Privacy-Preserving Publication of User Locations in the Proximity of Sensitive Sites

Bharath Krishnamachari, Gabriel Ghinita, and Panos Kalnis

Department of Computer Science
National University of Singapore
{kbharath,ghinitag,kalnis}@comp.nus.edu.sg

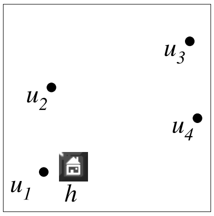
Abstract. Location-based services, such as on-line maps, obtain the exact location of numerous mobile users. This information can be published for research or commercial purposes. However, privacy may be compromised if a user is in the proximity of a sensitive site (e.g., hospital). To preserve privacy, existing methods employ the K-anonymity paradigm to hide each affected user in a group that contains at least $K - 1$ other users. Nevertheless, current solutions have the following drawbacks: *(i)* they may fail to achieve anonymity, *(ii)* they may cause excessive distortion of location data and *(iii)* they incur high computational cost.

In this paper, we define formally the attack model and discuss the conditions that guarantee privacy. Then, we propose two algorithms which employ 2-D to 1-D transformations to anonymize the locations of users in the proximity of sensitive sites. The first algorithm, called *MK*, creates anonymous groups based on the set of user locations only, and exhibits very low computational cost. The second algorithm, called *BK*, performs bichromatic clustering of both user locations and sensitive sites; BK is slower but more accurate than MK. We show experimentally that our algorithms outperform the existing methods in terms of computational cost and data distortion.

1 Introduction

The recent years have witnessed the widespread availability of positioning capabilities (e.g., GPS) in automobiles, handheld devices, etc. The emergence of novel applications based on user locations has created the potential for gathering large amounts of location data from mobile clients. Location data can also be collected from a variety of other sources. For instance, the Octopus system in Hong Kong, which employs a smart card for transportation and low-value purchases, can monitor the location where the card was used.

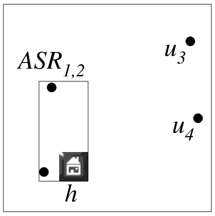
Location data can benefit a broad range of applications, such as alleviation of traffic congestion, or optimization of operations in a public transportation network. Nevertheless, Hu et al. [9] observed that publishing such data for research or planning purposes introduces serious privacy concerns. The location data can be joined with external information, such as schedules of hospital appointments, in order to reveal sensitive information about individuals. Consider the example in Figure 1a (adapted from [9]). Assume that the published location data for a specific day at $2pm$ consists of users $u_1 \dots u_4$. Furthermore, hospital h publishes the appointment schedule of Figure 1b; note that the



(a) User Locations (2pm)

Time	*Specialty*
01:00 PM	Dentistry
02:00 PM	Cardiology
04:00 PM	Surgery

(b) Hospital Appointments



(c) Anonymized Data

Fig. 1. Privacy violation in location data publishing

schedule is anonymous. Taken in isolation, neither of the two published datasets represents a privacy threat. However, by combining the two datasets, an attacker can infer that user u_1, who was the only one near the hospital at $2pm$, is consulting a cardiologist.

Previous work employed Spatial K-Anonymity (*SKA*) [10] to preserve privacy in Location-Based Services (*LBS*). SKA replaces the exact location of u with an Anonymizing Spatial Region (*ASR*), which encloses u as well as $K - 1$ other users; therefore the identification probability of u does not exceed $1/K$ (K is a user-defined parameter called anonymity degree). The process of replacing exact locations with ASRs is called *cloaking*. Several algorithms for spatial cloaking have been proposed [5,6,8,10,13]. Most of the previous work focused on hiding the association between a query sent to an LBS and the actual querying user; therefore, each time a user u sends a query, a single ASR is built around u. Our problem, on the other hand, has two characteristics that make it more difficult: *(i)* the privacy of *all* users must be preserved, as opposed to anonymizing only a single querying user, and *(ii)* an additional set of sensitive locations must be taken into account; therefore, anonymization is performed with respect to the external data that an attacker may have access to (e.g., the hospital schedule).

Continuing the earlier example, let the anonymity degree be $K = 2$. Figure 1c shows $ASR_{1,2}$ that encloses users u_1 and u_2, as well as the sensitive site h. In the published data, the exact location of u_1 and u_2 is replaced by the ASR. From the attacker's point of view, u_1 or u_2 can be anywhere inside the ASR with equal probability. Moreover, the ASR encloses h, so it is closest to h than any other user. Consequently, the attacker can only assume that either u_1 or u_2 is having a cardiologist appointment with probability at most $1/K = 1/2$. Observe that the locations of u_3 and u_4 do not need to be cloaked, because these users are further away from the sensitive site (compared to the ASR), so they would not be associated with h. Note that, while cloaking preserves privacy, it also reduces the accuracy of the published data: a researcher that studies Figure 1c cannot know exactly where the users are located. A tradeoff emerges between privacy and the amount of information that is lost in the process of cloaking. The data distortion is the sum of areas of all ASRs; this metric must be minimized.

The previous example assumed that the attacker knows the identity of u_1, therefore the association between u_1 and "cardiologist" can be performed. In practice, when publishing location data, the identity of the mobile users is removed. Nevertheless, as shown in [4], a number of methods can be employed to infer the identity of a user based on his location (e.g., through trajectory reconstruction [15]). For the rest of the paper

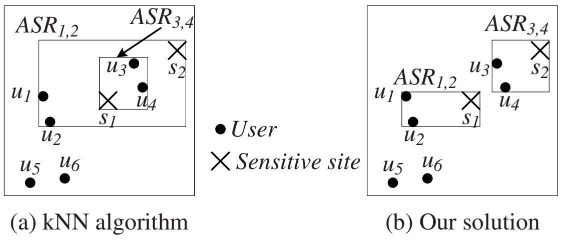


Fig. 2. The kNN approach may result to excessive data distortion

we assume that an attacker is able, in the worst case, to acquire the identity of the user associated to each location in the published data.

A naïve solution to generate ASRs in the proximity of sensitive locations is the following: For each sensitive site s_i, use a k-Nearest-Neighbor (*kNN*) algorithm to assign the K nearest users to s_i. However, the data distortion depends on the order of processing the sites. In the example of Figure 2a, s_1 is processed first resulting to $ASR_{3,4}$ since u_3 and u_4 are the nearest users; then s_2 is assigned to $ASR_{1,2}$. On the other hand, our solution (Figure 2b) assigns s_1 to $ASR_{1,2}$ and s_2 to $ASR_{3,4}$; clearly the resulting data distortion is lower. Moreover, we show in our experiments that the kNN solution is slow. Recently, Hu et al. [9] formulated the problem as a version of the *set cover* problem and proposed heuristic algorithms. However, their solution suffers from the following drawbacks: *(i)* the approach for generating and publishing ASRs does not guarantee anonymity, *(ii)* the data distortion is high and *(iii)* the computational cost is very high.

In this paper we propose efficient solutions that do not suffer from the above-mentioned drawbacks. Our methods map the 2-D user locations to 1-D space. Dimensionality reduction has been acknowledged as a suitable method to achieve privacy for both relational data [7] and in Location-based services [10]. Figure 3 shows an outline of our approach: the locations of users and sensitive sites are mapped to the 1-D domain using the Hilbert [14] space-filling curve. The Hilbert curve has good locality properties: if two points are close to each other in the 2-D space, with high probability they will also be close in the 1-D transformation. We devise two methods to generate ASRs: *(i) Monochromatic K-anonymity (MK)* is a multi-stage method: first the set of users is partitioned into groups containing K to $2K - 1$ users; the partitioning is optimal

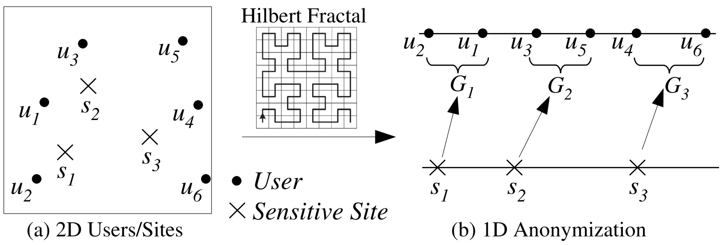


Fig. 3. Using 2-D to 1-D mapping, s_1 is assigned to $ASR_{2,1}$, s_2 to $ASR_{3,5}$, etc.

(i.e., lowest possible group extents) in the 1-D space. Then, a greedy approach is used to assign sensitive sites to user groups. MK is fast because the partitioning phase is linear to the number of users. However, since the initial partitioning is independent of the sensitive sites, the resulting data distortion is not optimal. *(ii) Bichromatic K-anonymity (BK)* is a one-stage algorithm that performs an optimal assignment (in the 1-D space) of users to sites by simultaneously clustering both users and sensitive sites (hence "bichromatic"). Although the assignment is not optimal in the 2-D space, the resulting data distortion is lower compared to MK. The tradeoff is that BK is computationally more expensive, since the search space of the solution is considerably higher than for MK. Nevertheless, we show experimentally that both MK and BK are much faster and achieve lower data distortion compared to existing methods.

The rest of the paper is organized as follows: Section 2 defines formally the problem and surveys the related work. Section 3 describes the Monochromatic K-anonymity technique, whereas Section 4 introduces Bichromatic K-anonymity. The experimental evaluation is presented in Section 5. Finally, Section 6 concludes the paper with directions for future work.

2 Background and Related Work

This section formalizes the attack on the published location data and defines the anonymi-zation problem. It also presents the related work on relational databases and Location-based services.

2.1 Problem Definition

Let U be the set of user locations and S the set of sensitive sites. Both users and sites may have arbitrary shapes and are represented by their Minimum Bounding Rectangle (*MBR*). Consider that U is published in its original form. Then, an attacker can compromise privacy by joining U and S. Formally:

Definition 1 (Attack on Location Privacy). *Given U, S and the anonymity requirement K, an attack is defined as the result of the following spatial join:*

```
SELECT user.id, site.id
FROM U as user, S as site
WHERE distance(user.mbr, site.mbr) =
      SELECT MIN(distance(U.mbr, S.mbr))
      FROM U, S
      WHERE S.id = site.id
```

An attack is successful iff the probability of distinguishing a particular user u in any of the resulting tuples of the above query is larger than $1/K$.

In the example of Figure 1a, the spatial join will output the tuple $\langle u_1, h\rangle$, since u_1 is closest to h than any other user. Therefore the attacker can infer that u_1 has a cardiology appointment.

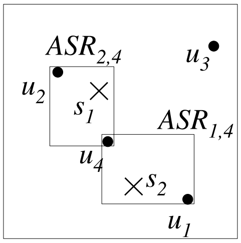


Fig. 4. User sharing violates privacy

To achieve anonymity, each sensitive site s is associated with an *anonymizing set*, denoted by $\mathcal{M}(s)$, of at least K users who are indistinguishable from each other. Instead of publishing the exact user locations, we publish the ASR, which is the MBR that encloses s and the users in $\mathcal{M}(s)$. Formally:

Definition 2 (Privacy-Preserving Location Publishing Format). *A privacy-preserving publication of U with respect to S is a mapping $\mathcal{M} : S \rightarrow 2^U$ (2^U is the set of all possible anonymizing sets). The published format consists of a collection of ASRs, one for each $s \in S$, where $ASR(s) = MBR(\{s\} \cup \mathcal{M}(s))$.*

Based on the attack model and publication format, we define below the K-anonymity condition for our problem:

Definition 3 (K-anonymous Location Publishing). *A privacy-preserving mapping $\mathcal{M} : S \rightarrow 2^U$ is K*-anonymous *iff* (i) *$\forall s \in S$, $|\mathcal{M}(s)| \geq K$ and* (ii) *$\forall s_1, s_2 \in S$, $\mathcal{M}(s_1) \cap \mathcal{M}(s_2) = \emptyset$.*

Condition *(i)* is imposed by the indistinguishability requirement, whereas condition *(ii)* specifies that the anonymizing sets of different sites should be disjoint. To demonstrate the need for the second condition, consider the example in Figure 4, where $K = 2$. S consists of sites s_1 and s_2, but the anonymizing sets of these sites overlap in user u_4. Therefore, only three distinct users are included in $\mathcal{M}(s_1) \cup \mathcal{M}(s_2)$. An attacker can infer that any of u_1, u_2 or u_4 was present at a sensitive site (either s_1 or s_2) with probability $2/3 = 0.66 > 1/K = 0.5$; hence, privacy is compromised.

Returning to the example of Figure 1c, the K-anonymity conditions are satisfied, since $|\mathcal{M}(h)| = |\{u_1, u_2\}| = 2$ and $\mathcal{M}(h)$ does not share users with any other site. This can be verified by executing the query of Definition 1. The exact locations of u_1 and u_2 have been replaced by the ASR, whose distance to h is 0. Therefore the query returns two tuples: $\langle u_1, h\rangle$ and $\langle u_2, h\rangle$; hence the probability of associating u_1 or u_2 with the cardiology appointment is at most $1/2$. Observe that u_3 and u_4 are not included in the query results, since their distance to h is larger than that of the ASR. Therefore, we can publish the exact locations of u_3 and u_4 without compromising privacy.

Besides providing privacy, the distortion (also called *generalization cost*) of the published data must be minimized. Similar to the related work in Location-based services [5,6,8,10,13], we measure the generalization cost by the sum of the areas of the published ASRs. Formally:

Definition 4 (Generalization Cost). *Given a set U of user locations, a set S of sensitive sites, and a privacy-preserving mapping $\mathcal{M} : S \rightarrow 2^U$, the generalization cost for site s is:*

$$GC(s) = Area(MBR(\mathcal{M}(s) \cup \{s\})) \tag{1}$$

The overall (global) generalization cost of the entire mapping is:

$$GGC(\mathcal{M}) = \sum_{s \in S} GC(s) \tag{2}$$

Recall that for some users we publish their exact locations (e.g., u_3 and u_4 in Figure 1c); no generalization cost is incurred for such users. Our problem is formally defined as:

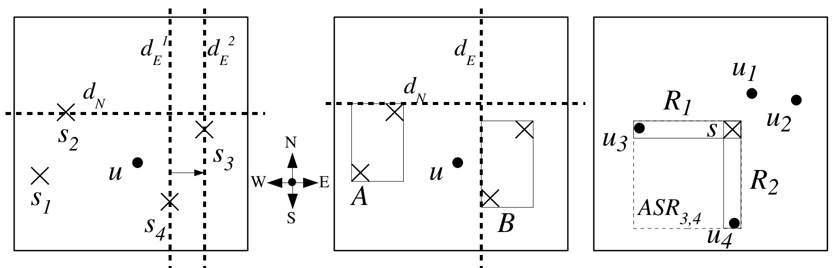
Problem 1 (Optimal K-Anonymous Location Data Publication). *Given U, S and K, determine a K-anonymous mapping $\mathcal{M} : S \rightarrow 2^U$ such that the generalization cost $GGC(\mathcal{M})$ is minimized.*

2.2 *K*-Anonymity in Relational Databases

K-anonymity [16,17] was initially proposed in relational databases for privacy preserving publishing of detailed data (or *microdata*), such as hospital records. Although identifying attributes (e.g., name) are removed, microdata contains *quasi-identifying attributes (QID)* (e.g., $\langle Age, Zipcode \rangle$) that can be joined with external information, such as voting registration lists, to expose the identity of individual records. To address this threat, K-anonymity requires that each record must be indistinguishable from at least $K - 1$ other records, with respect to the QID. Two techniques are commonly used to achieve *K*-anonymity: *suppression*, where some of the attributes or tuples are removed, and *generalization*, which involves replacing specific values (e.g., phone number) with more general ones (e.g., only area code). Both methods lead to information loss. Algorithms for anonymizing an entire relation are discussed in [2,11]. Xiao and Tao [18] consider the case where each individual requires a different degree of anonymity, whereas Aggarwal [1] shows that anonymizing a high-dimensional relation leads to unacceptable loss of information due to the dimensionality curse. Machanavajjhala et al. [12] propose ℓ-diversity, an anonymization method that provides diversity among the sensitive attribute values of each anonymized group. Ghinita et al. [7] employ multi-dimensional to 1-D transformations to solve efficiently the *K*-anonymity and ℓ-diversity problems.

2.3 *K*-Anonymity for Location Data

Most related work in the area of location K-anonymity focuses on *query* privacy in Location-based Services (*LBS*). Users issue queries such as “find the closest hospital to my current location”. Typically, there is a trusted Anonymizer Service (*AN*) between the users and the LBS. The users constantly update their location with AN. Queries are also sent through AN, which removes the user id and constructs an ASR that contains the querying user as well as at least $K - 1$ additional users. The AN forwards the ASR to the LBS, which computes the answer based on the ASR, instead of the exact user location. The result is also routed back to the querying user through the AN. In [8],



(a) Local Algorithm (b) R-Tree-based Pruning (c) "2-by-2" Publishing

Fig. 5. *Local* Algorithm

the anonymizer employs a quad-tree to index user locations. Given a query from u, the corresponding ASR is the lowest-level quadrant that contains u as well as $K-1$ other users. In [13] a similar structure is used, but two neighboring quadrants are allowed to form an ASR, before ascending one level up in the quad-tree. In [6] queries from multiple users form a graph. The graph is searched for cliques (i.e., queries from near-by users), which are used to form the ASR. Kalnis et al. [10] identify the *reciprocity* property, a sufficient condition to guarantee anonymity. To enforce reciprocity, the users are split into disjoint buckets based on their 1-D Hilbert ordering; the same transformation is used in our work. The previous algorithms generate a single ASR independently for each query. This approach is not applicable to our problem, since we must publish an anonymized version of the entire dataset U; furthermore, anonymization depends on the set of sensitive sites S.

Location publishing in the proximity of sensitive sites was first discussed by Hu et al. [9]. They formulated the problem as a version of the *set cover* problem and proposed a heuristic algorithm called *Local* (see Figure 5). *Local* is a user-centric method: for each user $u \in U$ (for simplicity the example shows only one user) the location of u is incrementally enlarged to include sensitive sites in its bounding box. *Local* consists of four nested loops, corresponding to four directions originating at u (North, East, South, West), and each loop advances a plane-sweep line in its direction. In Figure 5a, the North sweeping line d_N is fixed, and the East line is advanced from d_E^1 to d_E^2 to cover sites s_4 and s_3, respectively. Out of all combinations along the four directions, the bounding box with the optimal *coverage* (measured as the area of the bounding box divided by the number of enclosed sites) is retained as the *candidate box* $\Omega(u)$. The candidate boxes are determined for all $u \in U$, and user u_0 with the lowest coverage is output, at which point a *coverage counter* of all sites enclosed by $\Omega(u_0)$ is increased. If the counter of a site s reaches K, s is removed from S, the candidate boxes of all other users that enclose s are updated, and the algorithm continues for the remaining sites, until all sites are covered at least K times. The complexity of *Local* is $O(|S|^4 \cdot (|S|+|U|))$, which is very high. [9] proposes an optimization based on the R-Tree spatial index (see Figure 5b). Instead of performing the plane-sweep with respect to individual sites, the algorithm considers the nodes of the R-Tree. Each node represents a "super-site", which is considered to be situated at the point inside the node that is closest to u, and has a weight equal to the number of sites rooted in the subtree of that node. It is shown that by using the super-site concept, a lower bound of the actual coverage is obtained, and the search space of the solution is reduced.

We show in Section 5 that the execution time (even with the R-tree optimization) and generalization cost of $Local$ are very high. More importantly, $Local$ allows the anonymizing sets of distinct sensitive sites to share users. Therefore, it does not guarantee privacy (recall condition *(ii)* in Definition 3). Furthermore, the authors of [9] propose a publication format, further referred to as *"2-by-2"*, that discloses a collection of MBRs for each site s: each MBR encloses s and a *single* user in $\mathcal{M}(s)$. However, this format discloses exact user locations, since each published MBR only contains two locations, and one of them (i.e., s) is known to the attacker. Figure 5c shows an example: Local chooses users u_3 and u_4 as part of $\mathcal{M}(s)$, because the two rectangles R_1 and R_2, which are very skewed, have small areas (hence, low generalization cost). An attacker can infer that the users are situated at the opposite extremities of R_1, respectively R_2, from s. This is similar to publishing the exact locations of all users, therefore privacy is compromised. Should we choose a secure publishing format like the one in Definition 2, i.e. $ASR_{3,4}$ in the example, the resulting area is very large. We will investigate this issue further in Section 5.

3 Monochromatic K-Anonymity (MK)

In this section, we present Monochromatic K-Anonymity (*MK*). MK is a multi-stage algorithm: In the first stage, it partitions the set U into groups with K to $2K-1$ users each. In the subsequent stages, it uses a greedy approach to assign user groups to each site in S. The first stage of MK employs the *1DAnon* algorithm, which was used in [7] to partition relational data with 1-D quasi-identifiers. Below, we briefly explain *1DAnon*.

1DAnon takes as input the set U of user locations sorted according to their 1-D Hilbert values. We use u to denote a user, as well as his coordinate in the 1-D space. Furthermore, we denote by $|u_i - u_j|$ the 1-D distance between users u_i and u_j. Given a group of users $G = \{u_{begin}, \ldots, u_{end}\}$, where u_{begin} and u_{end} represent respectively the user with the minimum and maximum 1-D coordinate in G, we denote the *extent* of G in the 1-D space as: $1D\_Ext(G) = |u_{end} - u_{begin}|$. We refer to $begin$ and end as the *boundaries* of G. *1DAnon* finds the optimal K-anonymous partitioning $\mathcal{U} = \{G_1, \ldots, G_{|\mathcal{U}|}\}$ of U, such that the

$$1D\_Cost(\mathcal{U}) = \sum_{G \in \mathcal{U}} 1D\_Ext(G) \tag{3}$$

is minimized. Note that Eq. (3) is the one-dimensional equivalent of the GGC metric from Eq. (2). To find the optimal anonymous partitioning of U, *1DAnon* applies a dynamic programming recursive formulation which determines the best grouping for each prefix $\{u_1, \ldots, u_i\}$ (where $K \leq i \leq |U|$) of the user sequence. *1DAnon* returns a set of K-anonymous groups, each with size bounded between K and $2K-1$. The computation cost of *1DAnon* is $O(K \cdot |U|)$, hence linear to the number of users. In our MK algorithm, we vary *1DAnon* slightly: Instead of $1D\_Cost$, we use the GGC metric. The user partitioning is not optimal in the 2-D space, but due to the good locality properties of the Hilbert ordering, the results are adequate in practice.

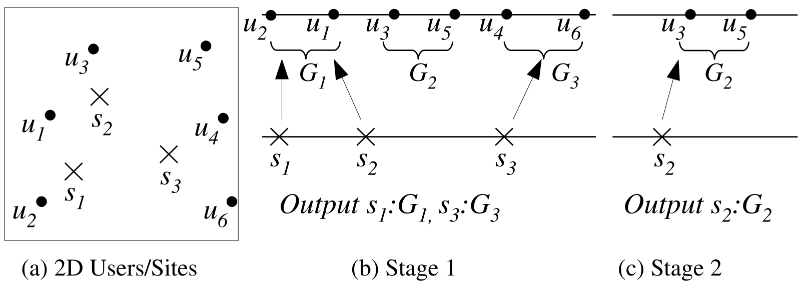
```
Monochromatic K-anonymity (MK)
Input: sets $U[1 \dots n]$ and $S[1 \dots m]$ sorted in ascending order of 1D Hilbert values
1.  $\mathcal{U}$ = $1DAnon(U, K)$
2.  while $|S| > 0$
3.    foreach $s \in S$
        // assign $s$ to a user group s.t. the resulting area is minimized
4.      $AS(s) = G_i$ s.t. $\forall j \neq i$, $Area(MBR(G_i \cup \{s\})) < Area(MBR(G_j \cup \{s\}))$
5.    foreach $G \in \mathcal{U}$
6.      if $\exists s$ s.t. $AS(s) = G$ then
          // choose the site that minimizes the resulting area
7.        choose $s_0$ s.t. $AS(s_0) = G$ and $\forall s \in S | s \neq s_0 \wedge AS(s) = G$,
                 $Area(MBR(G \cup \{s_0\})) < Area(MBR(G \cup \{s\}))\}$
8.        output $MBR(G \cup \{s_0\})$
9.        $U = U \setminus G$
10.       $S = S \setminus \{s_0\}$
```

Fig. 6. Monochromatic K-Anonymity Pseudocode

Figure 6 shows the pseudocode of MK: the input consists of sets U and S, sorted in the 1-D Hilbert order of the locations of users and sensitive sites, respectively. The cardinalities of the two sets are denoted as $n = |U|$ and $m = |S|$. Initially, MK invokes *1DAnon* (line 1) and obtains $\mathcal{U}$, which is the partitioning of U into K-anonymous groups. Subsequently, MK assigns the sensitive sites to groups of $\mathcal{U}$. At each stage, each site $s \in S$ is assigned to the user group G that minimizes the area of $MBR(G \cup \{s\})$ (lines 3-4). We say that G is the anonymizing set of s, i.e. $AS(s) = G$. Note that, multiple sites can be assigned to the same group, whereas some groups may not be assigned any site. Since sites cannot share users (condition *(ii)* in Definition 3), collisions are solved (line 7) by choosing for each group the site that minimizes the area of $MBR(G \cup \{s\})$ (in case of ties, a random site is chosen). For each assigned site, we output (line 8) the MBR that encloses s_0 and its corresponding anonymizing set $\mathcal{M}(s_0) \equiv G$. U and S are updated by eliminating the users and sites that have been output (lines 9-10). If there still exist unassigned sites (line 2), the algorithm starts a new stage with the remaining users and sites. Since at most $2K - 1$ users belong to each user group, the algorithm is guaranteed to terminate if the inputs satisfy the condition $n \geq (2K - 1) \cdot m$.

In the worst case, at each stage all sites are assigned to the same user group, and the number of required stages is m. Each stage takes $O(m \cdot n)$ to find the closest group for each site. Hence, the complexity of MK is $O(K \cdot n + m^2 \cdot n)$; the first term corresponds to *1DAnon*. In practice, the number of stages is significantly smaller than m; in Section 5 we show that MK is very fast.

Figure 7 illustrates an example of applying MK for $U = \{u_1 \dots u_6\}$ and $S = \{s_1 \dots s_3\}$. Initially (Figure 7b), the *1DAnon* algorithm is executed, resulting in anonymous groups $G_1 \dots G_3$. Then, sites s_1 and s_2 are assigned to G_1, whereas s_3 is assigned to G_3. Since the enclosing area of s_2 and G_1 is larger than that of s_1 and G_1, MK outputs s_1 with anonymizing set G_1, and s_3 with G_3. In the next stage (Figure 7c), the remaining users and sites are $\{u_3, u_5\}$ and s_2, which are output together. Note that,



u_3 u_5 s_2 u_1 u_4 s_3 s_1 u_2 u_6
u_2 u_1 u_3 u_5 u_4 u_6
G_1 G_2 G_3
s_1 s_2 s_3
Output s_1:G_1, s_3:G_3
u_3 u_5 G_2 s_2
Output s_2:G_2
(a) 2D Users/Sites
(b) Stage 1
(c) Stage 2

Fig. 7. Example of MK

it is possible for MK to terminate, even though some users do not have any site assigned to them. For those users we publish their exact location, since their privacy is not threatened (refer to Definition 3).

4 Bichromatic K-Anonymity (BK)

This section introduces the Bichromatic K-anonymity (*BK*) algorithm. BK also uses the 1-D Hilbert transformation for U and S. However, the process of creating anonymizing sets considers simultaneously U and S (as opposed to MK, which partitions U independently). As a result, BK achieves lower generalization cost.

Before presenting BK, we will study a restriction of the problem to the 1-D space. We seek to find an optimal K-anonymous mapping $\mathcal{M}$ that assigns sites to user groups, such that the 1-D cost is minimized. In the 1-D domain the generalization cost of a mapping is:

$$1D\_Cost(\mathcal{M}) = \sum_{s \in S} 1D\_Ext(\{s\} \cup \mathcal{M}(s)) \tag{4}$$

In Section 4.1 we identify three properties of an optimal 1-D mapping: *(i)* each anonymizing set contains *exactly* K users, *(ii)* each anonymizing set consists of users that are consecutive in the 1-D domain, and *(iii)* the extents of any two anonymizing sets do not overlap in the 1-D domain. Based on these properties, in Section 4.2 we present the BK algorithm, which employs dynamic programming to solve the problem in the 2-D space. Although the 2-D solution is not optimal, due to the good locality properties of the Hilbert ordering, BK achieves very low generalization cost in practice.

4.1 Properties of an Optimal 1-D Mapping

The following theorem states that there exists an optimal 1-D mapping where the anonymizing set of each site contains exactly K users.

Theorem 1. *Consider a set of user locations U and a set of sensitive sites S. Then, there exists an optimal mapping $\mathcal{M} : S \to 2^U$ such that, $\forall s \in S$, $|\mathcal{M}(s)| = K$.*

Proof. Let $\mathcal{M}'$ be the optimal mapping for U and S, and assume that $\exists s_0 \in S$ such that $G = \mathcal{M}'(s_0)$ and $|G| > K$. Let G' be the anonymizing set obtained by retaining only

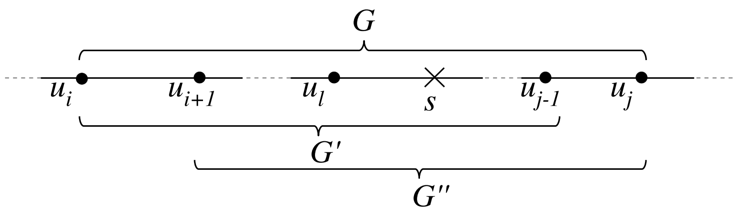


Fig. 8. Anonymizing sets consist of consecutive users in the 1-D sequence

K users from G. Define mapping $\mathcal{M}$ such that $\mathcal{M}(s_0) = G'$, and $\forall s \neq s_0, \mathcal{M}(s) = \mathcal{M}'(s)$. Then the K-anonymity condition for $\mathcal{M}$ is satisfied, according to Definition 3. Furthermore, since $G' \subset G$, we have that $1D\_Ext(G') \leq 1D\_Ext(G)$, hence the generalization cost of $\mathcal{M}$ does not exceed that of $\mathcal{M}'$. By applying the same reasoning for all groups with size larger than K, we obtain an optimal mapping $\mathcal{M}$ such that $\forall s \in S$, $|\mathcal{M}(s)| = K$. □

We further show that the anonymizing set of every site s consists of users that are consecutive in the user sequence. Formally:

Theorem 2. *There exists an optimal mapping $\mathcal{M}$ such that $\forall s \in S$, if $u_i, u_j \in \mathcal{M}(s)$ and $i < j$, then $\forall u_l$ with $i < l < j$, $u_l \in \mathcal{M}(s)$.*

Proof. Assume optimal mapping $\mathcal{M}'$ and that $\exists s \in S$ such that $u_i, u_j \in \mathcal{M}'(s)$, $i < j$, and $\exists u_l$, $i < l < j$, such that $u_l \notin \mathcal{M}'(s)$. This situation is depicted in Figure 8. Then, we can replace $\mathcal{M}'(s)$ with either $G' = \mathcal{M}'(s) \diagdown \{u_i\} \cup \{u_l\}$ or $G'' = \mathcal{M}'(s) \diagdown \{u_j\} \cup \{u_l\}$. The privacy condition is still satisfied, since $|G'| = |G''| = K$; furthermore, $1D\_Ext(G') \leq 1D\_Ext(G)$ and $1D\_Ext(G'') \leq 1D\_Ext(G)$. Therefore, we obtain a new K-anonymous mapping $\mathcal{M}$ with generalization cost not exceeding that of $\mathcal{M}'$, hence optimal. □

We also prove that there exists an optimal 1-D mapping, where the extents of the anonymizing sets do not overlap. Formally:

Theorem 3. *There exists an optimal mapping $\mathcal{M}$ such that, $\forall s_1, s_2 \in S$, and $G_1 = \mathcal{M}(s_1)$, $G_2 = \mathcal{M}(s_2)$, the 1-D extents of G_1 and G_2 do not overlap.*

Proof. Denote by $\mathcal{M}'$ the optimal mapping for U and S, and assume that $\exists s_1, s_2 \in S$, $G_1 = \mathcal{M}'(s_1)$ and $G_2 = \mathcal{M}'(s_2)$, such that G_1 and G_2 overlap in their 1-D extents. Let u_{i_1}, u_{i_2} be the start boundaries and u_{j_1}, u_{j_2} be the end boundaries of groups G_1 and G_2, respectively. Without loss of generality, consider that $u_{i_2} < u_{j_1}$, a situation depicted in Figure 9. We build anonymizing groups $G_1' = G_1 \cup \{u_{i_2}\} \diagdown \{u_{j_1}\}$ and $G_2' = G_2 \cup \{u_{j_1}\} \diagdown \{u_{i_2}\}$, that is, we swap the end user of G_1 with the start user of G_2. Since $|u_{i_2} - u_{i_1}| + |u_{j_2} - u_{j_1}| \leq |u_{j_1} - u_{i_1}| + |u_{j_2} - u_{i_2}|$, the generalization cost of the mapping is not enlarged. Furthermore, sets G_1' and G_2' have the same cardinality as G_1 and G_2, hence the privacy requirement is satisfied. Therefore, the new mapping $\mathcal{M}$ obtained by replacing G_1, G_2 with G_1', G_2' is optimal. By applying the same reasoning for every pair of overlapping groups, the theorem is proved. □

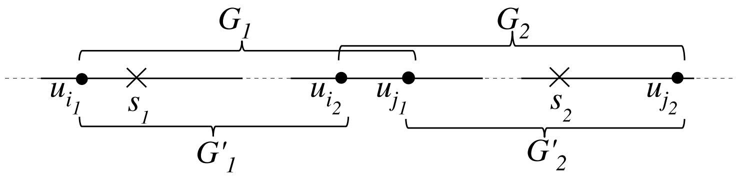


Fig. 9. Anonymizing sets do not overlap in their 1-D extent

4.2 The BK Algorithm in the 2-D Domain

BK is a dynamic programming algorithm, which is based on the properties of the 1-D ordering. BK finds the best mapping $\mathcal{M}$ by minimizing the 2-D GGC metric from Eq. (2). Recall that BK is not optimal in the 2-D domain.

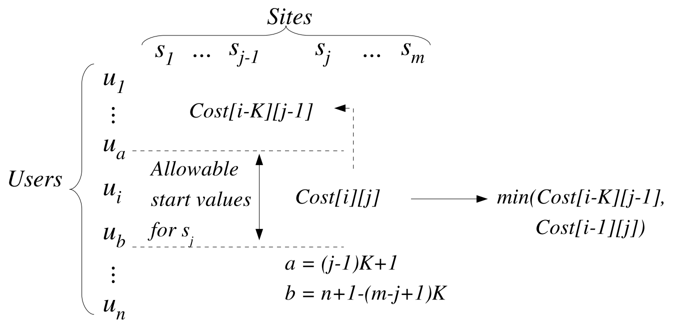
Let $AS(s)$ be the anonymizing set of site s. Each AS has cardinality K (Theorem 1) and consists of consecutive users in the 1-D order (Theorem 2). Therefore, we can uniquely identify a particular AS by its start boundary i (i.e., the group starting at i consists of users $u_i \dots u_{i+K-1}$).

BK determines recursively the optimal mapping for each sub-problem corresponding to prefixes $U_i = \{u_1 \dots u_{i+K-1}\}$ of U and $S_j = \{s_1 \dots s_j\}$ of S. Intuitively, U_i contains all users that may be part of anonymizing sets starting at boundary *at most* i. BK tabulates the values of a cost matrix $Cost[1 \dots n][1 \dots m]$, where element $Cost[i][j]$ contains the optimal solution to the sub-problem with inputs U_i and S_j.

According to Theorem 3, the users in $AS(s_j)$ must be after those in $AS(s_{j-1})$; therefore, not all start boundaries are acceptable for a given j. Let $a(j)$ be the minimum and $b(j)$ the maximum allowable start boundary for $AS(s_j)$. There must be enough users before $AS(s_j)$ to build AS for sites $s_1 \dots s_{j-1}$. Similarly, sufficient users must remain after $AS(s_j)$, to form AS for $s_{j+1} \dots s_m$. Formally:

$$a(j) = (j-1) \cdot K + 1, \; b(j) = n + 1 - (m - j + 1) \cdot K \tag{5}$$

Figure 10 illustrates the $Cost$ matrix and the possible choices of the start boundary for $AS(j)$.



Sites
s_1 ... s_{j-1} s_j ... s_m
Users
u_1
u_a
u_i
u_b
u_n
Cost[i-K][j-1]
Allowable start values for s_j
Cost[i][j]
min(Cost[i-K][j-1], Cost[i-1][j])
a = (j-1)K+1
b = n+1-(m-j+1)K

Fig. 10. BK: $Cost$ matrix tabulation

```
Bichromatic K-anonymity (BK)
Input: sets $U[1 \ldots n]$ and $S[1 \ldots m]$ sorted in ascending order of 1D Hilbert values
0.  $min\_val = \infty$, $best\_start\_value = -1$
    /* populate first column */
1.  for $i = 1$ to $n - m \cdot K + 1$ do /* for every allowed start boundary of $AS(s_1)$ */
2.      if $GC(u_i, \ldots, u_{i+K-1}, s_1) < min\_val$ then
3.          $min\_val = GC(u_i, \ldots, u_{i+K-1}, s_1)$
4.          $best\_start\_value = i$
5.      $solution[i][1] = best\_start\_value$
6.      $Cost[i][1] = min\_val$
    /* populate remaining columns */
7.  for $j = 2$ to $m$ do
8.      $min\_val = \infty$, $best\_start\_value = -1$
9.      for $i = (j-1)K + 1$ to $n + 1 - (m - j + 1)K$ do /*for every start boundary of $AS(s_j)$*/
10.         if $(Cost[i-K][j-1] + GC(u_i, \ldots, u_{i+K-1}, s_j)) < min\_val$ then
11.             $min\_val = Cost[i-K][j-1] + GC(u_i, \ldots, u_{i+K-1}, s_j)$
12.             $best\_start\_value = i$
13.         $solution[i][j] = best\_start\_value$
14.         $Cost[i][j] = min\_value$
    /* output solution */
15. $group\_start = solution[n - K + 1][m]$
16. output $\mathcal{M}(s_m) \equiv AS(s_m) = (u_{group\_start} \ldots u_{group\_start+K-1})$
17. for $j = m - 1$ downto $1$ do
18.     $group\_start = solution[group\_start - K][j]$
19.     output $\mathcal{M}(s_j) \equiv AS(s_j) = (u_{group\_start} \ldots u_{group\_start+K-1})$
```

Fig. 11. Bichromatic K-Anonymity Pseudocode

Note that some users may not be included in any AS, hence there may be "gaps" left in the user sequence when forming an AS. As mentioned earlier, $Cost[i][j]$ stores the best cost of the solution for sub-problem U_i, S_j. According to Eq. (5), the AS of the last site in S_j can start at any value between $a(j)$ and i. Hence, $Cost[i][j]$ contains the minimum cost over all choices of start boundary i', $a(j) \leq i' \leq i$. The $Cost$ value is recursively determined as:

$$Cost[i][j] = \min\{Cost[i-1][j],\ Cost[i-K][j-1] + GC(u_i, \ldots, u_{i+K-1}, s_j)\} \quad (6)$$

If the first element of the *min* function is smaller, it signifies that choosing to start $AS(j)$ at i is more costly than if we start it at $i-1$, or earlier. Hence, i should not be the start of $AS(s_j)$. Otherwise, $AS(s_j)$ should begin at i, and the value of $Cost[i][j]$ is updated as the sum of the immediate generalization cost associated to the area enclosing $\{s_j\} \cup \{u_i \ldots u_{i+K-1}\}$, and the recursive component $Cost[i-K][j-1]$. The latter corresponds to the best cost obtained for the sub-problem U_{i-K}, S_{j-1} (the $i-K$ is dictated by the requirement that $AS(s_{j-1})$ must end before i, hence must have start boundary at most $i-K$).

Figure 11 shows the BK pseudocode. The values in the first column of the matrix (i.e., $Cost[*][1]$) are determined directly (lines 1-6) by computing all possible anonymizing sets associated to sensitive site s_1. Formally:

$$Cost[i][1] = \min_{1 \leq i' \leq i} GC(u_{i'}, \ldots, u_{i'+K-1}, s_1),\ a(1) \leq i \leq b(1) \quad (7)$$

In addition to the minimum cost value, we also need to retain the i' value that minimizes the above cost, to reconstruct the solution once the tabulation is completed. For this purpose we use an additional table $solution[1 \ldots n][1 \ldots m]$, which contains at element $solution[i][j]$ the start boundary of $AS(s_j)$ of the best solution to sub-problem U_i, S_j.

The main loop (lines 7-14) tabulates the contents of $Cost$ and $solution$ in increasing value of j (i.e., by columns), and in increasing value of i within each column, based on the best solution obtained previously for column $j-1$. Finally, the mapping $\mathcal{M}(s_j), 1 \leq j \leq m$, is obtained in lines 15-19 with the help of the $solution$ table. The cost of the best mapping corresponds to the minimum value in the final column m (recall that each entry in column j of $Cost$ stores the *accumulated* cost of the solution to subproblem U_i, S_j). Formally:

$$BestCost = \min_{(m-1)\cdot K < i \leq n-K+1} Cost[i][m] \tag{8}$$

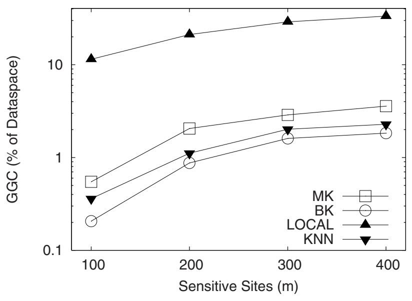
From Eq. (5), it results that the number of actual entries in each column j (i.e., the number of allowable i values) is $b(j) - a(j) + 1 = (n + 1 - m \cdot K)$. The total number of tabulated entries becomes $(n + 1 - m \cdot K) \cdot m \equiv O(m \cdot n)$. However, the tabulation proceeds column-by-column, and only the last column of $Cost$ needs to be retained at any time. Hence, the space complexity of storing $Cost$ is $O(n)$. Still, we need to store the entire $solution$ table. Nevertheless, only a constant $O(n)$ fraction (i.e., the current column) must be stored in main memory, while the rest can be saved to secondary memory and read one more time when the output is performed.

In terms of computational cost, BK needs to tabulate $O(n \cdot m)$ entries of $Cost$, and each entry requires $O(K)$ computation for determining the base-case cost of GC in Eq. (6) (line 10). The total cost is $O(m \cdot n \cdot K)$. Although this is asymptotically lower than MK, we show in Section 5 that BK is more expensive in practice, since the actual number of stages in MK is much smaller than the worst case analysis. Nevertheless, the generalization cost incurred by BK is considerably lower.

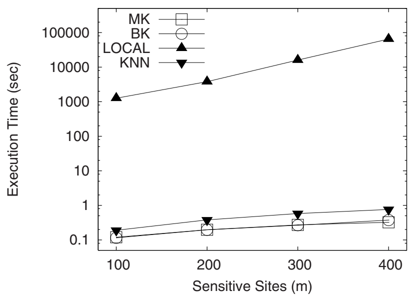
5 Experimental Evaluation

We implemented C++ prototypes of the proposed MK and BK algorithms, as well as the $Local$ technique from [9]. We also implemented a benchmark method based on nearest neighbor search, referred to as KNN. KNN picks sensitive sites in random order, and for each $s \in S$ and a given K, it includes in $\mathcal{M}(s)$ the K nearest users of s. Those users are then eliminated from U, to ensure that users are not shared among sites. For efficiency, in the KNN method we index the users with an in-memory R*-Tree [3].

Our experiments were run on a P4 3.0 Ghz machine with 1GB of RAM and Linux OS. We measured the execution time and the generalization cost GGC. GGC is expressed as the percentage of the sum of areas of all generalized locations, over the area of the entire dataspace (intuitively, this measures how much of the dataspace area is covered by the published locations). Formally:



(a) Generalization Cost



(b) Execution Time

Fig. 12. Variable m, $K = 5$, 10000 Users

$$GGC(\mathcal{M}) = 100 \cdot \frac{\sum_{s \in S} Area(MBR(\mathcal{M}(s) \cup \{s\}))}{DomainArea} \% \tag{9}$$

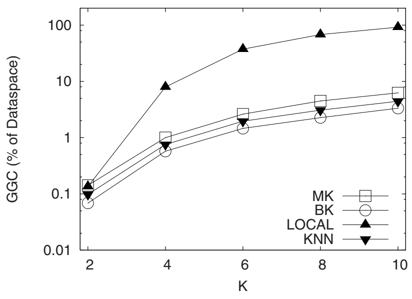
We used the NA[1] real dataset, consisting of $569,120$ locations on the North-American continent. We generated U and S sets of various sizes through random sampling from NA. In Section 5.1 we compare all algorithms for small input sizes, because of the very high overhead of $Local$, whereas in Section 5.2 we evaluate MK, BK and KNN for large inputs.

5.1 Comparison Against *Local*

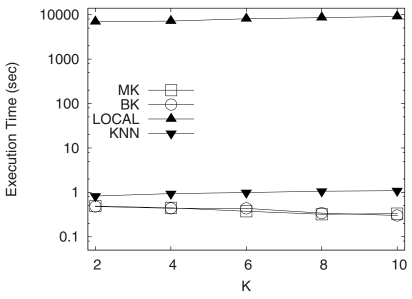
In this experiment we set the number of users to $10,000$, $K = 5$, and vary the number of sensitive sites m. Figure 12a shows that even for such a small value of K, the GGC incurred by $Local$ is one order of magnitude worse than that of other methods (roughly 10% of the dataspace). As discussed in Section 2.3, $Local$ tends to include in $\mathcal{M}(s)$ users that are very close to site s in one of the x or y coordinates, but they may be far away in actual distance. Therefore, the resulting MBR is very large. We also measured GGC using the publication method proposed in [9] (recall from Section 2 that this format has serious privacy drawbacks). For $m = 200$, for instance, $Local$ achieves a GGC value of 0.25, compared to 0.78 for KNN. However, $Local$ performs poorly when a secure publishing format is used. Among the other methods, BK obtains the best GGC. In terms of execution time, Figure 12b shows that $Local$ is several orders of magnitude slower than the other techniques. For 400 sites the absolute value is 18 hours. The results do no utilize the R*-Tree-based optimization described in [9]. However, a preliminary implementation that included that improvement did not show significant gains. Among the other algorithms BK and MK are very fast, outperforming KNN.

Figure 13 presents the results for variable K; $n = 10,000$ users and $m = 200$ sites. The only value for which $Local$ achieves low GGC is $K = 2$. For this value, it is likely that a site s includes in its $\mathcal{M}(s)$ two users with close-by x or y coordinates, resulting

[1] http://www.rtreeportal.org

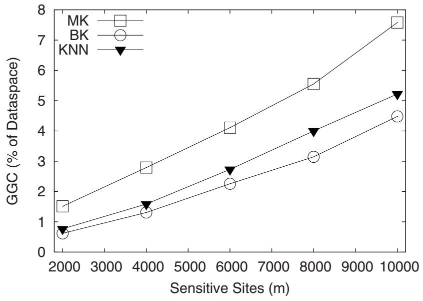


(a) Generalization Cost

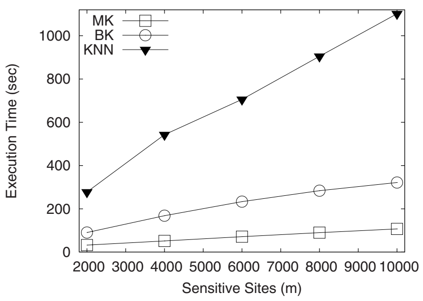


(b) Execution Time

Fig. 13. Variable K, 200 Sensitive Sites, 10000 Users



(a) Generalization Cost



(b) Execution Time

Fig. 14. Variable m, $K = 20$, 569120 Users

in an MBR with small area. As K increases, the resulting MBR becomes less skewed, and its area grows considerably.

5.2 Comparison of MK and BK Versus KNN

Below, we compare MK, BK and KNN for large input sizes that are relevant for practical applications (we exclude Local due to its excessive running time). Unless otherwise specified, U consists of the entire NA dataset (i.e., 569K users). Figure 14 compares the three algorithms for variable m and $K = 20$. There is a clear tradeoff between MK and BK: GGC is up to 2 times lower for BK compared to MK, but MK is up to 8 times faster. The execution time of MK is 42 seconds for the largest input. KNN is worse than BK in terms of GGC and it is also much slower.

In Figure 15, we vary K for $m = 4,000$ sensitive sites. BK maintains its advantage over KNN in terms of GGC, while being up to ten times faster. MK is the fastest method. Observe that the execution time of BK decreases with K, because the number of tabulated entries in the dynamic programming formulation is $(n + 1 - m \cdot K) \cdot m$. Intuitively, less candidate start boundaries need to be considered as K increases. For MK, there are two contrary effects as K increases: the initial cost of *1DAnon* is linear

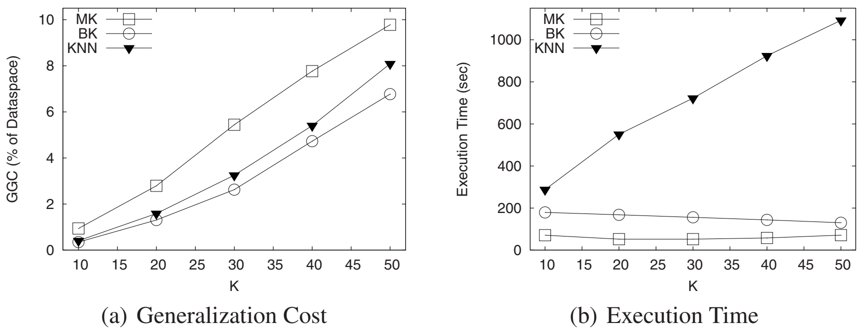


Fig. 15. Variable K, 4000 Sensitive Sites, 569120 Users

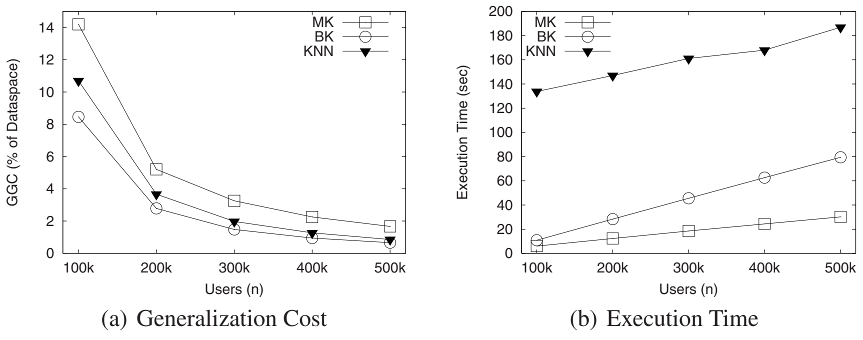


Fig. 16. Variable n, 2000 Sensitive Sites, $K = 20$

to K. However, fewer groups are generated and this reduces the user-to-site assignment phase of MK. As a result, the execution time remains almost constant.

Finally, in Figure 16 we fix $m = 2,000$, $K = 20$ and vary the number of users n. As n increases, the density of the users in the dataspace also increases, and more compact anonymizing sets can be formed. Therefore, GGC decreases with larger n for all methods. The execution time of KNN grows considerably with n, as more users need to be considered in the nearest-neighbor search. The execution time of both BK and MK is linear to n.

5.3 Discussion

Our two proposed methods, BK and MK, provide a clear tradeoff between generalization cost and execution time: BK is the best in terms of GGC out of all considered algorithms. It is also much faster than KNN and $Local$, but it is slower than MK. MK is faster at the expense of higher GGC (roughly 2 times worse than BK). Nevertheless, MK remains a good choice for applications where speed is essential; for instance, publishing real-time traffic updates.

$Local$ cannot be used for any input size of practical value, due to its extremely high computational overhead. Furthermore, $Local$ incurs very high generalization cost, if a secure location publishing format is used. Finally, the KNN method is outperformed by BK in terms of both GGC and execution time.

6 Conclusions

The collection of location data from mobile users has received considerable attention recently. To enable users with low-end communication devices (i.e., even without GPS) to access location-based services, certain LBS providers (e.g., GoogleMaps) have devised systems that calculate the user location from the identifiers of cellular network towers. As huge amounts of location data are becoming available, their privacy-preserving publication emerges as an important concern. In this paper we proposed two methods for the anonymous publishing of location data, which are fast and achieve low data distortion. Our methods are significantly better, compared to existing work.

In the future, we plan to study more complex attacks, based on traces of locations. By correlating information published at consecutive timestamps, an attacker may be able to gain additional knowledge and compromise the privacy of certain users. We also plan to address the scenario where the input location data is not entirely available before-hand, but instead it is generated in a streaming manner. This setting is more difficult, since data must be output before their expiration deadline; therefore, computational efficiency becomes a primary concern.

References

1. Aggarwal, C.C.: On k-Anonymity and the Curse of Dimensionality. In: Proc. of VLDB, pp. 901–909 (2005)
2. Bayardo, R., Agrawal, R.: Data Privacy through Optimal k-Anonymization. In: Proc. of ICDE, pp. 217–228 (2005)
3. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In: Proc. of ACM SIGMOD, pp. 322–331 (1990)
4. Bettini, C., SeanWang, X., Jajodia, S.: Protecting Privacy Against Location-Based Personal Identification. In: Jonker, W., Petković, M. (eds.) SDM 2005. LNCS, vol. 3674, pp. 185–199. Springer, Heidelberg (2005)
5. Chow, C.-Y., Mokbel, M.F.: Enabling Private Continuous Queries for Revealed User Locations. In: Papadias, D., Zhang, D., Kollios, G. (eds.) SSTD 2007. LNCS, vol. 4605, pp. 258–275. Springer, Heidelberg (2007)
6. Gedik, B., Liu, L.: Location Privacy in Mobile Systems: A Personalized Anonymization Model. In: Proc. of ICDCS, pp. 620–629 (2005)
7. Ghinita, G., Karras, P., Kalnis, P., Mamoulis, N.: Fast Data Anonymization with Low Information Loss. In: Proc. of VLDB, pp. 758–769 (2007)
8. Gruteser, M., Grunwald, D.: Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In: Proc. of USENIX MobiSys, pp. 31–42 (2003)
9. Hu, H., Xu, J., Du, J., Ng, J.K.-Y.: Privacy-Aware Location Publishing for Moving Clients. Technical report, Hong Kong Baptist University (2007), `http://www.comp.hkbu.edu.hk/~haibo/privacy_join.pdf`
10. Kalnis, P., Ghinita, G., Mouratidis, K., Papadias, D.: Preventing Location-Based Identity Inference in Anonymous Spatial Queries. IEEE TKDE 19(12), 1719–1733 (2007)

11. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient Full-Domain K-Anonymity. In: Proc. of ACM SIGMOD, pp. 49–60 (2005)
12. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-Diversity: Privacy Beyond k-Anonymity. In: Proc. of ICDE (2006)
13. Mokbel, M.F., Chow, C.Y., Aref, W.G.: The New Casper: Query Processing for Location Services without Compromising Privacy. In: Proc. of VLDB, pp. 763–774 (2006)
14. Moon, B., Jagadish, H., Faloutsos, C.: Analysis of the Clustering Properties of the Hilbert Space-Filling Curve. IEEE TKDE 13(1), 124–141 (2001)
15. Reid, D.: An algorithm for tracking multiple targets. IEEE Transactions on Automatic Control 24, 843–854 (1979)
16. Samarati, P.: Protecting Respondents' Identities in Microdata Release. IEEE TKDE 13(6), 1010–1027 (2001)
17. Sweeney, L.: k-Anonymity: A Model for Protecting Privacy. Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5), 557–570 (2002)
18. Tao, Y., Xiao, X.: Personalized Privacy Preservation. In: Proc. of ACM SIGMOD, pp. 229–240 (2006)