In the proceedings of the 17th ACM SIGSPATIAL International Conference on Advances
in Geographic Information Systems (GIS2009), page256-265, November 4-6, 2009, Seattle, Washington

Distortion-based Anonymity for Continuous Queries in Location-Based Mobile Services

Xiao Pan [1] Xiaofeng Meng [1] Jianliang Xu [2]
[1]School of Information
Renmin University of China
{smallpx,xfmeng}@ruc.edu.cn
[2]Dept of Computer Science
Hong Kong Baptist Unversity
xujl@comp.hkbu.edu.hk

ABSTRACT

Privacy preservation has recently received considerable attention for location-based mobile services. Various location cloaking approaches have been proposed to protect the location privacy of mobile users. However, existing cloaking approaches are ill-suited for continuous queries. In view of the privacy disclosure and poor QoS (Quality of Service) under continuous query anonymization, in this paper, we propose a δ_p-privacy model and a δ_q-distortion model to balance the tradeoff between user privacy and QoS. Furthermore, two incremental utility-based cloaking algorithms — *bottom-up cloaking* and *hybrid cloaking*, are proposed to anonymize continuous queries. Experimental results validate the efficiency and effectiveness of the proposed algorithms.

Categories and Subject Descriptors

H.2.m [**DATABASE MANAGEMENT**]: Miscellaneous-performance measures]

General Terms

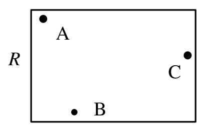
Algorithms, Performance, Information Privacy

Keywords

Privacy Protection, Continuous Queries, Location-Based Services

1. INTRODUCTION

With advances in wireless communication and mobile positioning technologies, location-based services (LBSs) have been gaining increasingly popularity in recent years. Research efforts have been put into investigating how to preserve the privacy of mobile users, while still ensuring high quality of LBSs. In general, there are two types of privacy issues: location privacy [6] (a sensitive location is protected from being linked to a specific user) and query privacy [4] (a query is protected from being linked to a specific user). For example, suppose Alice issues the following continuous query to the service provider (SP) (e.g., GoogleMap) via her mobile phone: "where is the nearest dermatosis hospital for next 30 minutes?". Concerning the location privacy, Alice wants to hide her exact location during her movement (e.g., being in a clinic or pub); concerning the query privacy, she wants to hide the fact that the above query about dermatosis hospital was issued by her.



(a)

User	User location	Cloaking region	Query content
A	l_A		Hospital
B	l_B	R	Hotel
C	l_C		Gas station

(b)

Figure 1: Location privacy and query privacy

To protect the location privacy, Gruteser and Grunwald [6] proposed spatio-temporal cloaking based on a location k-anonymity model, that is, the cloaked location is made indistinguishable from the location information of at least k-1 other users. To achieve location k-anonymity, each user location is extended to a cloaking region such that each region covers at least k users. Figure 1(a) illustrates an example of location 3-anonymity (k=3), where the locations of A, B and C are extended to region R(i.e., users A, B and C form a *cloaking set*), such that the adversary cannot figure out their genuine locations in R. Under some circumstances, the adversary knows the users' genuine locations [4]; thus, the location contained in a query would become a quasi-identifier (QI) [12] to link the query to a specific user. Fortunately, the location k-anonymity model is also applicable to tackle this query privacy issue. Consider the example shown in Figure 1(b), by simply extending the locations contained in the query to the same region R , the exact query locations can be successfully hidden and hence the query privacy is preserved.

Most of the existing cloaking algorithms focus on anonymizing *snapshot* queries [11, 5, 7]. As the cloaking sets for the same user are different at different timestamps [4], directly applying these algorithms to *continuous* queries is not sufficient to protect the query privacy. Figure 2 depicts an example where the query privacy is disclosed under continuous queries. As shown, six users A∼F issue six different continuous queries $Q_1 \sim Q_6$, respectively, and A is successfully cloaked as {A,B,D}, {A,B,F} and {A,E,C} at the timestamps t_i, t_{i+1} and t_{i+2}, respectively. Each of these cloaking sets is consistent to the location 3-anonymity. However, as their intersection contains A only, the adversary can easily infer that A issued query Q_1 and, hence, A's query privacy is disclosed.

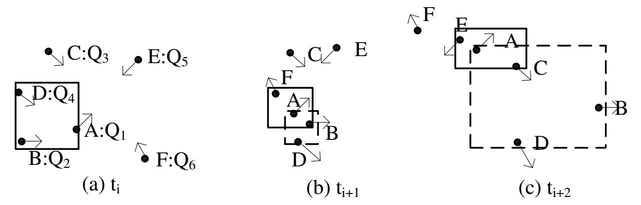


Figure 2: A continuous query example

As can be observed from the above example, the privacy disclosure is due to the use of different cloaking sets for the same user at different timestamps. To conquer this problem, queries issued from the same cloaking region should stick together at all timestamps [4]. In detail, under such a scheme, the cloaking set of A in Figure 2 should always be kept as {A, B, D} during t_i through t_{i+2} (the cloaking regions are represented as dashed rectangles in Figures 2(b) and 2(c)). Although this scheme successfully protects the query privacy, it leads to new problems: 1) User location privacy might be disclosed. As shown in Figure 2(b), the minimum bounding rectangle (MBR) of {A, B, D} shrinks to a smaller region at t_{i+1} which might violate the location privacy requirement of A. In the worst case, it may shrink to a point, thereby exposing the genuine user location. 2) The quality of service becomes poor. As shown in Figure 2(c), the size of the cloaking region {A, B, D} is significantly large at t_{i+2}, which would make the subsequent query processing much more costly. In an extreme case, the users might scatter over the entire space over time, forcing the cloaking region to cover the whole area.

The reason behind the aforementioned new problems is that, the algorithm exploits the proximity of current user locations only, but ignores their future locations. As we known, a user's future location depends on the velocity of her movement and the duration of the continuous query. In an ideal case, all users within the same cloaking set move with the same velocity such that the size of the cloaking region remains the same at all timestamps. Unfortunately, this is unlikely to happen in practice, and the location proximity tends to change once the involved locations update. Specifically, on one hand, queries whose locations are close at the current timestamp may become far away from each other at a future timestamp; on the other hand, queries who are now far away from each other may meet at some future timestamp. For a continuous query whose location is dynamically changing, it is hard to find an optimal cloaking region for all timestamps. The main challenge is how to achieve a good QoS while still preserving query privacy during the query period with frequent location updates.

In this paper, we consider protection of both location privacy and query privacy for both continuous queries. To address this issue, we propose a δ_p-privacy model and a δ_q-distortion model to balance the tradeoff between user privacy and QoS. The perimeter of a cloaking region is adopted to evaluate the distortion of the location information. As pointed out in [10], moving objects with similar patterns would move in a cluster eventually. Motivated by this observation, we propose to map the location distortion to a similarity distance of the queries, based on which queries are clustered such that the distortion of location information in each cluster is minimized. These clusters are incrementally maintained as queries move in and out.

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

- We propose a δ_p-privacy model and a δ_q-distortion model to balance the tradeoff between user privacy and QoS under continuous queries.
- We propose to map the location distortion to a temporal similarity distance of the queries. Furthermore, we propose two incremental utility-based cloaking algorithms.
- 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 problem under investigation is formally defined in Section 3. Several utility-based cloaking algorithms are proposed in Section 4. Algorithms for distortion and privacy verification are proposed in Section 5. Section 6 presents the performance evaluation results of our proposed algorithms. Finally, the paper is concluded in Section 7.

2. RELATED WORK

Location privacy and query privacy are two types of privacy issues concerned in location-based mobile services. Location k-anonymity is the most popular location privacy metric. It was proposed by Gruteser and Grunwald [6], and was later refined in [11, 1]. In terms of the techniques used for protecting location privacy, existing approaches can be classified into cloaking [6], dummy [8], and encryption [5]. However, all of the above work focuses on privacy protection for *snapshot* queries.

Most of the prior research does not distinguish location privacy and query privacy.The first work to distinguish them and to explore privacy protection for continuous queries is presented in [4]. Nonetheless, it has two drawbacks. First, only the query locations at the issuing time are employed to generate cloaking sets, which may lead to location privacy disclosures and poor QoS, as discussed in the previous section. Second, as the valid period of each query is ignored, continuous queries may be cloaked with snapshot queries. If any snapshot query moves out of the service area, all continuous queries within the same cloaking set may no longer meet the location k-anonymity privacy requirement. Our work also employs location k-anonymity and memorizes users, but it differs from [4] in the following aspects. First, a temporal location distortion model is employed to find the cloaking set. Second, the queries with similar expiration times are clustered together, which guarantees that every continuous query will always satisfy the privacy requirement during its valid period.

Another work addressing location anonymity for continuous queries is presented in [15]. It employs *entropy* to measure the anonymity level of a cloaking region by assuming that the probabilities of users in a cloaking region are not equal. However, as entropy does not consider whether the user locations are really different or not, location privacy might be disclosed when k different users are at the same location. In [13], a mobility-aware cloaking algorithm is proposed to defend trace analysis attacks. However, the privacy metric employed in [13] is location granularity, rather than location k-anonymity considered in this paper.

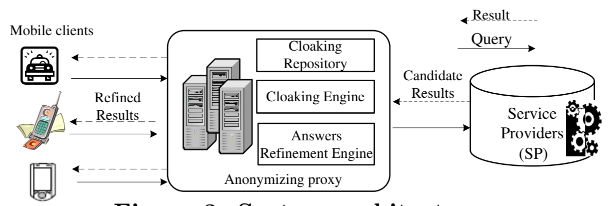


Figure 3: System architecture

Although most research on privacy protection in LBSs does not discuss about the location data utility, some of its measurements have been proposed in data publishing, such as *generalization height*, *discernibility*, *information loss*, *classification metric* and *information-gain-privacy-loss ratio* [9]. Data utility in data publishing mainly focuses on how the distribution of the original data is preserved for the purpose of data mining, whereas in LBSs, we mainly focus on how the generalized location approximates to the original location. Therefore, in this paper, we employ information loss, namely location distortion, as the utility measure. We remark that our location distortion is different from that of [14] in the following aspects. First, as QI attributes in data publishing are independent, each attribute can be associated with a weight to reflect its importance. Nevertheless, the information in LBSs, including location (x, y) and velocity v, is dependent to each other w.r.t. time t. Second, the information distortion in [14] is static as long as the anonymizing table is given, while the location distortion in our paper is a temporal function, which changes as time evolves.

3. PRELIMINARIES

3.1 System Architecture

Like most existing work [6, 11], we employ a centralized system, which consists of mobile users, a trusted anonymizing proxy, and an un-trusted SP, as shown in Figure 3. Each mobile user sends location-based queries to the anonymizing proxy. There are two types of queries: *new query* and *active query*. New query, as the name implies, is a query newly issued by a user. Active query is a continuous query which was issued at some previous time but not yet expired. For example, a user issues a continuous query Q at t_i, and its valid period is Δt. Then, at t_i, Q is a new query, while for any $t \in (t_i, t_i + \Delta t]$, it is regarded as an active query.

The anonymizing proxy consists of *cloaking engine*, *cloaking repository* and *answer refinement engine*. Upon receiving a new query, *cloaking engine* replaces the user *id* with a pseudonym id'. Meanwhile, it invokes the location cloaking algorithm to generate a cloaking region in accordance with the user's privacy requirement. This cloaking set is saved in the cloaking repository in the form of $(CID, Qset, R_{L,t}, R_{v,t})$[1]. Upon receiving an active query, *cloaking engine* searches for the original cloaking set, which was generated at the issuing time, in the cloaking repository and then computes the new cloaking region $R_{L,t}$. Later on, the anonymizing proxy forwards the cloaked query to the SP. By maintaining a cloaking repository, the anonymizing proxy can incrementally compute the cloaking set (i.e., by updating the original cloaking region) for the active queries and thereby achieving a higher efficiency.

Finally, candidate results generated by SP are first refined by *answer refinement engine*, and then relayed to the mobile user. In this paper, we focus on the location cloaking algorithm, which considers the location data utility as well as the user-specified privacy requirements.

[1]The meanings of these parameters will be explained later in Definition 2.

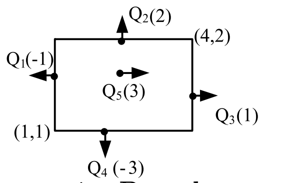


Figure 4: Boundary velocities

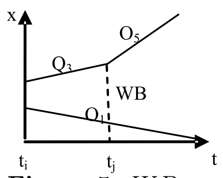


Figure 5: WB

To facilitate our study, we further make the following assumptions. 1) *Every mobile user is trusted* — this is a common assumption in the conventional location privacy preserving techniques [6, 11, 1]. 2) *The movement velocity of a user remains unchanged during the query period.* Under this assumption, the movement function of every query is linear.

3.2 Preliminaries

Definition 1. (Location-based query) Query Q is represented as $Q = (l, \bar{v}, t, T_{exp}, con)$, where $(l, \bar{v}, t)$ implies Q is at location l=$(x,\ y)$ with velocity $\bar{v} = (v_x, v_y)$ at the timestamp t, T_{exp} is the timestamp when the query expires, and con is the content of this query.

Definition 2. (Cloaking set) Each cloaking set CS is formalized as:

$$CS = (CID, Qset, R_{L,t}, R_{v,t}),$$

where CID is the identifier of this cloaking set, $Qset$ is the set of queries contained in CS, $R_{L,t} = (L_{x-,t}, L_{y-,t}, L_{x+,t}, L_{y+,t})$ is the location MBR for the queries in $Qset$ at t, and $R_{v,t}$= $(v_{xmin,t},\ v_{ymin,t}, v_{xmax,t},\ v_{ymax,t})$ is the BVR (Boundary Velocity Rectangle). $v_{xmin,t}$=min$(v_{x+,t},\ v_{x-,t})$, $v_{xmax,t}$=max$(v_{x+,t},\ v_{x-,t})$, $v_{ymin,t}$=min$(v_{y+,t},\ v_{y-,t})$, $v_{ymax,t}$=max$(v_{y+,t}, v_{y-,t})$, where $v_{x-,t}$ $(v_{x+,t})$ is the boundary velocity of the query at $L_{x-,t}$ $(L_{x+,t})$ on the x dimension, and $v_{y-,t}$ $(v_{y+,t})$ is the boundary velocity of the query at $L_{y-,t}$ $(L_{y+,t})$ on the y dimension at time t.

Note that $v_{xmax,t}$ $(v_{ymax,t})$ and $v_{xmin,t}$ $(v_{ymin,t})$ may not be the maximum and minimum velocity in $Qset$ on the x (y) dimension. Hence, the queries on the boundary of a cloaking set change with users movements. Thus, $R_{v,t}$ and $R_{L,t}$ are both piece-wise functions w.r.t. t, i.e.:

$$(L_{x-,t}, L_{y-,t}) = (L_{x-,t_{i-1}}, L_{y-,t_{i-1}}) + (v_{x-,t}, v_{y-,t}) \times [t - t_{i-1}]$$
$$(L_{x+,t}, L_{y+,t}) = (L_{x+,t_{i-1}}, L_{y+,t_{i-1}}) + (v_{x+,t}, v_{y+,t}) \times [t - t_{i-1}] \quad (1)$$

where $t \in [t_{i-1}, t_i]$. Taking Figure 4 as an example, queries $Q_1 \sim Q_5$ constitute a cloaking set CS at time t_i. The number in parenthesis is the query's velocity and the arrow indicates the movement direction. $CS.R_{L,t_i}$=(1,1,4,2) and $CS.R_{v,t_i} = (-1, -3, 1, 2)$. Here the maximum speed is 3 on the x dimension (for Q_5), but it is not a boundary speed at time t_i. If Q_5 overtakes Q_3 at time t_j, v_{xmax} will change to 3 at t_j.

Definition 3. (Width/Height of boundary) For a cloaking set CS with MBR $R_{L,t}$, its width at time t, denoted as WB_t, is

$$WB_t = L_{x+,t} - L_{x-,t} \quad (2)$$

Similarly, its height at time t, denoted as HB_t, is

$$HB_t = L_{y+,t} - L_{y-,t} \quad (3)$$

WB_t/HB_t is also a piece-wise linear function. For the example in Figure 4, the changing trend of WB is shown in Figure 5. The changing trend of HB is similar; so we omit it here.

Recall that privacy is protected by reducing the resolution of the location information. Obviously, the more does the privacy preserve, the less is the utility of the location data. In this paper, we use distortion to measure the utility of anonymized location. In other words, distortion reflects the location information loss, i.e., how much is sacrificed for privacy preserving. The smaller is the distortion value, the higher is the data utility.

Definition 4. (Distortion of a query) Let query $Q \in CS$, whose MBR (BVR) at time t is denoted as $R_{L,t}$ $(R_{v,t})$. Let A_{width} (A_{height}) be the width (height) of the whole space. The distortion for a query Q at time t is defined as

$$Distortion_{R_{v,t}}(Q, R_{L,t}) = \frac{(L_{x+,t} - L_{x-,t}) + (L_{y+,t} - L_{y-,t})}{A_{height} + A_{width}}.$$

Thus, the distortion of Q during its valid period can be represented as

$$\int_{T_s}^{T_{exp}} Distortion_{R_{v,t}}(Q, R_{L,t})dt \qquad (4)$$

where T_s is the timestamp when Q is cloaked successfully.

For the sake of convenience, let $P_A = A_{height} + A_{width}$, $P_{L,t} = (L_{x+,t} - L_{x-,t}) + (L_{y+,t} - L_{y-,t})$, $P_{v,t} = (v_{x+,t} - v_{x-,t}) + (v_{y+,t} - v_{y-,t})$. Let $TSet$ denote the set of timestamps $\{t_1, t_2, \cdots, t_n\}$ $(t_1 = T_s, t_n = T_{exp})$ when the boundary queries change. Then, Equation (4) can be rewritten as

$$\begin{aligned}&\int_{T_s}^{T_{exp}} Distortion_{R_{v,t}}(Q, R_{L,t})dt \\ &= \frac{1}{P_A}\{\int_{t_1}^{t_2}[P_{L,t_1} + P_{v,t_1}(t - t_1)]dt + \\ &\cdots + \int_{t_{i-1}}^{t_i}[P_{L,t_{i-1}} + P_{v,t_{i-1}}(t - t_{i-1})]dt + \\ &\cdots + \int_{t_{n-1}}^{t_n}[P_{L,t_{n-1}} + P_{v,t_{n-1}}(t - t_{n-1})]dt\}\end{aligned}$$

Definition 5. (Distortion of a cloaking set) Let CS be a cloaking set with MBR $R_{L,t}$ and BVR $R_{v,t}$ at time t. The distortion of CS at time t is defined as

$$Distortion_{R_{v,t}}(CS, R_{L,t}) = \sum_{Q_i \in CS} Distortion_{R_{v,t}}(Q_i, R_{L,t})$$

Thus, the distortion of CS during its valid period is defined as

$$\int_{T_s}^{maxT} Distortion_{R_{v,t}}(CS, R_{L,t})dt \qquad (5)$$

where T_s is the timestamp when CS is generated and $maxT = max_{Q_i \in CS}(Q_i.T_{exp})$.

For any two queries, if their states (i.e., initial locations and velocities) are similar, their future locations tend to be near to each other. In extreme cases, if two queries are on the same initial location with the same velocity, they will have the same location during their common valid periods. This implies that if queries with similar states are cloaked together, their distortions during the valid period are likely to be small. This observation inspires us to map the distortion with the similarity distance of queries, as defined as follows:

Definition 6. (Temporal similarity distance between two queries) Let Q_1 and Q_2 be two queries, and they constitute a cloaking set CS_{12} with MBR (BVR) $R_{L_{12},t}$ $(R_{v_{12},t})$ at time t.

The temporal similarity distance between Q_1 and Q_2 is defined as

$$SimDis(Q_1, Q_2) = \int_{T_s}^{maxT} Distortion_{R_{v_{12},t}}(CS_{12}, R_{L_{12},t})dt$$

where $maxT$=max$(Q_1.T_{exp}, Q_2.T_{exp})$.

The similarity distance possesses the following properties:

- SimDis(Q_1,Q_1)=0
- SimDis(Q_1, Q_2)=SimDis(Q_2, Q_1)
- SimDis(Q_1, Q_2)$\leq$ SimDis(Q_1, Q_3)+SimDis(Q_3, Q_2)

The proof is obvious. Due to space limitations, we omit it here.

Definition 7. (Temporal similarity distance between two query sets) Let U_1 and U_2 denote two non-interleaved query sets (i.e., $U_1 \cap U_2 = \phi$), and $U = U_1 \bigcup U_2$. $R_{L,t}$ $(R_{v,t})$ denotes the MBR (VBR) of U at time t.

The similarity distance between U_1 and U_2 is defined as

$$\begin{aligned}SimDis(U_1, U_2) = &\int_{T_s}^{maxT} Distortion_{R_{v,t}}(U_1, R_{L,t})dt + \\ &\int_{T_s}^{maxT} Distortion_{R_{v,t}}(U_2, R_{L,t})dt\end{aligned}$$

where $maxT = max_{Q \in U}(Q.T_{exp})$. It is easy to know that the similarity distance between two queries can be regarded as the special case for two query sets where $|U_1| = |U_2|$=1.

3.3 Privacy Model

Recall that the queries within the same cloaking set are required to stick together before they expire, which makes it hard to strike a good balance between the user privacy and QoS for continuous queries. In this section, two models, namely, *δ_p-privacy* and *δ_q-distortion*, are proposed to formalize user privacy and QoS requirements.

The location and velocity of a query are projected to the x and y dimensions. Now let us first discuss a one-dimension case. Assume that a candidate cloaking set contains three queries $\{Q_1, Q_2, Q_3\}$, whose velocities on the x dimension are shown in Figure 6(a). From the figure, we can see that WB decreases in early stage and shrinks to a point at T_w; after that, WB increases again. Similar observations can be drawn on the y dimension. In the worst case, two boundary segments on the x dimension and y dimension would shrink at same time, as shown in Figure 6(b). Consequently, the cloaking region would shrink to a point, which leads to the exposure of the genuine location.

Location disclosures are prohibited, regardless of how many dimensions they are disclosed on. If neither WB nor HB has the opportunity to shrink to a point, apparently, the privacy would be preserved. A privacy model is designed to formalize how much WB and HB are allowed to shrink.

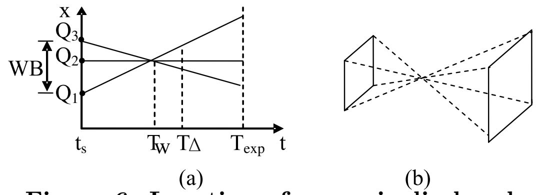


Figure 6: Location of query is disclosed

Definition 8. (δ_p-privacy model) Let WB_t (HB_t) be the width (height) of the boundary segment on the x (y) dimension at time t, and δ_p be the privacy threshold specified by the users. If for $\forall t \in [T_s, maxT]$, $\min(WB_t, HB_t) \geq \delta_p$, δ_p-privacy is satisfied.

While Definition 8 guarantees the protection of user privacy, the following Definition 9 ensures the quality of services. The distortion of a query set CS should not grow larger than the user's requirement δ_q, which is the user's tolerable worst service quality. Note that forcing the distortion to be smaller than δ_q at T_s cannot ensure that it always meets the requirement δ_q during the entire period T_s through $maxT$.

Definition 9. (δ_q-distortion model) Assume that the user's tolerable worst service quality is δ_q, the set of user queries is CS with MBR $R_{L,t}$ and VBR $R_{v,t}$. If for any $t \in [T_s, maxT]$ and any $Q \in CS$, $Distortion_{R_{v,t}}(Q, R_{L,t}) \leq \delta_q$, then δ_q-distortion is satisfied.

In summary, the requirements for a successful cloaking set CS include:

- $|CS| \geq K$;
- Let $minT = min_{Q \in CS} Q.T_{exp}$, $maxT = max_{Q \in CS} Q.T_{exp}$, $maxT - minT \leq \delta_T$;
- CS satisfies δ_p-privacy and δ_q-distortion during $[T_s, maxT]$.

4. UTILITY-BASED ALGORITHMS

4.1 Greedy Cloaking Algorithm

The main idea of the greedy cloaking algorithm (GCA) is as follows. For every newly arrived query r, we first compute its temporal similarity distance with those existing queries which have not yet been anonymized successfully. Then, the one having the minimal similarity distance with r is put into the cloaking set. The above steps repeat until no more queries can be added into the cloaking set. We detail it in Algorithm 1.

Specifically, when a new query r arrives, it is first inserted into the candidate cloaking set U (see step 1∼2). Then each query r_m in the set of existing queries which are not yet anonymized (denoted as $RSet$) is retrieved. If the difference between $r_m.T_{exp}$ and $r.T_{exp}$ is bigger than the value of δ_T, then r_m cannot be clustered with r and therefore it is filtered out (see step 5). Otherwise, the boundary queries during its valid period are calculated and each boundary query is stored in the **b**oundary **t**ime **q**ueue (BTQ) bq, which would be detailedly elaborated in Section 5.1. After all boundary queries are captured, the δ_q-distortion requirement (the detailed procedure is given in Section 5.2) is verified. If true, the request r_{min} with the minimal temporal similarity distance is inserted into U. The above steps repeat until no more queries can be put into U or $|U| \geq K$. Finally, if U covers more than K users, the δ_p-privacy is also verified (we will present its detailed procedure in Section 5.3). If both δ_q-distortion and δ_p-privacy are satisfied, U is returned as the cloaking set.

Algorithm 1 : Greedy cloaking algorithm(GCA)

```
1: a candidate cloaking set $U$=null;
2: put $r$ into $U$;
3: while true do
4:   for each query $r_m$ in $RSet$ do
5:     if $|r.T_{exp} - r_m.T_{exp}| > \delta_T$ then
6:       get the next query in $RSet$;
7:     else
8:       BoundaryObjectsComputing($r_m, bq, U$)
9:       if DistortionDetection($r_m, bq, U$)=true then
10:        $dis$=SimDis($r_m, U$);
11:        if ($mindis > dis$) then
12:          $mindis = dis$;
13:          $r_{min} = r_m$;
14:  insert $r_{min}$ into $U$;
15:  $RSet = RSet - \{r_{min}\}$;
16:  if $|U|$ does not change or $|U| \geq K$ then
17:    break;
18: if ($|U| \geq K$) then
19:   Check $\delta_p$-privacy and return cloaking set;
```

4.2 Bottom-up Cloaking Algorithm

The drawback of GCA is that for every newly arrived query, it needs to search the cloaking set from scratch, which incurs expensive computational cost. Actually, intermediate results computed in the previous iterations can be exploited. The basic mechanism is to cluster those queries whose distortions are always less than δ_q together during their valid periods, and then to incrementally maintain these clusters. Obviously, the cloaking sets are the subsets of these clusters. Before presenting our new cloaking algorithm, we give the definition of continuous cluster:

Definition 10. (Continuous cluster) A query set C is a continuous cluster during $[t_1, t_2]$ if (1) C satisfies δ_q-distortion; (2) $maxT_{exp} - minT_{exp} \leq \delta_T$, where $maxT_{exp} = max_{Q \in C}(Q.T_{exp})$ and $minT_{exp} = min_{Q \in C}(Q.T_{exp})$

Based on the continuous clusters, the basic idea of *bottom-up cloaking* algorithm (BCA) is as follows. When a new query r arrives, r itself forms a cluster $\{r\}$, which naturally satisfies δ_q-distortion. Then, among the existing continuous clusters, the one with the minimal temporal similarity distance with r, denoted as c_r, is selected to merged with $\{r\}$. If $\{c_r, r\}$ contains not less than K queries, it is verified to see if it meets the δ_p-privacy requirement. If fails, this cluster is kept in service space for merging future queries. Algorithm 2 shows the pseudo-code of *bottom-up cloaking* upon the arrival of a new query r.

In Algorithm 2, r is the newly arrived query, and CR is the set of existing continuous clusters in service space. For each cluster c in CR, it associates with a BTQ bq_c, which maintains its boundary queries during the valid period. When r arrives, each cluster c in CR is scanned and the following steps are conducted. First, new boundary queries of c with r inserted are computed (see step 5); Second, δ_q-distortion is checked (see step 6); Third, the temporal similarity distance between c and r is calculated (see step 8). After that, the

Algorithm 2 : Bottom-up cloaking algorithm(BCA)

```
1: for each cluster $c \in CR$ do
2:   $maxT$=max($c.maxT_{exp}$, $r.T_{exp}$);
3:   $minT$=min($c.maxT_{exp}$, $r.T_{exp}$);
4:   if $maxT - minT \leq \delta_T$ then
5:     BoundaryObjectsComputing($r, bq_c, c$);
6:     if DistortionDetection($r, bq_c, c$)=false then
7:       continue;
8:     $dist$=SimDis($r, c$);
9:     if $dc_{min}$>$dist$ then
10:      $dc_{min}$=$dist$;
11:      $c_{min} = c$;
12:    else
13:      if $dc$=$dist$ and $|c_{min}| < |c|$ then
14:        $dc_{min}$=$dist$;
15:        $c_{min}$=$c$;
16: if $c_{min}$ not exists then
17:   put $\{r\}$ into $CR$;
18: else
19:   $c_{min} = c_{min} \cup \{r\}$;    /*do the merging*/
20:   if $|c_{min}| \geq K$ then
21:     Check $\delta_p$-privacy and return the cloaking set $c_{min}$;
```

cluster c_{min} which has the minimal temporal similarity distance with r is sought. If such c_{min} exists, c_{min} is updated by merging it with $\{r\}$. Otherwise, r itself forms a cluster and is added into CR. Note that if there exist two clusters with the same minimal similarity distance with r, the one with more queries is preferable to be chosen for $\{r\}$ to be merged with (see step 13∼15) so that more queries can be cloaked.

Consider the example shown in Figure 7, where C_1, C_2, C_3 and C_4 are continuous clusters, whose distortions are always smaller than δ_q before they expire. When a new query r arrives, according to BCA, C_4 is selected to be merged with $\{r\}$ as it has the minimal temporal similarity distance with r. Then, C_4 would have four queries after the merging is conducted. If $K \leq 4$, C_4 is sent to check the δ_p-privacy requirement, otherwise, it will stay in the service space to merge with other arriving queries.

4.3 Hybrid Cloaking Algorithm

Many possible cluster merges are ignored in *bottom-up cloaking* algorithm (BCA), which might decrease the success rate of cloaking. Continue with the example in Figure 7, as discussed previously, C_4 cannot be returned as a cloaking set if K=5. However, by merging some queries of other cluster, e.g., C_1, it is possible that C_4 would successfully become the cloaking set. In fact, $C_4 \cup \{A\}$ satisfies both δ_q-distortion and δ_p-privacy requirements, and thus would be returned as a cloaking set. However, such opportunities are omitted in BCA. On the other hand, as cluster merges are time-consuming, especially when the locations of queries are frequently updating, conducting merges in the granularity of queries surely deteriorates the performance of the cloaking algorithms.

In order to resolve this problem, we propose *hybrid cloaking* algorithm, which aims to combine the advantages of BCA and GCA together. Specifically, BCA is used to search the proper cluster for each newly arrived query, while GCA is for cluster refinement. To further improve the searching efficiency, a TPR-tree is adopted to index existing clusters,

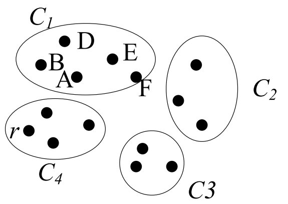


Figure 7: BCA

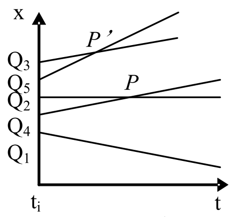


Figure 8: Location on x-dimension

such that some clusters can be filtered out and thereby accelerating the searching process. Before presenting the specific algorithm, for ease of exposition, we introduce *nearest neighbor cluster* whose definition is as follows:

Definition 11. (Nearest neighbor cluster, NNC) A cluster C_n is the NNC of C iff for any cluster $C_i(C_i \neq C$ and $i \neq n)$, perimeter(MBR(C_i, C))> perimeter (MBR(C_n, C)).

With the help of TPR-tree, we can easily get the set of nearest neighbor clusters (denoted as CS_{nn}) for each newly arrived query r during its valid period. The following theorem shows that, if there exists a cluster C in CS_{nn} and C's distortion with r violates δ_q-distortion, then the distortions between other clusters and r must also violate δ_q-distortion.

THEOREM 1. *Let CS_{nn} be query r's CNN during its valid period $[T_s, T_{exp}]$. If $\exists$ $C_{i,t_i,t_{i+1}} \in CS_{nn}$,*

$$distortion(C_{i,t_i,t_{i+1}}, r, t) > \delta_q$$

where $t \in [t_i, t_{i+1}], T_s < t_i < t_{i+1} < T_{exp}$, then for any cluster $C'(C' \neq C_{i,t_i,t_{i+1}})$,

$$distortion(C', r, t) > \delta_q.$$

PROOF. For ease of presentation, without loss of generality, we assume every cluster has the same valid period as r. CS_{nn} is in the form of $\{(C_{1,t_1,t_2}, t_1, t_2), \ldots, (C_{i,t_i,t_{i+1}}, t_i, t_{i+1}), \ldots, (C_{n,t_{n-1},t_n}, t_{n-1}, t_n)\}$, where $t_1 = T_s$, $t_n = T_{exp}$, and (C_i, t_i, t_{i+1}) represents that $C_{i,t_i,t_{i+1}}$ is r's NN during $[t_i, t_{i+1}]$. For any cluster C' which is not NN at timestamp t ($t \in [t_i, t_{i+1}]$), according to Definition 11, perimeter(C', r) > perimeter ($C_{i,t_i,t_{i+1}}, r$) at t. Apparently, if distortion $(C_{i,t_i,t_{i+1}}, r, t)$ > δ_q, we have distortion (C', r, t)> δ_q. □

According to Theorem 1, for a newly arrived request r, if its nearest neighbor cluster at any timestamp of its valid period violates the δ_q-distortion requirement, other clusters are filtered out from checking and thus some computational cost can be saved. However, two new problems arise: 1) how to efficiently find CNN of r during its valid period; 2) how to find the cloaking set based on CS_{nn}.

As the queries within a cluster are dynamically changing in terms of their locations and velocities, under such circumstance, it is a complicated and time-consuming to identify the CS_{nn} for a query. However, since the purpose of CS_{nn} is just for filtering, instead of finding out the exact nearest neighbor for a query, we turn to the approximate computing. Before that, we first define the centroid of a cluster.

Definition 12. (Centroid of a cluster) The centroid O_{cn} of a cluster C is represented as (x, y, v_x, v_y), where (1)$x = \frac{\sum_{Q \in C} Q.x}{|C|}$ and $y = \frac{\sum_{Q \in C} Q.y}{|C|}$; (2) $v_x = \frac{\sum_{Q \in C} Q.v_x}{|C|}$ and $v_y = \frac{\sum_{Q \in C} Q.v_y}{|C|}$

Algorithm 3 : Hybrid cloaking algorithm(HCA)

```
1: find the CNN cluster $CS_{nn}$ for $r$ on TPR-tree;
2: invoke BCA on $CS_{nn}$ to find cluster $c_{min}$;
3: if $c_{min}$ not found then
4:     insert $r$ into TPR-tree;
5: else
6:     $c_{min} = c_{min} \cup \{r\}$;    /*do the merging*/
7:     if $|c_{min}| < K$ then
8:         for each query $o$ in $CS_{nn} - c_{min}$ do
9:             $maxT$=max($c_{min}.maxT_{exp}$, $o.T_{exp}$);
10:            $minT$=min($c_{min}.minT_{exp}$,$o.T_{exp}$);
11:            if $|maxT - minT| < \delta_T$ then
12:                if DistortionDetection($o, bq, c_{min}$)=false then
13:                    insert $o$ into $c_{min}$;
14:                    delete $o$ from the cluster $c$ it is in;
15:                    insert $c$ into queue $uq$;
16:                    if $|c_{min}| \geq K$ then
17:                        break;
18:     for each cluster $c$ in $uq$ do
19:         update centroid of cluster $c$ ;
20:         update it in TPR-tree;
21:     if $|c_{min}| \geq K$ then
22:         Check $\delta_p$-privacy and return the cloaking set $c_{min}$;
23:     else
24:         insert centroid of $c_{min}$ into TPR-tree;
```

Now, each cluster can be simply represented by its centroid, and a TPR-tree can be built on the centroids of clusters. Also, the NNC for a query can be quickly discovered by searching over this TPR-tree. In other words, the original problem is successfully transformed to a traditional CNN problem on moving objects, which has been well studied in literature. We detail the proposed *hybrid cloaking* in Algorithm 3. We employ best-first traversal using min metric [2] to compute CS_{nn} for r (step 1). Based on CS_{nn}, BCA is adopted to find the cluster c_{min} with minimum temporal similarity distance first (step 2). If such c_{min} does not exist, r itself forms a cluster and its centroid is inserted into TPR-tree (step 4). Otherwise, the merging of $\{r\}$ and c_{min} is conducted. Finally, if c_{min} has not less than K queries, it is directly returned as a candidate cloaking set for privacy verification (step 22). Otherwise, we invoke GCA to do the cluster refinement (step 7∼17) — for each query o in CS_{nn} but not in c_{min}, if $c_{min} \cup \{o\}$ satisfies δ_q-distortion, then o is moved into c_{min} and their centroids in TPR-tree are updated correspondingly. Such process repeats until c_{min} contains K queries.

Continue with the example shown in Figure 7. Suppose that the set of clusters $\{C_1, C_2, C_3, C_4\}$ is the CNN (denoted as CS_{nn}) found on the TPR-tree when r arrives, and K is equal to 5. After invoking BCA, C_4 is selected to merge with $\{r\}$. As C_4 only contains four queries after merging, cluster refinement is conducted. Specifically, each query o in $\{C_1, C_2, C_3\}$ is checked to see if it can be inserted into c_{min}. Hence, in this example, A is found and is moved from C_1 to C_4. Consequently, the centroid of C_1 is updated and C_4 is removed from the TPR-tree. Finally, $C_4 \cup \{A\}$ is returned as a valid cloaking set.

5. DISTORTION AND PRIVACY VERIFICATIONS

The proposed cloaking algorithms, namely, GCA, BCA and HCA, involve three main steps: boundary query computing, δ_q-distortion and δ_p-privacy verifications. We will elaborate them in Section 5.1, Section 5.2 and Section 5.3, respectively.

5.1 Boundary Query Computing

As discussed previously, each cluster is associated with a queue BTQ[2], which maintains the boundary queries at different timestamps. Each item in BTQ is in the form of <*time*, *query*>, where *time* is the timestamp when this *query* becomes boundary. Inside BTQ, the boundary queries are kept sorted in ascending order of the timestamp. As queries are changing along with the movement of its issuers, it is costly to track the boundary queries online.

Figure 8 shows five queries projected on the x-dimension. For the timestamps $t_i \sim t_j$, each query at the timestamp t ($t < t_j$) can be located by

$$x = x_{t_i} + v_x * (t - t_i) \tag{6}$$

Hence, the crossed points of lines in Figure 8 can be easily computed. Note that we only need to compute those crossed points which can contribute to the width of the boundary's segments. In Figure 8, the crossed point P can be ignored. Although we take the case on the x-dimension as an example, the case on the y-dimension can be handled similarly. For every cluster C, let $VS+/VS-$ be the velocity sets of boundary queries on positive/negative x-dimension during its valid period. The main idea of boundary query computing is as follows: when a query r is inserted into C, if $\forall v_+ \in VS+$, $r.v_x < v_+$, and $\forall v_- \in VS-$, $r.v_x > v_-$, r is impossible to be the boundary on the x-dimension. Otherwise, the timestamp when r becomes the boundary is computed by using equation (6). In addition, those crossed points are inserted into BTQ. For the example in Figure 8, if Q_3 is the query to be inserted, as Q_5's velocity is larger than Q_3's, it would be picked up to compute the crossed point P'. In this way, all boundary queries of a cluster can be calculated. Due to space limitations, we omit the detailed algorithm.

5.2 δ_q-Distortion Verification

By maintaining BTQ, it is easy to get any boundary query at any time for a cluster. Therefore, during two consecutive timestamps $[t_i, t_{i+1}]$ in BTQ, as each boundary movement is a linear function with timestamp t, $P_{L,t}$ and $P_{v,t}$ can be computed. To satisfy δ_q-distortion, for any timestamp $t \in [t_i, t_{i+1}]$, the following inequation should be held:

$$\frac{1}{P_A}[P_{L,t_i} + P_{v,t_i}(t - t_i)] < \delta_q. \tag{7}$$

By setting the left side of the inequation (7) to be δ_q, we can compute the upper bound of t, denoted as t^+. If t^+ locates in $[t_i, t_{i+1}]$, it is easy to know that δ_q-distortion is violated. Otherwise, δ_q-distortion is satisfied. The algorithm is quite straightforward and thus is omitted here.

5.3 δ_p-Privacy Verification

Like δ_q-distortion, δ_p-privacy can be verified during each time interval between two consecutive timestamps in BTQ by using equations (2) and (3). However, it is not necessary

[2]In the implementation, there are four queues, namely, $BTQ_{x-}, BTQ_{x+}, BTQ_{y-}, BTQ_{y+}$, which maintain the boundary queries on every direction of the cluster MBR. For ease of presentation, we unify them as a queue BTQ.

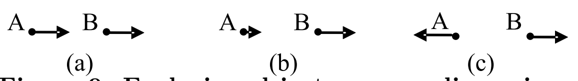


Figure 9: Exclusive objects on one dimension

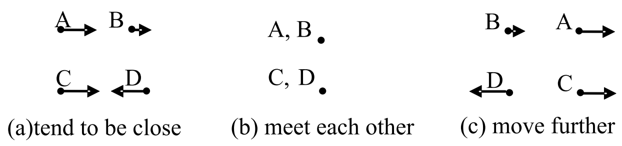


Figure 10: Non-exclusive objects on one dimension

for δ_p-privacy to be checked during every time interval of the valid period.

Let's define *exclusive* first:

Definition 13. (Exclusive) Let *dist* be the distance between two queries r_1 and r_2. For any two timestamps t_i, t_j, $t_i < t_j$, if dist$(r_1, r_2, t_i) \leq$dist(r_1, r_2, t_j), we say these two queries are exclusive to each other.

Intuitively, the distance between two exclusive queries would increase as the time elapses. Moreover, two observations can be drawn. First, if two boundary queries on each dimension are exclusive and WB/HB is larger than δ_p at t_i, it will not be less than δ_p after t_i. Second, for any two boundary queries, even if they are not exclusive to each other at the current timestamp, they will become exclusive at some future time. Figure 9 shows the cases when two queries are exclusive. (a) Two queries have the same velocity, thus the distance between them remains constant. (b) The velocities of two queries A and B have the same direction, but B, which is in front of A on the moving direction, has a larger velocity. Apparently, (a) is a special case of (b). (c) Two queries move on the opposite direction. Figure 10 demonstrates the cases when two queries are not exclusive at the early stage but become exclusive to each other after a certain timestamp. As shown, the queries A and B (C and D) are not exclusive in (a). However, as time goes by, after A and B converge in (b), they become exclusive to each other (e.g., in (c)).

The main idea of δ_p-privacy verification is as follows. If the boundary queries of a cluster on both dimensions are exclusive, the actions are subject to different cases: when both WB and HB are larger than δ_p, the cluster can be returned as a cloaking set directly; when WB or HB is less than δ_p, we can delay to the time to publish the cloaking set, until both $|WB|$ and $|HB|$ are larger than δ_p. If the boundary queries on any dimension are not exclusive, their information are kept on tracking in the BTQ until they are exclusive or all boundary queries in BTQ have been checked. The former terminal condition implies this cluster is a successful cloaking set, while the latter one indicates this cluster should remain in the service space and wait for anonymization. Due to space limitations, we omit the specific algorithm here.

Figure 6(a) shows an example when the cloaking set needs to be delayed for publishing. Assume $Q_1 \sim Q_3$ constitute a cluster at $t_s(K{=}3)$. As shown, WB shrinks to zero at T_w, and it increases to δ_p at $T_\triangle$. Therefore, this cluster would wait until $T_\triangle$ is to be published as a cloaking set.

6. PERFORMANCE EVALUATION

In this section, the effectiveness and efficiency of our proposed algorithms, including GCA, BCA, and HCA, are experimentally evaluated under various system settings. Although the privacy technique proposed in [4] is the most representative approach for continuous queries, as discussed in Section 2, it suffers from disclosing the location privacy in some cases. Moreover, none of the existing cloaking algorithms consider the temporal utility of the cloaking region during the anonymization. Hence, we do not include any existing cloaking algorithm for comparison. The evaluation metrics include the cloaking success rate, the cloaking cost, the cloaking time, and the processing time for successful queries.

Table 1: Default system settings

Parameters	Default values
Number of queries	10,000
Valid period	Randomly chosen from [0,1440]
Privacy level K	10
δ_p	1% of min(A_{width},A_{height})
δ_q	10% of the space
δ_T	60s
Perimeter of road map	~6,000km

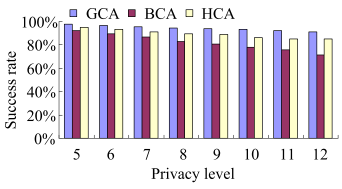
We use the well-known Thomas Brinkhoff Network-based Generator [3] to generate the moving objects in the system. The input of the generator is the road map of Oldenburg County (with perimeter around 6,000km). Our algorithms are implemented in Java and evaluated on a desktop running Windows XP SP2 with an Intel 2.0GHz CPU and 2GB main memory. A total of 10,000 moving objects are generated at the beginning of the simulation. Each object issues a continuous query, whose valid period is a random number in the range of [0, 1440]. Meanwhile, we assume a new query is not issued until the last query is successfully cloaked or expired. By default, for each query, the privacy level K is set to 10, δ_p is set to 1% of the min(A_{width},A_{height}), δ_q is set to 10% of the system service space, and δ_T is set to 60s. We summarize the default parameter settings in Table 1.

6.1 Cloaking Success Rate

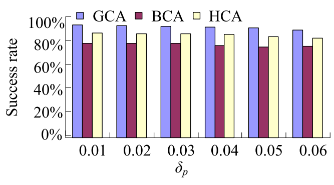
In this section, the average cloaking success rates of GCA, BCA and HCA are evaluated under various settings of K, δ_p and δ_q.

Increasing K implies the privacy requirement becomes more constrained, which indicates more queries should be covered by a cloaking set. From Figure 11(a), we can observe that the success rate decreases when K increases. Among the proposed algorithms, GCA is the best and its success rate decreases slightly with K increasing. By contrast, BCA is the worst. That is because every query in GCA finds its cloaking set greedily in the entire *query set*, while the cloaking set is found among *cluster sets* in BCA. As explained in Section 4.3, BCA omits some possible cluster merges, and hence a number of cloaking sets cannot be successfully created. For HCA, it successfully resolves the problem of BCA by introducing a post-step to refine the cluster. However, as it insists to do the cluster merges on the granularity of clusters in the pre-step, some possible merges are still missing, which harms its cloaking success rate. Therefore, the success rate of HCA is between that of BCA and that of GCA.

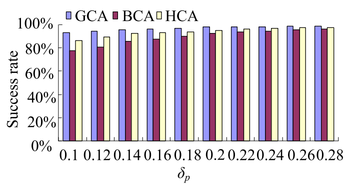
Increasing δ_p also implies a higher privacy requirement. Figure 11(b) shows that the cloaking success rate slightly decreases, with δ_p increasing. This indicates that δ_p has little impact on the success rate. On the other side, increasing δ_q implies the requirement for QoS is relaxed. Figure 11(c) shows that the cloaking success rate increases when δ_q increases. The success rate of BCA increases obviously, as



(a) Different privacy levels



(b) Different δ_p



(c) Different δ_q

Figure 11: Cloaking success rate

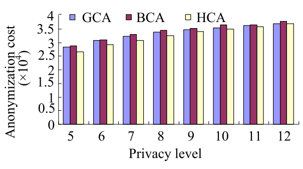
more clusters can be cloaked together. Compared with BCA, both the success rates of GCA and HCA increase slowly, indicating that δ_q has smaller impact on their success rates.

6.2 Cloaking Cost

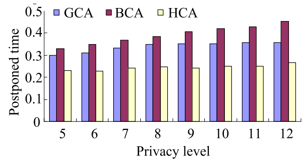
This section evaluates the average cloaking cost under different settings of privacy level K, δ_p and δ_q. The cloaking cost concerned in this paper can be divided into two parts: the anonymization cost and the postponed time.

The anonymization cost is the average perimeter of the cloaking region, which indicates the distortion of location information. As shown in Figure 12(a), for all cloaking algorithms, the average anonymization cost increases when the privacy level K increases. This is as expected because each cloaking set is required to embrace more queries so as to meet a higher privacy requirement. GCA has a lower anonymization cost than BCA, because the cloaking region is built by adding queries one by one in GCA, while in BCA, the basic incremental unit is a cluster rather than a query. HCA has the lowest anonymization cost among three algorithms. This can be explained as follows. Different from GCA, HCA finds the cloaking set for each newly arrived query from its CNN clusters and thus it can guarantee that those residing queries of the cloaking set are close to this new query. As a result, HCA would have a lower anonymization cost than GCA. In addition, though both HCA and BCA seek for the cloaking set from CNN clusters, as there is a post-step to refine the clusters in HCA, BCA has a relatively higher anoymization cost than HCA.

Postponed time evaluates the cost of postponing T_{exp} when forming cloaking sets. As the expiration time of a cloaking set is the largest T_{exp} among all residing queries, queries whose T_{exp} are prolonged are regarded as dummies after they expire. Postponed time is defined as the ratio of the prolonged time over the valid period for each successful query. As shown in Figure 12(b), the postponed time increases as K increases. The rationale behind is that, as more queries are covered by a cloaking set, the expiration time of the cloaking sets becomes larger and therefore the queries can be prolonged for a longer period. As can be observed from Figure 12(b), HCA increases very slightly, and it has the smallest postponed time among the proposed algorithms. This is due to the benefits of the cluster refinement, which enables each cluster to have more queries and thus accelerate the procedure of forming the cloaking sets. In addition, its processing time is relatively longer (see Figure 15(b)), therefore, every query would have more chances to be clustered with those queries which hold a similar expiration time. As can be observed from the figure, GCA has a smaller postponed time, when comparing with BCA. The reason is twofolded. First, same as HCA, GCA has a longer processing time, which provides more chances for those queries with similar expired time to be anonymized together. Second, for BCA, as each query becomes fixed in a cluster once it is inserted, queries within a cluster cannot be merged with the queries of other clusters, even if they have the same expired time.



(a) Anonymization cost

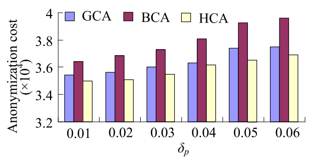


(b) Postponed time

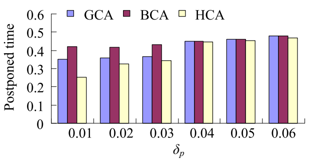
Figure 12: Different privacy levels K

Along with the increment of δ_p, the lower bound of the width/height of the cloaking boundary grows larger, which makes the newly arrived query to find farther queries for cloaking. As a result, the anonymization cost increases with δ_p increasing, just as shown in Figure 13(a). From Figure 13(b), we observe that each successful query has to be postponed for a longer period when δ_p grows larger. When δ_p is smaller (i.e., 0.01∼0.03), HCA is the best, and BCA is the worst. When δ_p grows beyond 0.03, as each query needs to be clustered with much farther queries for cloaking, their postponed time increases to a half of their valid periods. Meanwhile, their difference on the postponed time is overshadowed.

As shown in Figure 14(a) and Figure 14(b), δ_q has little effect on both of the anonymization cost and the postponed time. They are stable when δ_q increases.

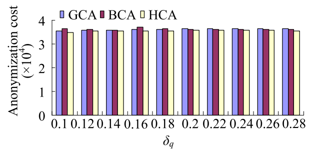


(a) Anonymization cost

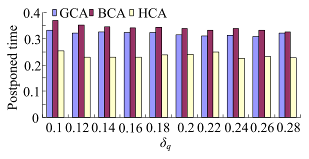


(b) Postponed time

Figure 13: Different δ_p



(a) Anonymization cost



(b) Postponed time

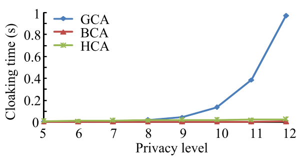
Figure 14: Different δ_q

6.3 Cloaking Time and Processing Time

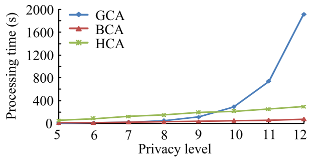
The average cloaking time and the processing time are evaluated in this section. The cloaking time of a query is the elapsed time between the moment when the query is received and the moment when it is successfully cloaked. It includes the computational time for maintaining the data structure (e.g., TPR-tree in HCA), boundary query computing, δ_q-distortion verification and δ_p-privacy verification.

From Figure 15(a), we can see that cloaking time increases as K increases. This can be explained: the bigger is the value of the privacy level K, the more is the queries of each cloaking set. As a result, a longer cloaking time is needed. Among the proposed algorithms, BCA has the shortest cloaking time. This is because BCA finds the cloaking sets from the existing continuous clusters directly, and incurs little overhead for data structure maintenance. We can also observe that, GCA performs better than HCA when K is small (e.g., < 9). The reason is that, a query can easily find its cloaking set under a small value of K, thus few queries are maintained in the service space to wait for anonymization which weakens the advantage of TPR-tree. Meanwhile, HCA has the overhead to maintain TPR-tree and clusters. Nonetheless, when K grows larger (e.g., ≥ 9), the cloaking time of GCA increases exponentially, while the advantage of TPR-tree in HCA becomes obvious. Consequently, HCA outperforms GCA after the value of K reaches 9. Note that such performance gap becomes bigger with the increment of K.

The processing time includes the cloaking time and the time waiting for cloaking. As shown in Figure 15(b), the waiting time dominates the overall processing time, and the average processing time increases when K increases. Based on the results shown in the figure, we can have the following observations: (1) BCA has the shortest processing time; (2) when K is small (e.g., < 6), GCA requires less processing time than HCA. However, when K grows larger (e.g., ≥ 6), the processing time of GCA increases exponentially, and the performance improvement of HCA over GCA becomes much larger. The rationales behind these observations are similar as described in the last paragraph.



(a) Cloaking time



(b) Processing time

Figure 15: Different privacy level K

7. CONCLUSIONS

In this paper, we investigated utility-based cloaking algorithms which protect both location privacy and query privacy for continuous queries. We observed that most of the existing location cloaking algorithms cannot effectively prevent from privacy disclosure or poor QoS for continuous queries. To address this problem, we proposed a greedy cloaking algorithm (GCA) and two incremental utility-based cloaking algorithms, called *bottom-up cloaking* (BCA) and *hybrid cloaking* (HCA). A series of experiments has been conducted to evaluate these algorithms under various system settings. Experimental results show that, GCA has the highest success rate, but suffers from a long cloaking time especially when the privacy level is high; BCA has the best efficiency, but its anonymization cost, cloaking success ratio and postponed time are relatively worse; HCA achieves the best overall performance in terms of various performance metrics.

8. ACKNOWLEDGMENTS

This research was partially supported by the grants from China 863 High-Tech Program(No:2007AA01Z155, 2009AA-011904);the Natural Science Foundation of China under grant number 60833005. Jianliang Xu's work was supported in part by the Research Grants Council, Hong Kong SAR, China (Project Nos. HKBU211206 and HKBU211307).

We would like to thank Xing Hao of Renmin University of China for her valuable help in the experiments.

9. REFERENCES

[1] B. Bamba and L. Liu. Supporting anonymous location queries in mobile environments with privacygrid. In *Proceedings of International Conference on World Wide Web (WWW)*, 2008.

[2] R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis. Nearest and reverse nearest neighbor queries for moving objects. *The VLDB Journal*, 15(3):229–249, September 2006.

[3] T. Brinkhoff. Thomas brinkhoff network-based generator of moving objects, 2008.

[4] C. Chow and M. F. Mokbel. Enabling privacy continuous queries for revealed user locations. In *Proceedings of the International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, 2007.

[5] G. Ghinita, P. Kalnis, A. Khoshgozaran, and et al. Private queries in location based services: anonymizers are not necessary. In *Proceedings of SIGMOD*, 2008.

[6] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2003.

[7] H. Hu and J. Xu. Non-exposure location anonymity. In *Proceedings of ICDE*, 2009.

[8] H. Kido, Y. Yanagisawa, and T. Satoh. Protection of location privacy using dummies for location-based services. In *Proceedings of ICPS*, 2005.

[9] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *Proceedings of SIGMOD*, 2006.

[10] Y. Li, J. Han, and J. Yang. Clustering moving objects. In *Proceedings of the tenth ACM international conference on Knowledge discovery and data mining(SIGKDD)*, pages 617–622, 2004.

[11] M. F. Mokbel, C. Y. Chow, and W. G. Aref. The new casper: query processing for location services without compromising privacy. In *Proceedings of VLDB*, 2006.

[12] L. Sweeney. K-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[13] J. Xu, X. Tang, H. Hu, and J. Du. Privacy-conscious location-based queries in mobile environments. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2009, to appear.

[14] J. Xu, W. Wang, J. Pei, and et al. Utility-based anonymization using local recoding. In *Proceedings of KDD*, 2006.

[15] T. Xu and Y. Cai. Location anonymity in continuous location based services. In *Proceedings of GIS*, 2007.