u$ is a negative candidate then
12: negc $\leftarrow c_u$;
13: $fc$ $\leftarrow c_v$;
14: merge negc into $fc$;
15: delete negc from NegSet;
16: if $|fc| \geq k$ and $fc$ was a negative candidate before merging then
17: delete $fc$ from NegSet;
18: insert $fc$ into PosSet;

The basic idea of combining negative candidates from scanning edges is as follows. Pick out the edge $(v, u)$ with the minimum edge weight from $k^\prime$NNG. If $v$ or $u$ is in a negative candidate, merge the two candidates, which $v$ is in and $u$ is in. Then pick out the edge with the second minimum edge weight. Repeat above steps until all edges are scanned. The detailed algorithm is shown in Algorithm 2.

First, the edges in $k^\prime$NNG $G$ are saved in a stack edgeset and sorted by edge weight with non-decreasing order (Line 1). Then, pop up the top edge $(v, u)$ from edgeset (Line 3). The candidate sets $c_v$ and $c_u$, which $v$ is in and $u$ is in respectively, are found (Line 4 and Line 5). If $v$ and $u$ are not in a same candidate, check the candidate $c_v(c_u)$ whether is a negative candidate. Save the negative candidate into negc, and the other candidate into $fc$ (Line 7 to Line 13). Merge the users in the negative candidate negc into $fc$ (Line 14). If the size of $fc$ is not less than $k$ and $fc$ was a negative candidate before merging, insert $fc$ into the positive candidates set $PosSet$ and delete old $fc$ from NegSet (Line 16 to Line 18). Repeat Line 3 to Line 18 until edgeset is NULL. The time complexity of Algorithm 2 is $O(n|E|)$, where $n$ is the number of vertices in
$G$, and $|E|$ is the edges number in $G$.

Continuing with the example in Fig. 8, $\text{edgeset} = \{(B, C), (C, D), (H, I), (H, G)\}$ \footnote{The edges in cliques (e.g. (A, B), (A, I), (B, I), et al.) are removed after the clique is found from Algorithm 1 (see Line 10 in Algorithm 1).}. (B, C) is popped up first. B is in the negative candidate $\{A, B, I\}$, and C is in the candidate $\{C, H\}$. Then $\{A, B, I\}$ is removed from $\text{NegSet}$ and combined with $\{C, H\}$. As $\{C, H\}$ is a negative candidate, $\{C, H\}$ is removed from $\text{NegSet}$ first, and then $\{A, B, C, H, I\}$ is inserted into positive candidates set $\text{PosSet}$. The remaining edges in $\text{edgeset} = \{(C, D), (H, I), (H, G)\}$ are checked one by one without any combing as two kinds of reasons. The one is that the two vertices lying on the edge are in a same candidate (e.g. (H, I)). And the other is that neither of the candidates, which the two vertices are in, is negative (e.g. (C, D)).

Obviously, the edge-based combing strategy may cause many useless checking, e.g. the last three times checking in the above example. In order to avoid this case, we propose another combing strategy that searching from a vertex in a negative candidate. The basic idea is as follows. For each users set $c$ in the negative candidates set $\text{NegSet}$, start from any point $v_p$ in $c$. Candidate users set $c'$ is merged with $c$, if $c'$ is adjacent with $c$ through $v_p$, and the weight sum of $c$ and $c'$ is minimum. Repeat above steps until no negative candidate could be merged from $\text{NegSet}$. The detailed algorithm is shown in Algorithm 3.

### Algorithm 3 Vertex-based combing algorithm

**Input:** negative candidates set $\text{NegSet}$, positive candidates set $\text{PosSet}$, $k^\ast\text{NNG}$ $G$, anonymity level $k$

**Output:** negative candidates set $\text{NegSet}$, positive candidates set $\text{PosSet}$

1. while $|\text{NegSet}|$ is decreasing do
2. $c$=pop up the top item from $\text{NegSet}$;
3. for each node $v_p$ in $c$ do
4. list=vertices which are adjacent to $v_p$, but are not in $c$;
5. for each node $nl$ in list do
6. $cl$=the candidate where $nl$ is;
7. if $c \neq cl$ and $\omega_\sum(c) + \omega_\sum(cl) < min$ then
8. $min = \omega_\sum(c) + \omega_\sum(cl)$;
9. $min = cl$;
10. if $\text{minc} \neq \text{NULL}$ then
11. $\text{minc}=\text{merge} c$ with $\text{minc}$;
12. if $|\text{minc}| \geq k$ then
13. if $\text{minc}$ was in $\text{NegSet}$ then
14. delete $\text{minc}$ from $\text{NegSet}$;
15. insert $\text{minc}$ into $\text{PosSet}$;

While $\text{NegSet}$ is not NULL, pop up the top item $c$ from $\text{NegSet}$ (Line 1 and Line 2). For each vertex $v_p$ in $c$, find the neighbors list of $v_p$, which are not in $c$, in $k^\ast\text{NNG}$ (Line 4). Then we can get the neighbor candidates which the nodes of list are in. From the neighbor candidates, the candidate $\text{minc}$ which has the minimum edges weight sum with $c$ is selected (Line 5 to Line 9). Then, merge the vertices in $c$ to $\text{minc}$ (Line 11). If now the size of $\text{minc}$ is not less than anonymity level $k$ and $\text{minc}$ was a negative candidate before combining, delete $\text{minc}$ from the negative candidates set (Line 14). And $\text{minc}$ is inserted into positive candidates set $\text{PosSet}$ (Line 15). Repeat above steps (Line 2 to Line 15) until the size of $\text{NegSet}$ doesn’t change.

As for the time complexity of Algorithm 3, in each while loop iteration, the most expensive operation is to find the merging candidate with minimum sum of edge weights (Line 3 to Line 9). In each iteration, Line 3 to Line 9 takes $O(kk^\ast n)$, where $k$ represents the upper bound of the size for a negative candidate $c$ and $k^\ast$ is the maximum degree of the $k^\ast\text{NNG}$ $G$.

Since $|\text{NegSet}|$ decreases to 0 in the worst case, the worst-case time complexity of Algorithm 3 is $O(kk^\ast n|\text{NegSet}|)$.

Still taking Fig. 8 as the running example, $\text{NegSet} = \{(H, C), (A, B, I)\}$. (H, C) is popped up first. {A, B, I} and {D, E, F, G} are both neighbor candidates w.r.t. (H, C). The edges weight sum of $\{(H, C), (A, B, I)\}$ is minimum. Thus, (H, C) is merged into {A, B, I} and removed from $\text{NegSet}$. {A, B, I} changes to {A, B, C, H, I}, which becomes a positive candidate. Finally, {A, B, I} is also removed from $\text{NegSet}$. {A, B, C, H, I} is inserted into $\text{PosSet}$.

### 4.2.3 $ek^\ast\text{NNCA}$ and $vk^\ast\text{NNCA}$

Let’s summarize what we have discussed above. For a $k^\ast\text{NNG}$, we first partition it into independent cliques employing Algorithm 1. Then, the cliques are categorized into positive candidates and negative candidates. Next, delete the edges in the cliques from $k^\ast\text{NNG}$. After that, in order to improve the success rate of negative candidates, two combining strategies are proposed. One is to combine negative candidates based on scanning edges in $k^\ast\text{NNG}$, and the other is based on scanning vertices in the negative candidate. Finally, users in the positive candidate constitute the candidate cloaking set. And they will be employed in Algorithm 6 (in Section 5) to generate cloaked regions. We call the cloaking algorithm with edges combining strategy as $ek^\ast\text{NNCA}$, and the one with vertices combing strategy as $vk^\ast\text{NNCA}$. $ek^\ast\text{NNCA}$ is shown in Algorithm 4. $vk^\ast\text{NNCA}$ is same with $ek^\ast\text{NNCA}$, except the combing strategy (calling Algorithm 3) in Line 8 (detailed algorithm is omitted).
For the time complexities for both algorithms, they are both dominated by the Algorithm 1 and the algorithm for refining the candidates in NegSet (Line 8). As we known, the time complexity of Welsh-Powell Algorithm is $O(n + |E|)$. Thus, the time complexity of $ek\times$NNCA is $O(n + |E| + n|E|)$, and the one of $vk\times$NNCA is $O(Cn + |E|)$, where $C = kk\times|NegSet| + 1$.

**Algorithm 4 ek\times$NNCA**

**Input**: $k\times$NNG $G$, anonymity level $k$

**Output**: cloaking sets with cloaked regions

1: call Algorithm 1 to partition $G$ into different components $CP$;
2: for each $c$ in $CP$ do
3: delete the edges in $c$ from $G$;
4: if $|c| \geq k$ then
5: insert $c$ into positive candidates set PosSet;
6: else
7: insert $c$ into negative candidates set NegSet;
8: call Algorithm 2 to refine candidates in NegSet;
9: push users in PosSet to Algorithm 6 to find cloaked regions;

4.2.4 Further refinement

Comparing the WPG in Fig. 2 and the $k\times$NNG in Fig. 7(b), we observe that some proximity information is missing in $k\times$NNG, e.g. $(u_1, u_2)$, $(u_1, u_6)$, and so on. That is because the edge definition in $k\times$NNG is more constrained. Such information missing may result in users, who are supposed to cloak successfully, failure to find cloaking sets. For example, in Fig. 2, $u_1$ is supposed to be anonymized successfully if $k=2$. However, $u_1$ becomes an isolated user in Fig. 7(b).

In order to resolve the problem of isolated users, we find the nearest neighbors for isolated users by checking the missing information in WPG. Then, similar with [30], the isolated users are inserted into the candidates which the nearest neighbors are in. The detail algorithm is shown in Algorithm 5. For each isolated user $v$ in $k\times$NNG, we find its nearest neighbor $v_n$ in WPG. If $v_n$ is in a candidate $c_{v_n}$ whose size is not less than $k-1$, insert $v$ into $c_{v_n}$. Otherwise, find the next nearest neighbor for $v$, and repeat above steps. The time complexity of Algorithm 5 is $O(|isolv|n)$, where $\Delta$ is the maximum vertex degree of the WPG $proximityG$ and $|isolv|$ is the number of isolated users.

From Fig. 7, the $k\times$NNG is partitioned into $\{\{u_2, u_3, u_4\}, \{u_5, u_6\}\}$ and an isolated user $u_1$. From Fig. 2, NN($u_1$)=$u_2$. $u_1$ is inserted into $\{u_2, u_3, u_4\}$ if $k=2$.

**Algorithm 5 Isolated users refinement**

**Input**: WPG $proximityG$, negative candidates set $NegSet$, positive candidates set $PosSet$, anonymity level $k$, isolated users set $isolv$

**Output**: positive candidates set $PosSet$

1: for each user $v$ in $isolv$ do
2: $nnlist$=find the neighbors of $v$ in $proximityG$;
3: sort vertices in $nnlist$ in ascending order by edge weight;
4: for each vertex $v_n$ in $nnlist$ do
5: $c_{v_n}$ = the candidate where $v_n$ is;
6: if $|c_{v_n}| \geq k-1$ then
7: insert $v$ into $c_{v_n}$;
8: break;

5 Cloaked regions generating algorithm

In previous section, mobile users are partitioned into positive candidates set, each of which satisfies location $k$-anonymity. Recall that besides location $k$-anonymity, users in a cloaking set should also follow $p$-anti-conspiring privacy model (see Definition 3). In this section, we propose the strategy for mobile users generating cloaked regions without exposing exact locations under $p$-anti-conspiring privacy model.

Obviously, if the location privacy is protected on $x$-dimension and $y$-dimension respectively, the user’s location is without the danger of disclosing. Therefore, in our cloaking region generating algorithm, we ensure the user’s location privacy on each dimension respectively. For easily understanding, we specify the $p$-anti-conspiring privacy model in this section as follows. Let $CS$ be the cloaking set. For any user $u$ in $CS$, assume that the malicious user could infer $u$ location being at $[loc1l, loc1r]$ on one dimension with its whole background knowledge, then $[loc1r-loc1l]$ could not be less than $P (= p \times \min(width, height)$, where $p$ is user $u$’s conspired privacy level and $width (height)$ is the width (height) of system space).

The basic idea of generating the cloaked region for a users set is as follows. The user who issues an LBS query is sacrificed first. The query initiator issues a candidate cloaked region according to its $p$-anti-conspiring privacy model. Then this candidate cloaked region is passed to other users in the same cloaking set one by one. For each receiving user, it picks up one direction from [right, left] and [top, down] of the candidate region respectively. Then a random number $\delta$ is computed as user’s conspired privacy requirement. Finally, extend the width and height of the candidate region by $\delta$ on
Let \( R \) be a candidate cloaked region, and \( V \) be the chosen directions. Before elaboration how to select the extending direction and compute \( \delta \), we classify the users into four categories according to the relative location of a user w.r.t. a rectangle: x-type user, y-type user, xy-type user and inside user.

**Definition 10.** Let \( R \) be a candidate cloaked region, and \( V \) be a user waiting for extending \( R \). MBR is the minimum bounding rectangle of \( R \) and \( V \).

- If the location of \( V \) only extends the MBR boundary on \( x(y) \)-dimension, \( V \) is an x-type (y-type) user, such as the user \( V \) in Fig. 9(a)(Fig. 9(b)).
- If \( V \) extends the boundaries of MBR on both \( x \) and \( y \) dimensions, \( V \) is an xy-type user, such as the user \( V \) in Fig. 9(c).
- If \( V \) is inside \( R \), \( V \) is an inside user, such as the user \( V \) in Fig. 9(d).

**Rule 1. (query initiator)** A query initiator \( u_1 \) gets two random numbers \( R_1 \) and \( R_2 \) from \([P, \text{max}T]\), where \( \text{max}T \) is the tolerable worst QoS on one dimension that is specified by system. A rectangle \( R \) is generated with width \( R_1 \) and height \( R_2 \), which covers \( u_1 \). The probability of user \( u_1 \) locating in \( R \) is evenly distributed.

The query initiator sends the candidate cloaked region \( R \) to a nearby user in the same cloaking set. According to the different relative locations of the receiving user w.r.t the candidate cloaked region, the extending rule is different. For stating conveniently, let \( \text{wid} \) (\( \text{hgt} \)) be the width (height) of the candidate cloaked region \( R \).

**Rule 2. (x-type user)** If the receiving user \( u \) is an x-type user w.r.t. the candidate cloaked region \( R \), a random number \( r_n \) is taken from \([b_n, \text{max}T - \text{wid}]\) according to a probability distribution (e.g. normal distribution), where

\[
b_n = \begin{cases} C_x, & \text{if wid} + C_x \geq u.P, \\ u.P - (\text{wid} + C_x), & \text{if wid} + C_x < u.P. \end{cases}
\]

where \( C_x \) is the contribution of \( u \) to the MBR boundary on x-dimension. Let the extending width be \( \delta \),

\[
\delta = \begin{cases} r_n, & \text{if wid} + C_x \geq u.P, \\ r_n + C_x, & \text{if wid} + C_x < u.P. \end{cases}
\]

Then, pick a direction \( \text{Dir} \) from \{top, down\} randomly. Finally, the candidate cloaked region \( R \) is extended by \( \delta \) on the Dir and \( C_x \) directions.

We use the x-type user in Fig. 9(a) as an example. User \( V \) is on the right of \( R \). Thus, in order to cover \( V \), the right side of the rectangle \( R \) should be at least extended by \( C_x = V_x - R_x \), where \( V_x \) is the x-coordinate of \( V \) and \( R_x \) is the x-coordinate of the right boundary of \( R \). Assume that \( \text{wid} + C_x \geq V.P \), it means that the new width satisfies the \( p \)-anti-conspiring privacy requirement of \( V \). Thus, as the equation (1) in Rule 2, the minimum expanded value \( b_n = C_x \). A random number \( r_n \) is got from \([b_n, \text{max}T - \text{wid}]\) according to a probability distribution. Then as the equation (2) in Rule 2, \( \delta = r_n \). Assume that \( \text{down} \) is picked from \{top, down\}. Thus, in this example, \( R \) is extended by \( \delta \) on the right and down directions.

In symmetry, we get the extending rule for y-type users.

**Rule 3. (y-type user)** If the receiving user \( u \) is a y-type user w.r.t. the candidate cloaked region \( R \), a random number \( r_n \) is taken from \([b_n, \text{max}T - \text{hgt}]\) according to a probability distribution, where

\[
b_n = \begin{cases} C_y, & \text{if hgt} + C_y \geq u.P, \\ u.P - (\text{hgt} + C_y), & \text{if hgt} + C_y < u.P. \end{cases}
\]

where \( C_y \) is the contribution of \( u \) to the MBR boundary of \( u \) and \( R \) on y-dimension. Let the extending height be \( \delta \),

\[
\delta = \begin{cases} r_n, & \text{if hgt} + C_y \geq u.P, \\ r_n + C_y, & \text{if hgt} + C_y < u.P. \end{cases}
\]

Then, pick a direction \( \text{Dir} \) from \{right, left\} randomly. Finally, \( R \) is extended by \( \delta \) on the \( \text{Dir} \) and \( C_y \) directions.

After applying Rule 2 or Rule 3, on the random selected direction, the height (width) of the candidate cloaked region may not follow \( p \)-anti-conspiring privacy requirement, for example \( R \) is a slim rectangle. Thus, for Rule 2 and Rule 3, we give a supplementary rule.
**Supplementary Rule:** After applying Rule 2 or Rule 3, if the height (width) of the candidate cloaked region doesn’t satisfy the user’s p-anti-conspiring privacy requirement, the height and width both continue to be extended by Δn, where Δn = u.P – min(hgt, wid).

Combing Rule 2 with Rule 3, we get the extending rule for xy-type users.

**Rule 4. (xy-type users)** If the receiving user u is an xy-type user w.r.t. the candidate cloaked region R, compute the extended width δx using Rule 2 first. Then compute the extended height δy using Rule 3. δ = max(δx, δy). Finally, R is extended by δ on both x and y dimensions.

**Rule 5. (inside user)** If the receiving user u is an inside user w.r.t. the candidate cloaked region R, the extending rule is as follows.

- If min(wid, hgt) < u.P, a random number δ is taken from \{u.P – min(wid, hgt), maxT – max(wid, hgt)\} according to a probability distribution. Then pick a direction dir1 (dir2) from {right, left} ({top, down}) randomly. Finally, R is extended by δ on both dir1 and dir2 directions.
- If min(wid, hgt) ≥ u.P, R doesn’t change.

---

**Algorithm 6 Cloaked region computing algorithm**

**Input:** users set CS in positive candidates set

**Output:** cloaked region R

1. query initiator ut generates a region R as Rule 1;
2. while there exists un-visited user in CS do
3. sort list by the relative distance with non-decreasing order;
4. ut’ = pop up the top item from list;
5. while ut’ exists do
6. pass R to ut’;
7. find the type ty of ut’ w.r.t. R;
8. compute δ as the rule of type ty;
9. R’ = extending R as the rule of type ty;
10. if R’ doesn’t exist then
11. put ut’ into failure set;
12. delete ut from CS;
13. ut’ = pop up the top item from list;
14. else
15. R = R’;
16. ut = ut’;
17. break;
18. if |CS| < k then
19. insert users in CS into failure set;
20. else
21. return CS with R’

---

From Rule 2 to Rule 5, the extended length δ is computed and the extended directions are selected as different types of users. However, from the attackers’ view, for all cases, the candidate cloaked region is extended by δ on both x-dimension and y-dimension. In other words, though the receiving user’s types are different, the extending actions are same. Therefore, the user’s privacy is protected.

The detail procedure for generating the cloaked region is shown in Algorithm 6. Query initiator ut generates an initial rectangle R as Rule 1 (Line 1). When there exist un-visited users in CS, find the neighbors list of ut in CS (Line 3). User ut’ with the minimum edge weight is picked out (Line 4 and Line 5). According to the relative position of ut’ w.r.t. R, the correspondent rule is employed to generate the extending length δ and the extending direction (Line 8 to Line 9). Then, R is extended to R’ by δ on selected directions (Line 10). If R’ doesn’t exist, it implies user ut’ is an outlier. Thus ut’ is removed from CS (Line 12 and Line 13). Then pop the top item from list. Return to Line 6 and repeat the above process. If the extended region R’ exists, set ut’ as the new sending user (Line 16 and Line 17), and repeat above steps until all users in CS have been visited. Finally, if the size of CS is not less than k, CS is returned as the cloaking set with the cloaked region R’ (Line 19 to Line 22).

Continuing with the running example in Fig. 8, {A, B, C, H, I} is a positive candidate set, as shown in Fig. 10(a). Assume that user C is the query initiator. As Rule 1, R1 is generated. B and H are both adjacent to C. As H is nearer to C than B, C sends R1 to H. H is an xy-type user w.r.t. R1, thus δ1 is computed as Rule 4. And, R1 is enlarged to R2 by δ1 on directions where H locates, as shown in Fig. 10 (b). Afterwards, H sends R2 to unvisited user I. I is an inside user w.r.t. R2, and assume that the width and height of R2 both satisfy I’s p-anti-conspiring privacy requirement. Therefore,
I directly passes $R2$ to A, as A is nearer to I than B. A is an 
{x-type user w.r.t. $R2$. Therefore, Rule 2 is employed to get a 
random number $\delta_2$ and extended directions (namely left and 
top). $R2$ is extended to $R3$ by $\delta_2$ on the left and top directions, 
as shown in Fig. 10(c). Then A sends $R3$ to B. B finds himself 
inside $R3$. Now all users have been visited, therefore, $R3$ is 
the finally cloaked region.

6 Experiments

In this section, the effectiveness and efficiency of our pro-
posed algorithms are experimentally evaluated under various 
system settings. We first describe the experiment setup in 
Section 6.1, followed by the performance evaluation results 
presented in Sections 6.2-6.7.

6.1 Experiment setup

[15] is the first and representative work for location privacy 
protection employing the proximity graph, thus we compare 
four algorithms, namely $e^k$-NNCA, $v^k$-NNCA, $k$-Clustering 
and PNNCA. $e^k$-NNCA and $v^k$-NNCA are cloaking algo-
rithms proposed in this paper. $k$-Clustering is revised from 
the centralized cloaking algorithm proposed in [15] with new 
cloaked region generating method proposed in this paper. 
PNNCA is shorted for pairwise nearest neighbor cloaking 
algorithm. In this method, edges of the $k^\ast$NNG are sorted in 
the ascending order by the edge weight first. Then, remove 
edges from the $k^\ast$NNG iteratively. In each iteration, two 
nodes (mobile users) lying on the removed edge are merged 
to a cluster. Finally, users are partitioned into different clus-
ters. If the cluster size is not less than $k$, this cluster is a can-
didate cloaking set. PNNCA ensures users nearest to each 
other are cloaked together, as the edge with the minimum 
edge weight is removed in each iteration. Thus, PNNCA is 
used as the minimum bound for anonymization cost.

As we known, cloaking reduces the resolution of the lo-
cation information. Thus, the quality of service is degraded. 
It is often desirable to strike a balance between the location 
privacy and the quality of service. Therefore, to the best of 
our knowledge, existing work on location privacy protection 
did experiments to show the balance. The widely used eval-
uation metrics include the cloaking success rate, the average 
anonymization cost, and the cloaking time. The three param-
eters show the quality of services. Besides, we also show the 
sacrificed cost for cloaking without exact locations. In sum-
mmary, the evaluation metrics we used include the cloaking 
success rate, the average anonymization cost (Definition 4), 
the sacrificed cost for cloaking without exact locations, and 
the cloaking time for successful requests.

In most of our experiments, we use the well known 
Thomas Brinkhoff Network-based Generator of Moving Ob-
jects [37] to generate the moving objects in the system. The 
input of the generator is the road map of Oldenburg County. 
In the default setting, there are a total of 50,000 moving ob-
jects. By default, every mobile object sends at most 10 near-
est neighbors to the CServer, and the system parameter $\text{max}$ $T$ 
is set to 5% of max(height, width) of the system space. For 
each user, its anonymity level ($k$) is set to 9 and conspired 
privacy level ($p$) is set to a number randomly in $[0.05, 0.1]% 
of the max(height, width) of the system space. When com-
puting the cloaked region, a normal distribution (mean $b_0$ and 
standard deviation $p$) is used to generate the random value $r_n$.

Table 1 lists the default system settings.

<table>
<thead>
<tr>
<th>parameter</th>
<th>default setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of users</td>
<td>50,000</td>
</tr>
<tr>
<td>max number of nearest neighbors ($k^\ast$)</td>
<td>10</td>
</tr>
<tr>
<td>anonymity level ($k$)</td>
<td>9</td>
</tr>
<tr>
<td>conspired privacy level ($p$)</td>
<td>a random number from $[0.05%, 0.1%]$ of max(height, width) of the system space</td>
</tr>
<tr>
<td>$\text{max}$ $T$</td>
<td>5% of max(height, width) of the system space</td>
</tr>
<tr>
<td>normal distribution (mean, standard variance)</td>
<td>($b_0$, $p$)</td>
</tr>
</tbody>
</table>

In addition to the simulated data, we also adapt two dif-
ferent real data sets, the Athens trucks [4] and the Geolife [17], 
to validate the effectiveness of our cloaking algorithms. All 
cloaking algorithms are implemented in C++ and run on a 
desktop PC with a dual AMD 780MHz processor and 2GB main 
memory.

6.2 Impact of anonymity level

In this section, we investigate the impact of anonymity level 
$k$ on the performance of cloaking algorithms. Increasing 
anonymity level $k$ indicates each cloaking set should cover 
more users. From Fig. 11(a), we observe that the suc-
cess rates of four cloaking algorithms all decrease with 
more constrained anonymity level. $e^k$-NNCA and $v^k$-NNCA 
show the similar trend (two lines almost overlap together), 
as the two algorithms are only different on the scanning 
strategy for negative candidates refinement (Line 7 in Al-
gorithm 4). Both of them have the highest success rates 
among the four algorithms. Though the success rates
of \( k \)-Clustering are just slightly lower than \( ek'\text{-NNCA} \) and \( vk'\text{-NNCA} \) on all anonymity levels, \( k \)-Clustering incurs much higher anonymization cost (see Fig. 11(b)). \( \text{PNNCA} \) performs the worst. That is mainly because \( \text{PNCA} \) finds cloaking sets through combing nodes in the \( k'\text{-NNG} \). The cloaking set is removed from the system when the anonymity level is satisfied. This might affect the cloaking set result of some remaining mobile users. However, \( vk'\text{-NNCA} \) and \( ek'\text{-NNCA} \) both have negative candidate set refinement strategy, and clusters in \( k \)-Clustering wouldn’t be partitioned further when the cluster size is less than the anonymity level.

![Fig. 11](image)

**Fig. 11** Impact of anonymity level (a) avg. success rate (b) avg. anonymization cost (c) avg. sacrificial cost (d) avg. time for finding cloaking set

We evaluate the cost of the algorithms from two aspects: the average anonymization cost and the sacrificial cost for cloaking without exact locations. From Fig. 11(b), for each cloaking algorithm, the average anonymization cost increases with anonymity level \( k \) increasing, as each cloaking set should cover more mobile users. \( \text{PNCA} \) merges two nearest users in each iteration, thus \( \text{PNNCA} \) has the smallest anonymization cost. In contrast, \( k \)-Clustering performs the worst. The anonymization cost of \( k \)-Clustering is over three (seven) times than the anonymization cost of \( ek'\text{-NNCA} \) and \( vk'\text{-NNCA} \) (\( \text{PNNCA} \)) when \( k=3 \). \( k \)-Clustering partitions the WPG through removing edges with the maximum edge weight. This strategy can only guarantee that two farthest users are not in a cloaking set, but cannot ensure users in a cloaking set are nearest to each other. Thus, the average anonymization cost of \( k \)-Clustering is much higher. \( ek'\text{-NNCA} \) and \( vk'\text{-NNCA} \) find cloaking set from cliques in \( k'\text{-NNG} \), which satisfy \( k' \)-mutual nearest neighbor consistency. Therefore, their anonymization costs are between \( \text{PNCA} \) and \( k \)-Clustering. Besides, \( vk'\text{-NNCA} \) shows better anonymization cost than \( ek'\text{-NNCA} \), which results from the different merging strategy. \( ek'\text{-NNCA} \) finds the merged candidate with the minimum sum of edge weights. However, \( ek'\text{-NNCA} \) can only guarantee the weight of the edge associating with two merged candidates is minimum.

In our cloaked region generating algorithm, mobile users cooperate with each other to generate the cloaked region without the exact locations. This strategy must incur more anonymization cost than the one cloaking with exact locations. Thus, we evaluate the sacrificial cost by comparing the average anonymization cost with the MBR area of mobile users with exact locations, as shown in Fig. 11(c). We observe that \( k \)-Clustering sacrifices the least cost, whereas \( \text{PNNCA} \) sacrifices the most. This is totally opposite to the performance shown in Fig. 11(b). Less sacrificial cost of \( k \)-Clustering just illustrates that users in a cloaking set are far to each other, thus the area of MBR is large, which approximates to the average anonymization cost of obscure locations. However, \( \text{PNNCA} \) merges nearest users into a cloaking set, thus the area of MBR is so small that more cost is sacrificed. Moreover, from Fig. 11(c), the sacrificial costs for \( ek'\text{-NNCA} \), \( vk'\text{-NNCA} \), and \( \text{PNNCA} \) decrease with the anonymity level increasing. That is mainly because further users are included into the cloaking set. Comparing \( ek'\text{-NNCA} \) and \( vk'\text{-NNCA} \), the former sacrifices less than the latter.

The cloaking time of a user request consists of two parts: the time for finding cloaking sets in the \( \text{CServer} \) and the time for generating cloaked regions between mobile users. The time for generating cloaked regions is related to the communication cost in the network. We assume the communication cost in the network per user is fixed. Then in each cloaking set, the bound verification message in Algorithm 6 must transfer at least \(|CS| \) times. Obviously, the communication cost increase when the size of cloaking set increases. In the following parts, we only evaluate the time for finding cloaking set in the \( \text{CServer} \), which dominates the whole cloaking time. As shown in Fig. 11(d), the times for finding cloaking sets of all algorithms increase due to a more constrained anonymity requirement. Obviously, \( ek'\text{-NNCA} \) and \( vk'\text{-NNCA} \) require a much shorter cloaking time than \( k \)-Clustering and \( \text{PNNCA} \). Comparing between \( ek'\text{-NNCA} \) and \( \text{PNNCA} \), \( ek'\text{-NNCA} \) is slighter slower than \( vk'\text{-NNCA} \), as \( ek'\text{-NNCA} \) wastes some time on useless edges checking. Though \( k \)-Clustering, \( ek'\text{-NNCA} \) and \( vk'\text{-NNCA} \) all partition the graph into different components, and finds the cloaking set from the components. But \( k \)-Clustering checks the edges one by one and wastes more time for checking whether the partitioned component is connected when removing an edge.
Thus, $k$Clustering is worst among the algorithms.

6.3 Impact of conspired privacy level

![Fig. 12](image)

**Fig. 12** Impact of conspired privacy level (a) avg. success rate (b) avg. anonymization cost (c) avg. sacrificial cost (d) avg. time for finding cloaking set

Recall that the privacy profile of a user consists of an anonymity level $k$ and a conspired privacy level $p$. The effect of $k$ has been evaluated in the previous section. This section examines the impact of varying conspired privacy level $p$. Increasing $p$ implies that the privacy requirement becomes more constrained. Recall that the parameter $p$ is only used in generating cloaked region. Therefore, it is expected that $p$ has little influence on the cloaking time for finding cloaking sets. What might be affected by $p$ is the cloaking success rate and the average anonymization cost.

Fig. 12(d) validates that the cloaking time for finding cloaking set is not affected much by setting of $p$ as expected. The average cloaking time for $k^*\text{NNCA}$ ($ek^*\text{NNCA}$) is about $0.16\text{ms}$ ($0.17\text{ms}$). $k$Clustering is about two times slower than $vk^*\text{NNCA}$ and $ek^*\text{NNCA}$. And PNNCA is 1.5 times slower than the two algorithms. It is interesting to observe that from Fig. 12(a), the conspired privacy level also has little influence on the success rates, but the anonymization cost increases correspondingly (see Fig. 12(b)).

Fig. 12(b) shows that the average anonymization costs of all algorithms increase with conspired privacy level increasing as expected. It implies that mobile users generate a larger cloaked region to satisfy the more constrained privacy requirement. This phenomenon is also shown in Fig. 12(c). When mobile users require more privacy protection, more anonymization cost is sacrificed.

6.4 Impact of the proximity graph density

![Fig. 13](image)

**Fig. 13** Impact of density of proximity graph (a) avg. success rate (b) avg. anonymization cost (c) avg. sacrificial cost (d) avg. time for finding cloaking set

More edges implies more informative proximity information in the graph. Therefore, the success rates of PNNG, $vk^*\text{NNG}$ and $ek^*\text{NNCA}$ all increase slightly with $k^*$ increasing. However, the success rate of $k$Clustering increases first, and then decreases. With larger $k^*$, the average cluster size of $k$Clustering increases. Such that, even users are successful to find the candidate cloaking set in the the first phase of the cloaking algorithm, they would fail to generate cloaked region at the second phase of the algorithm, whose cloaked region area breaks through the system threshold. This can be evident from Fig. 13(b). The anonymization cost of $k$Clustering increases significantly with $k^*$ increasing.

In contrast, the anonymization costs for PNNG, $vk^*\text{NNG}$ and $ek^*\text{NNCA}$ increase very slowly. Comparing $vk^*\text{NNCA}$ and $ek^*\text{NNCA}$, the anonymization cost of the former is smaller than the later. When $k^*$ is larger than 12, the situation is reverse. The reason is as follows. When $k^*$ is small, the increase rate of the cloaking set size of $vk^*\text{NNCA}$ is faster than the one of $ek^*\text{NNCA}$. It implies more neighbors are
anonymized together in a same cloaking set in \( \text{v}^k\text{NNCA} \), such that the average anonymization cost is small. However, when \( k^* \) increases to be larger than 12, the increase rate of \( \text{ek}^k\text{NNCA} \) become larger. Therefore, \( \text{ek}^k\text{NNCA} \) shows better anonymization cost than \( \text{v}^k\text{NNCA} \) when \( k^* \) is larger than 12. From Fig. 13(c), we observe that increasing \( k^* \) has little effect on the average sacrificial cost.

From Fig. 13(d), we observe that \( \text{v}^k\text{NNCA} \) and \( \text{ek}^k\text{NNCA} \) beat \( k\text{Clustering} \) and PNNCA again w.r.t the cloaking time for finding cloaking set. For example, when \( k^* \) increases to 15, the cloaking time of \( k\text{Clustering} \) is about 3 times slower than \( \text{v}^k\text{NNCA} \) and \( \text{ek}^k\text{NNCA} \). For \( \text{v}^k\text{NNCA} \) and \( \text{ek}^k\text{NNCA} \), the cloaking time both decreases slightly first, then increases with \( k^* \) further increasing. That is because the cloaking time for finding cloaking set is divided into two parts: finding candidate cloaking set in \( k^*\text{NNG} \) and isolated users refinement in WPG. With \( k^* \) increasing, the time spending on finding candidates in \( k^*\text{NNG} \) increases. With more users finding the candidates successfully, fewer isolated users are left. Therefore, the time spending on isolated users refinement in WPG decreases. Overall, \( \text{v}^k\text{NNCA} \) and \( \text{ek}^k\text{NNCA} \) decrease first. When \( k^* \) becomes larger, the number of isolated users is stable. Then, the cloaking time is dominated by finding candidates in \( k^*\text{NNG} \). Therefore, the cloaking time presents a growth tendency with larger \( k^* \). Besides, comparing \( \text{vk}^k\text{NNCA} \) and \( \text{ek}^k\text{NNCA} \), their cloaking times are similar when \( k^* \) is small. With the number of edges increasing, \( \text{ek}^k\text{NNCA} \) spends more time on useless edge checking. Thus, \( \text{ek}^k\text{NNCA} \) is slower than \( \text{v}^k\text{NNCA} \) when \( k^* \) is larger than 12.

### 6.5 Impact of \( maxT \)

We know that a mobile user fails to be anonymized in two cases. One case is that it fails to find a candidate cloaking set in Section 4. And the other case is the user fails to find a cloaked region in Section 5. \( maxT \) is a system parameter, which is only used in generating the cloaked region. Fig. 14(a) shows the success rates under different settings of \( maxT \). We can see that the success rates increase with increasing \( maxT \) for all cloaking algorithms. This is because larger \( maxT \) implies a higher chance for generating a successful cloaked region. However, when \( maxT \) increases to a value (0.02 for PNNCA, 0.03 for \( \text{v}^k\text{NNCA} \) and \( \text{ek}^k\text{NNCA} \)), the success rate increases slowly. This implies that most of users who find cloaking sets could find cloaked regions successfully when \( maxT \) is large enough.

Fig. 14(b) shows that for all the cloaking algorithms the average anonymization cost increases by enlarging \( maxT \). With anonymization cost increasing, it is naturally expected that the sacrificial cost also increases. The changing trends of PNNCA, \( v^k\text{NNCA} \) and \( e^k\text{NNCA} \), shown in Fig. 14(c), prove our idea. However, it is interesting to observe that the sacrificial cost of \( k\text{Clustering} \) reduces with increasing \( maxT \). In fact, for \( k\text{Clustering} \) the candidate cloaking sets, which are generated under different \( maxT \), are same. Some farther users in the candidate cloaking set fail to find proper cloaked regions when \( maxT \) is small. When \( maxT \) increases, those farther users are successfully included in the cloaking set (see Fig. 14(a)), the success rate of \( k\text{Clustering} \) increases continuously with \( maxT \) increasing). The size of the cloaked region is much approximate to the MBR area. Therefore, the sacrificial cost of \( k\text{Clustering} \) is reduced.

![Fig. 14](image-url) Impact of \( maxT \) (a) avg. success rate (b) avg. anonymization cost (c) avg. sacrificial cost (d) avg. time for finding cloaking set
6.6 Scalability

We now evaluate the effect of number of users on the performance of cloaking algorithms. The number of users indicates the user density of the service area and the workload of the system. We vary the number of users from 10,000 to 100,000.

![Fig. 15](image-url)

**Fig. 15** Different dataset sizes (a) avg. success rate (b) avg. anonymization cost (c) avg. sacrificial cost (d) avg. time for finding cloaking set

As shown in Fig. 15(a), the success rates of all the four algorithms increase with increasing the number of users. This is mainly because of the increased user density, which implies high average vertex degree. Among four algorithms, k-Clustering is the one which is most sensitive to the user density. When there are 10,000 users, its success rate is only 73%. When data size increases to 100,000, its success rate is nearly to 96%. However, the success rates of our proposed algorithms ek*NNCA and vk*NNCA can arrive to 91% when the data size is only 10,000. This indicates that our proposed algorithms are more robust.

Fig. 15(b) shows the effect of the users number on the average anonymization cost. As mentioned above, increasing the number of users implies increasing the user density. When the user density is low, each user tends to find farther users cloaking together. As expected, the average anonymization costs of all the algorithms generally reduce when the number of users increases. For PNNCA, ek*NNCA and vk*NNCA, the sacrificed costs increase with the number of users increasing. This is obvious. When the users density increases, much nearer neighbors are cloaked together such that the MBR area of cloaking set reduces. However, for k-Clustering, the sacrificed cost reduces first and then increases slightly. From Fig. 15(a), we observe that the success rates of k-Clustering increases shapely when data size increases from 10,000 to 40,000. The average anonymization cost and the MBR area of the cloaking set both reduce. However, the decreased degree of the latter is smaller than the former, thus the sacrificed cost reduces from 10,000 to 40,000. When data size is larger than 40,000, like the other three algorithms, k-Clustering’s sacrificed cost increases slightly.

The cloaking time for finding cloaking set is shown in Fig. 15(d). A large number of users implies a heavy workload. The cloaking times of PNNCA and k-Clustering increase with the number of users increasing. For vk*NNCA and ek*NNCA, their cloaking times decrease when data size increases from 10,000 to 30,000, then increase when data size increases to 40,000 or more. The cloaking procedure consists of two steps for both algorithms. First, the k*NNG is partitioned into cliques. Then, two refinement steps, negative candidates and isolated users refinement, are employed. When user density is low, the average vertex degree is low. Thus, the size of cliques generated from k*NNG is small. Users in the clique cannot satisfy the anonymity requirement. Therefore, ek*NNCA and vk*NNCA spend more time for negative candidates refinement. When data size increases, most mobile users could directly find cloaking set from cliques, such that the cloaking time reduces. When the data size increases to 40,000 or more, ek*NNCA and vk*NNCA spend more time for finding cliques, since the increased number of vertices and edges. Therefore, the cloaking time increases then. It is clear that ek*NNCA and vk*NNCA outperform PNNCA and k-Clustering generally when data size is large.

6.7 Real datasets results

In this section, we adapt two different real datasets, the Athens trucks dataset [4] and the Geolife dataset [17], to evaluate the effectiveness and efficiency of our proposed algorithms.

The original trucks dataset consists of 276 trajectories of 50 trucks. We use each location on each trajectory of a truck represents a mobile user. Then we get more than 110,000 mobile users, who register in the CServer for LBS. We report the evaluation results in Fig. 16.

The performance trends on the success rate are similar to that of the simulation results in Section 6.2. As observed from Fig. 16(a), the success rates of four algorithms decrease with the increasing anonymity level. vk*NNCA and ek*NNCA outperform k-Clustering and PNNCA. The success rate of PN-NCA (k-Clustering) is dropped to 75% (87%) when k is set to 12. In contrast, vk*NNCA and ek*NNCA still get a success rate above 95% at the same setting.


\documentclass{article}

\begin{document}

\section{Conclusions}

In this paper, we investigated location privacy cloaking algorithms without exact locations over semi-honest users. We observed that most of existing location cloaking algorithms protect location privacy with exact locations, which are just the information mobile users want to hide. To address this problem, we have employed \( k^* \)-nearest neighbor graph (\( k^* \)NNG) to formalize the location proximity information and proposed two \( k^* \)NNG-based cloaking algorithms, called \( v^k \)NNCA and \( e^k \)NNCA. A series of experiments has been conducted to evaluate these algorithms under various system settings. The experimental results show that the price paid for location cloaking without exact locations is acceptable. The average cloaking time is only 0.16\( ms \) and the cloaking success rate is about 94\% for most cases, which validate the efficiency and effectiveness of the proposed algorithms.

\subsection*{Acknowledgements}

This research was partially supported by the grant from the Hebei Education Department (No. Q2012131).

\end{document}
References

Xiao Pan: Location privacy without exact locations

the 36th International Conference on Very Large Data Bases (VLDB). 2010, 985–996


Xiao Pan is a lecturer at Shijiazhuang Tiedao University. She received the PhD degree in computer science from Renmin University of China in 2010. Her research interests include data management on moving objects and location privacy protection.

Xiaofeng Meng is a professor in the School of Information, Renmin University of China. He received the BS degree from Hebei University, the MS degree from Renmin University of China, and the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, all in computer science. His research interests include mobile data management, Web data integration, native XML databases, and flash-based databases. He is the secretary general of Database Society of the China Computer Federation (CCF DBS). He has published more than 100 technical papers in refereed international journals and conference proceedings.