Anonymizing Moving Objects: How to Hide a MOB in a Crowd?

Roman Yarovoy[1] Francesco Bonchi[2*] Laks V. S. Lakshmanan[1] Wendy Hui Wang[3]

[1]University of British Columbia
Vancouver, BC, Canada
{yra,laks}@cs.ubc.ca

[2]Yahoo! Research
Barcelona, Spain
bonchi@yahoo-inc.com

[3]Stevens Institute of Technology
Hoboken, NJ, USA
hwang@cs.stevens.edu

ABSTRACT

Moving object databases (MOD) have gained much interest in recent years due to the advances in mobile communications and positioning technologies. Study of MOD can reveal useful information (e.g., traffic patterns and congestion trends) that can be used in applications for the common benefit. In order to mine and/or analyze the data, MOD must be published, which can pose a threat to the location privacy of a user. Indeed, based on prior knowledge of a user's location at several time points, an attacker can potentially associate that user to a specific moving object (MOB) in the published database and learn her position information at other time points.

In this paper, we study the problem of privacy-preserving publishing of moving object database. Unlike in microdata, we argue that in MOD, there does not exist a fixed set of quasi-identifier (QID) attributes for all the MOBs. Consequently the anonymization groups of MOBs (i.e., the sets of other MOBs within which to hide) may not be disjoint. Thus, there may exist MOBs that can be identified explicitly by combining different anonymization groups. We illustrate the pitfalls of simple adaptations of classical k-anonymity and develop a notion which we prove is robust against privacy attacks. We propose two approaches, namely *extreme-union* and *symmetric anonymization*, to build anonymization groups that provably satisfy our proposed k-anonymity requirement, as well as yield low information loss. We ran an extensive set of experiments on large real-world and synthetic datasets of vehicular traffic. Our results demonstrate the effectiveness of our approach.

1. INTRODUCTION

Recent years have witnessed the popular use of location-aware devices, e.g., GPS-enabled cell phones and PDAs, location sensors and active RFID tags. This enables monitoring of moving objects and real-time analysis of their motion patterns, as well as the collection of the traces left by these moving objects.

As an example, in the context of the GeoPKDD[1] project, we received a real-world and large dataset of trajectories of cars equipped with GPS and moving in the city of Milan (Italy). Indeed, many citizens accept to equip their car with GPS devices, because this way they obtain a substantial discount on the mandatory car insurance. As another example, a recent paper analyzes the case of a company in Hong Kong called Octopus [2] that collects daily trajectory data of Hong Kong residents who use Octopus smart RFID card [21]. The data could be published for further analysis of the movement and behavioral patterns of Hong Kong residents. Although publishing trajectory data is useful for obtaining knowledge in support of mobility-related decision making processes, sustainable mobility and intelligent transportation systems, it may represent a serious threat to the individual privacy. For example, as [21] pointed out, when a person, say Alice, uses her Octopus card to pay at different convenience stores that belong to the same chain (e.g., 7-Eleven), by collecting her transaction history in all these stores, the company can construct a subset of her complete trajectory. If this constructed trajectory uniquely identifies Alice (we call it the *quasi-identifier* of Alice's trajectory), then by matching it with the published trajectory database, even though the IDs of users may be removed, Alice still can be re-identified, as can the other locations that she visited.

In practice, the quasi-identifier values of moving objects can be obtained from diverse sources, e.g., from location based services such as the Octopus RFID cards shown above, or from the public knowledge (e.g., the travel schedule of the President published in the newspaper). We argue that unlike in relational microdata, where every tuple has the same set of quasi-identifier attributes, in mobility data we can not assume a set of particular locations, or a set of particular timestamps, to be a quasi-identifier for all the individuals. It is very likely that various moving objects have different quasi-identifiers and this should be taken into account in modeling adversarial knowledge.

Hence the concept of quasi-identifier must be modeled subjectively, on an individual basis.

Figure 1(a) shows a moving object database. Each row represents the trajectory of a moving object (MOB) and shows the position of the MOB at different times, in a rectangular grid. The original identities (say I_1, I_2, I_3) of the individuals have been suppressed and have been replaced by MOB ids O_1, O_2, O_3. For the three individuals I_1, I_2 and I_3, a possible scenario is that an adversary knows the location

[*] This work started while he was still a researcher at C.N.R., Pisa, Italy, supported by the EU project GeoPKDD (IST-6FP-014915).

[1] http://www.geopkdd.eu

[2] http://www.octopuscards.com

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the ACM. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permissions from the publisher, ACM.

EDBT 2009, March 24–26, 2009, Saint Petersburg, Russia.
Copyright 2009 ACM 978-1-60558-422-5/09/0003 ...$5.00

of I_1 at time point t_1, and that of I_2 and I_3 at t_2. In this paper, we study the problem of *privacy-preserving publishing of a moving objects database.* Our goal is to anonymize the trajectories so that no MOB can be re-identified by matching its quasi-identifier with the published dataset. We will return to this example database of Figure 1(a) and discuss the challenges in anonymizing it.

We borrow the concept of k-anonymity [19] for relational microdata. In the classical k-anonymity framework the attributes are partitioned into *quasi-identifiers* (i.e., a set of attributes whose values can be linked to external information to reidentify the individual), and *sensitive attributes* (publicly unknown, which we want to keep private). In the k-anonymity model, the values of the quasi-identifiers (QID in the following) are generalized to be less specific so that there are at least k individuals in the same group, which we call the *anonymization group*, who have the same (generalized) quasi-identifer values. Although it has been shown that the k-anonymity model presents some flaws and limitations [11], and that finding an optimal k-anonymization is NP-hard [2, 12], it remains a fundamental model of privacy with practical relevance. Unfortunately, due to the peculiar nature of the quasi-identifier values of mobility data, the existing k-anonymity framework of relational microdata is not sufficient to protect the privacy of moving object database.

The main issue in anonymizing MOB data is that, *due to the fact that different objects may have different QIDs, anonymization groups associated with different objects may not be disjoint*, as illustrated below.

Example 1. Consider the trajectories in Figure 1(a) and illustrated in Figure 1(c). Let $k = 2$ and $QID(O_1) = \{t_1\}$, $QID(O_2) = QID(O_3) = \{t_2\}$. Intuitively the best (w.r.t. information loss) anonymization group for O_1 w.r.t. its QID $\{t_1\}$ is $AS(O_1) = \{O_1, O_2\}$. This is illustrated in Figure 1(c) with a dark rectangle. This means in the anonymized database we assign the region $[(1, 2), (2, 3)]$ to O_1 and O_2 at time t_1. The best anonymization group for O_2 as well as for O_3 w.r.t. their QID $\{t_2\}$ is $\{O_2, O_3\}$. Thus, in the anonymized database, O_2 and O_3 will both be assigned to the common region $[(2, 6), (3, 7)]$ (the second dark rectangle) at time t_2. Clearly, the anonymization groups of O_1 and O_2 overlap. □

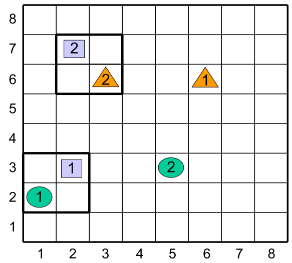
Due to this fact providing a robust and sound definition of k-anonymity in the case of MOD is challenging. The most obvious way of defining k-anonymity is to say that a given object must be indistinguishable from all the objects in its own anonymization group for each timestamp in its own QID. According to this definition the database in Figure 1(b) is 2-anonymous and thus "safe". This obvious definition of k-anonymity still suffers privacy breaches. Indeed, due the fact that anonymization groups may not be disjoint, *it is possible that by combining overlapping anonymization groups, some moving objects may be uniquely identified*, as explained next. Recall the previous example. There, I_1 and I_2 are in the same anonymization group (i.e., have the same generalized location) at time point t_1 (i.e., the QID of I_1), while I_2 and I_3 are in the same anonymization group at time point t_2 (i.e., the QID of I_2 and I_3). However, when the adversary tries to map the three MOBs O_1, O_2, O_3 to the three individuals I_1, I_2, I_3, with the adversary knowledge of QID values of these three MOBs, he can infer that I_1 must be mapped to either O_1 or O_2, while I_2 (and I_3) should be mapped to either O_2 or O_3. If I_1 is mapped to O_2, we cannot find a consistent assignment for I_2, I_3. As a result, *the adversary can conclude that O_1 must map to I_1*. Thus, we need a more sophisticated definition of k-anonymity to avoid privacy breaches in the case of moving object databases.

MOB	t_1	t_2
O_1	$(1, 2)$	$(5, 3)$
O_2	$(2, 3)$	$(2, 7)$
O_3	$(6, 6)$	$(3, 6)$

(a)

MOB	t_1	t_2
O_1	$[(1, 2), (2, 3)]$	$(5, 3)$
O_2	$[(1, 2), (2, 3)]$	$[(2, 6), (3, 7)]$
O_3	$(6, 6)$	$[(2, 6), (3, 7)]$

(b)



(c)

Figure 1: Example MOB Database (assuming $QID(O_1) = \{t_1\}$, $QID(O_2) = QID(O_3) = \{t_2\}$): (a) original database; (b) a 2-anonymity scheme that is not safe, and (c) its graphical representation.

Finally, another challenge arising from the fact that anonymization groups may not be disjoint, is that *overlapping anonymization groups can force us to revisit earlier generalizations*, as we will shown in Section 5.

In this paper, we make the following contributions:

- We illustrate the challenges in simple adaptations of k-anonymity for MOB databases and develop a notion of k-anonymity to the setting of moving objects and formally show that it does not lead to privacy breaches.
- We develop an efficient algorithm for finding good anonymization group for a given MOB w.r.t. its QID. Our algorithm makes a clever use of Hilbert indexes and top-K algorithms.
- We develop two algorithms, namely *extreme-union* and *symmetric anonymization*, for anonymizing MOB databases and show that both of them satisfy our notion of k-anonymity.
- Overlapping anonymization groups can cause previously computed generalizations to be revisited. We develop an efficient algorithm for computing the generalization of every MOB at every timestamp in one shot, without backtracking.
- We conducted an extensive set of experiments on large real-world and synthetic datasets of vehicular traffic to assess the effectiveness of our algorithms. Our results show that symmetric anonymization is faster than extreme union and leads to substantially less information loss.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 provides preliminary notions while Section 4 formally defines the privacy preservation problem. Section 5 introduces our anonymization algorithms. Experimental results are shown in Section 6. Finally, Section 7 concludes the paper. While some work exists on anonymity of MOBs (see Section 2), to the best of our knowledge, this is the first work tackling the challenges of anonymizing MOBs w.r.t. their QIDs by means of space-generalization.

2. RELATED WORK

Existing work on anonymity of spatio-temporal moving points has been mainly conducted in the context of *location based services* (LBS). In this context a trusted server is usually in charge of handling users' requests and passing them on to the service providers, and the general goal is to provide the service on-the-fly without threatening the anonymity of the user requiring the service. The trusted server uses *spatial-cloaking* (i.e., space generalization) to make the request sent from a user indistinguishable from $k-1$ other requests [7]. Based on this idea a system that supports different queries and various anonymization requirements was developed in [14, 13]. The approach introduced in [4] is geared on the concept of *location based quasi-identifier* (LBQID), i.e., a spatio-temporal pattern that can uniquely identify one individual. The idea is that if a service provider can successfully track the requests of a user through all the elements of a LBQID, then there would be at least $k-1$ other users whose personal history of locations is consistent with these requests, and thus may have issued those requests. The main difference with our context, is the fact that they consider data points (requests) continuously arriving, and thus they provide *on-line anonymity*: the anonymization group is chosen once and for all, at the time this is needed. This means that the $k-1$ MOBs passing closer to the point of the request are selected to form the anonymization group regardless of what they will do later, as this information is not yet available. In our context anonymity is *off-line.* Our aim is that of anonymizing a static database of moving objects, assuring that the *quality* of the data is kept high. To this end, we must select anonymization groups considering the MOBs as we already know them. Thus the solution proposed in [4] is not suitable for our purposes.

While there has been considerable work on anonymity in LBS, to the best of our knowledge only two papers (both published this year) tackle the problem of k-anonymity of moving objects by a data publishing perspective.

Abul *et al.* [1] propose the concept of (k, δ)-anonymity where δ represents the possible location imprecision. The idea here is to exploit this inherent uncertainty to reduce the amount of data distortion needed. A method for obtaining (k, δ)-anonymity is introduced and named $\mathcal{NWA}$ ($\mathcal{N}$ever $\mathcal{W}$alk $\mathcal{A}$lone). The method is based on trajectory clustering and spatial translation (i.e., moving in space the necessary points, in order to make a trajectory lie within the anonymity cylinder). The main difference of this work with our proposal is that it does not account for any notion of quasi-identifiers: the trajectories are clustered together considering their similarity all along their lifetime

Terrovitis and Mamoulis study the problem of protecting privacy in the publication of trajectory databases in [21]. The basic assumption of this work is that different adversaries own different, disjoint parts of the trajectories, and more importantly, the data publisher is aware of the adversarial knowledge. The privacy that is required is that any adversary can not learn anything more than what he already knows from the data publication. Anonymization is obtained only by means of *suppression* of the dangerous observations from each trajectory. As generalization is not used, it is not even needed to consider the spatial proximity of the locations, and many of the challenges (e.g., overlapping anonymity groups) that arise when using generalization, are not an issue in this setting.

3. PRELIMINARIES

By a *moving object database* (MOD), we mean a collection of moving objects (MOBs) $D = \{O_1, ..., O_n\}$ that correspond to n individuals, a set of m discrete time points $T = \{t_1, ..., t_m\}$, and a function $\mathcal{T} : D \times T \rightarrow \mathcal{R}^2$, that specifies, for each object O and a time t, its position at time t. In this paper, we use positive integers $1, 2, ...$ to denote position coordinates. This corresponds to using a grid to discretize space. The function $\mathcal{T}$ is called the trajectory. Indeed, $\mathcal{T}(O)$ denotes the trajectory of object O, i.e., its positions at times $t \in T$. For convenience, we also write a MOD as $D = \{(O_1, \tau_1), ..., (O_n, \tau_n)\}$ where O_i is a MOB and $\tau_i = \{(x_i^1, y_i^1, t_1), ..., (x_i^m, y_i^m, t_m)\}$ is O_i's trajectory, with (x_i^j, y_i^j) representing the position of O_i at time t_j. We represent a MOD as a table with rows corresponding to trajectories of MOBs and columns corresponding to times. An example of MOD is already given in Figure 1. We write $D(O, t)$ to denote the position of MOB O in the database D at time t. For the database in Figure 1, $D(O_1, t_1) = (1, 2)$.

In k-anonymization of microdata, which is basically relational, there is a fundamental notion of quasi-identifier attributes, which correspond to the attributes that constitute public knowledge and that may be used as key for a linking attack leading to re-identification of the individuals. E.g., `age`, `gender`, `zipcode` are typical QID attributes. In the case of a moving object database, where we think of times as attributes with objects' positions forming their values, we cannot assume a fixed set of times forms the quasi-identifier (QID) for every MOB. To capture this, we define the *quasi-identifier* as a function: $QID : \{O_1, ..., O_n\} \rightarrow 2^{\{t_1, ..., t_n\}}$. That is, every MOB may potentially have a distinct QID. Note that the timestamps in a QID need not be consecutive. The QIDs may be provided directly by the users when they subscribe to a service, or be part of the users' personalized settings (as it happens in many *location based services*), or they may be found by means of data analysis. Discussing how to obtain the users' QIDs is beyond the scope of this paper. In Section 6.2 we will describe how we generated synthetic QIDs for experimental purposes.

Given a MOD D, a *distorted version* of D is any database D^* over the same time points $\{t_1, ..., t_n\}$, where D^* contains one row for every MOB O in D, and either $D^*(O, t) = D(O, t)$ or $D(O, t) \sqsubseteq D^*(O, t)$, where with $\sqsubseteq$ we denote spatial containment among regions. A distorted version intuitively makes it harder to determine the exact position of an object at certain times. It never "lies" about the position of an object. Our goal is to find a distorted version of the MOD D, denoted by D^*, such that on the one hand, when published, D^* is still useful for analysis purposes, and on the other, a suitable version of k-anonymity is satisfied. The type of data distortion we allow is *space generalization.* In the input MOD D, each position is an exact point. With the application of a grid, we may regard each point as a cell. Figure 1(c) illustrates this for our running example database. We permit generalizations of points into rectangles. The dark rectangles in Figure 1(c) illustrate the idea. E.g., for O_1, in the published database D^*, we would publish its position as the rectangle defined by the endpoints $(1, 2)$ and $(2, 3)$. In future, we refer to rectangles as regions. We represent a region as $[(x, y), (x', y')]$ where (x, y) is its bottom-left corner, and (x', y') is its top-right corner. The region above is written as $[(1, 2), (2, 3)]$. By the area of a region, we mean the number of cells in it. E.g., both the regions corresponding to dark rectangles in Figure 1(c) have four cells in them and have an area of 4.

As in traditional k-anonymization, generalization results in information loss. We measure information loss as the reduction in the probability with which we can accurately determine the position of an object at a given time. For the above rectangle, with probability 1/4 we can determine the correct position of O_1 at t_1. Thus, the information loss for O_1 at this time is $1 - 1/4$ since in D, we know the exact position of O_1.

Given a distorted version D^* of a MOD D, we define the *information loss* as follows (where $area(D^*(O_i, t_j))$ denotes the area of the region $D^*(O_i, t_j)$):

$$\mathtt{IL}(D, D^*) = \Sigma_{i=1}^{n} \Sigma_{j=1}^{m} (1 - 1/area(D^*(O_i, t_j)).$$

Example 2. For the running example MOD, consider the generalized MOD D^* as in Figure 1(b). The information loss associated with D^* is $2 \times (1 - 1/4) + 2 \times (1 - 1/4) = 3$. □

When seeking a distorted version of a database for publishing, we want to minimize the information loss. At the same time, we want k-anonymity to be satisfied, where k is a user-supplied parameter. In the introduction, we showed that a direct application of the classical notion of k-anonymity does not work for MOB data. In the next section, we further illustrate the challenges in devising a suitable notion of k-anonymity for MOB data.

4. MOB ANONYMITY

A basic building block for devising any notion of anonymity is a notion of indistinguishability. Let D^* be a distorted version of a MOD D. We say that two MOBs O, O' are indistinguishable in D^* at time t provided that $D^*(O, t) = D^*(O', t)$, i.e., both are assigned to the same region in D^*. The most obvious way of defining k-anonymity is the following: a distorted version D^* of a MOD D satisfies *k-anonymity* provided: for every MOB O in D, $\exists k - 1$ other distinct MOBs $O_1, ..., O_{k-1}$ in D^*: $\forall t \in QID(O)$, O is indistinguishable from each of $O_1, ..., O_{k-1}$ at time t. Revisit Example 1 in the introduction. It is easy to see that the distorted database D^* in Figure 1(b) satisfies this notion. Yet, we saw that privacy can be breached in this case. We next formalize the attack model employed by the adversary and use it to show how privacy is breached. The first notion we introduce is an *attack graph*.

Definition 1 (Attack Graph). An attack graph associated with a MOD D and its distorted version D^* is a bipartite graph G consisting of nodes for every individual I in D (called *I-nodes*) and nodes for every MOB id O (called *O-nodes*) in the published database D^*. G contains an edge (I, O) iff $D(O, t) \sqsubseteq D^*(O, t), \forall t \in QID(I)$. □

An assignment of individuals to MOBs is consistent provided there exists a perfect matching in the bipartite graph G. In our running example, consider the distorted database shown in Figure 1(b): the corresponding attack graph is shown in Figure 2(a). It is obvious that the edge (I_1, O_1) must be a part of every perfect matching. Thus, by constructing the attack graph an attacker may easily conclude that MOB O_1 can be re-identified as I_1.

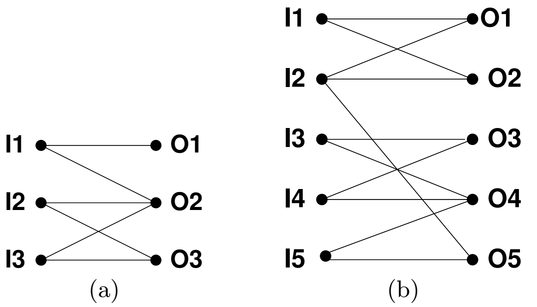


Figure 2: Attack graphs for different anonymization schemes: (a) for D^* in Figure 1(b); (b) for a hypothetical database D^* satisfying modified definition of k-anonymity.

One of the key shortcomings in the straightforward definition of k-anonymity given above is that while it ensures every I-node corresponding to an individual has at least k neighbors, it does not have any restriction on the degree of the O-nodes. *What if we required that in addition, every O-node must have degree at least k?* Suppose we say that a distorted database is k-anonymous provided in the corresponding attack graph, every I-node as well as every O-node has degree $\geq k$. Figure 2(b) shows a possible attack graph that satisfies this condition. In this graph, every I-node and every O-node has degree 2 or more. Yet, O_5 can be successfully re-identified as I_5 as follows. Suppose O_5 is instead assigned to I_2, to which it is adjacent as well. Then it is easy to see that no I-node can be assigned to one of O_1, O_2. Thus, *the edge (I_2, O_5) cannot be a part of any perfect matching*. Thus, this edge can be pruned, leaving I_5 as the only I-node to which O_5 can be assigned.

This example is subtler than the previous example and clearly shows the challenges involved in devising a notion of k-anonymity that does not admit privacy breaches. We next propose just such a notion. Before we do that, we formally specify the attack model.

Definition 2 (Attack Model). The attacker first constructs an attack graph associated with the published distorted version of D and the known QIDs as described in Definition 1. Then, he repeats the following operation until there is no change to the graph:

1. Identify an edge e that cannot be part of any perfect matching.
2. Prune the edge e.

Next, he/she identifies every node O with degree 1. He concludes the (only) edge incident on every such node must be part of every perfect matching. There is a privacy breach if the attacker succeeds in identifying at least one edge that must be part of every perfect matching. □

We already saw that both graphs in Figure 2 (and their underlying distorted databases) admit privacy breach. Next, we propose a notation of k-anonymity, which guarantees there exists a perfect matching for any arbitrary node in the attack graph.

Definition 3 (k-anonymity). Let D be a MOD and D^* its distorted version. Let G be the attack graph w.r.t. D, D^*, given a set of QIDs for the MOBs in D. Then D^* satisfies the k-anonymity condition provided: (i) every I-node in G has degree k or more; and (ii) G is symmetric, i.e., whenever G contains an edge (I_i, O_j), it also contains the edge (I_j, O_i). □

An immediate observation is that in an attack graph that satisfies the above conditions, every O-node will have degree k or more as well. We call a distorted database D^* *k-anonymous* if it satisfies Definition 3.

THEOREM 1 (CORRECTNESS). Let D^* be a k-anonymous version of a MOD D satisfying Definition 3. Then it does not admit any privacy breach w.r.t. the attack model defined in Definition 2.

Proof: Consider any edge (I_i, O_j) in the attack graph G. We will show that there is a perfect matching involving this edge. This shows that no edge can be eliminated. Since for every I-node there are at least k destinations and none of them can be pruned, the theorem follows from this. Notice that for every I_p, G necessarily contains the edge (I_p, O_p), where O_p is the counterpart, i.e., the anonymized version, of I_p. The trivial case is $j = i$, i.e., $O_j = O_i$ is indeed the anonymized version of I_i. In this case, the perfect matching can assign every I_p to the corresponding MOB O_p. So suppose $j \neq i$. Let I_j (resp., O_i) be the counterpart of O_j (resp., I_i). In this case, we can assign I_i to O_j. By symmetry, the edge (I_j, O_i) must exist and so we can assign I_j to O_i. For every other node $I_p, p \neq i, j$, we can assign I_p to O_p. This assignment is a perfect matching. This was to be shown. □

5. ALGORITHMS

The essence of computing the anonymization D^* is to determine anonymization groups, and then to generalize groups to common regions according to the QIDs. The quality of the grouping is decided by the information loss caused by generalization. The less the information loss, the better the grouping. For classical relational microdata, a k-anonymization also corresponds to a *disjoint partition* of the input relation. The QID attributes are fixed for the entire relation and tuples in each partition are generalized on these QID attributes. After generalization every tuple must be indistinguishable from at least $k-1$ others on the QID. We refer to the set of objects with which an object O is to be made indistinguishable (after generalization) on the QID as the *anonymization group $AG(O)$*.

We have already seen that when QIDs of objects are distinct, the anonymization groups need not be disjoint, and that this represents one of the main differences between anonymization in microdata and in MODs. Another challenge is that *overlapping anonymization groups can force us to revisit earlier generalizations*, as illustrated by the next example.

EXAMPLE 3. Consider a MOD $D = \{O_1, O_2, O_3\}$ with $QID(O_1) = \{t_1, t_2\}$ and $QID(O_2) = QID(O_3) = \{t_2, t_3\}$. Let $k = 2$. Suppose the trajectories are such that the best anonymization group we can find for O_1 is $AG(O_1) = \{O_1, O_2\}$, w.r.t. $QID(O_1)$. Now, we construct the best anonymization group for O_2 (and O_3). Suppose it is $AG(O_2) = AG(O_3) = \{O_2, O_3\}$, w.r.t. $QID(O_2)$ and $QID(O_3)$. Suppose according to the anonymization groups, we decide to generalize O_1 together with O_2 at times t_1 and t_2 first and then generalize O_2 and O_3 together at times t_2 and t_3. But then at time t_2, O_2 has already been generalized with O_1. So, we then have to revisit the previous generalization and then adjust it to make sure that at time t_2, objects O_1, O_2, O_3 are all assigned the same region. □

This example shows that after we find the best anonymization groups for objects, if we start generalizing objects (w.r.t. their QIDs) along with all objects in their anonymization group, we may sometimes have to revisit previous generalizations and possibly re-generalize them with other objects. This happens precisely because some anonymization groups overlap both in objects and in QIDs. This phenomenon causes computational difficulties. The challenge is *whether we can find a schedule for generalizing objects along with a set of objects (w.r.t. their QIDs) so that we don't have to revisit previously computed generalizations*.

The next challenge is finding anonymization groups for MOBs in such a way that the attack graph induced by them satisfies the condition of k-anonymity. We organize the rest of this section as follows. In the next section, we focus on finding anonymization groups. In Section 5.2, we propose an approach for computing generalizations of object trajectories given possibly overlapping anonymization groups. Our algorithm avoids backtracking and computes the generalized position of every object at every timestamp in one shot.

Before we proceed presenting the algorithms for finding anonymization groups, we recall from [12, 10] that for classical relational microdata, finding optimal k-anonymization is NP-hard. It is not difficult to show that this hardness carries over to the problem of finding k-anonymizations with minimum information loss for MOBs. Given this, we focus on efficient heuristics.

5.1 Computing anonymization groups

One of the key steps in finding a k-anonymization of a MOD is finding the so-called anonymization group: given a MOB O, find the $(k-1)$ best other MOBs $\{O_i, ..., O_{k-1}\}$ such that when O is generalized together with $O_i, ..., O_{k-1}$ on all time stamps $t \in QID(O)$, the resulting generalization will have minimum information loss. In the following, when we speak of computing the anonymization group of a MOB O, we will refer to O as the *subject* and speak of distances of other MOBs from the subject at a given time.

Finding anonymization groups is beset with two challenges: (i) computing the distance from a MOB to other MOBs is expensive, and (ii) finding the best candidates for the anonymization group, i.e., those MOBs whose aggregate distance from the subject over all its QID time points is minimum, is expensive independently of the previous step. Intuitively, we could pick the $k-1$ MOBs which are nearest neighbors to the subject. However, MOBs which are close to the subject at one time, may be far from it at another. Figure 3 illustrates this point: Ignore the numbers inside the cells for now. They will be explained shortly. Figure 3(a) (resp., (b)) shows the positions of three MOBs at time t_1 (resp., time t_2).

What we seek is a set of $k-1$ MOBs which have the minimum aggregate distance from the subject O over the entire set of times in $QID(O)$.

We could create any data structure for fast retrieval of nearest neighbors at every time point. There is vast literature on data structures and efficient algorithms for kNN of moving objects [15, 22, 20]. We make use of space filling curves for this purpose. Specifically, we use the Hilbert index of spatial objects for efficient indexing of MOBs at each time point. The Hilbert curve [9] is a continuous fractal space-filling curve that naturally maps a multi-dimensional space to one dimension. The Hilbert index of a list of points

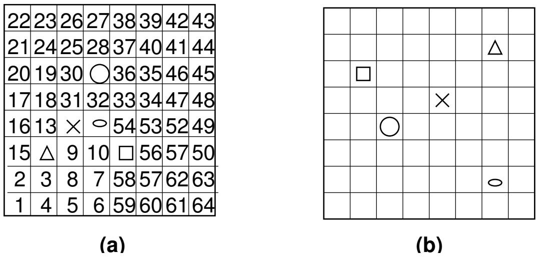


Figure 3: Illustration of MOB positions and their Hilbert indexes: (a) time t_1; (b) time t_2.

is assigned following the order in which the Hilbert curve visits these points in an n-dimensional space. It is well known that the Hilbert index preserves *locality*, i.e., points close in the multi-dimensional space remain close in the linear Hilbert ordering. Figure 3 illustrates how Hilbert index is assigned to object positions corresponding to various cells in a two-dimensional grid. It is easy to see that it preserves locality. We make use of this property. Specifically, at each time point t, we construct the Hilbert index of all MOBs according to their locations at t. Hamilton and Rau-Chaplin [8] have devised an efficient algorithm for computing the Hilbert index of objects in multiple dimensions. We make use of their algorithm (see Algorithm 2 in [8]) to compute the Hilbert index of MOBs at any given time. In our implementation, we insert MOB id's into a list in sorted order of their Hilbert index. Given a MOB O, denote its Hilbert index at time t as $H_t(O)$. We refer to the list of MOBs and their Hilbert index at time t as the *Hilbert list of MOBs* at time t and denote it as L_t. Each entry in L_t is of the form $(O, H_t(O))$, containing a MOB id and its Hilbert index.

By the *deviation at time t* between two MOBs O, O', we mean the absolute difference $|H_t(O) - H_t(O')|$. A key advantage of Hilbert lists is that given a subject O, the MOBs which are closest to O at t can be approximated by MOBs with the least deviation from O at t, which in turn can be efficiently found from the list L_t. Our goal though, is to find those MOBs whose aggregate distance from a subject O over a set of time points is the least possible. Any aggregate function such as avg, sum etc. can be used. We use sum in our exposition. We can reduce this problem to that of finding the top-K MOBs as follows.

Given a subject O, define the local score of a MOB O' in L_t to be the deviation $score_t(O') = |H_t(O) - H_t(O')|$. The overall score of O' is $score(O') = \Sigma_{t \in QID(O)} score_t(O')$. The problem of finding MOBs with the closest aggregate distance from O thus reduces to finding the top-K MOBs with the lowest overall score, where $K = k - 1$. Efficient answering of top-K queries where scores are computed by aggregation of local scores has been studied extensively in recent years. In particular, a family of algorithms were developed and analyzed by Fagin et al. [16, 17] and improved by other researchers subsequently (e.g., see [18]). The key insight is that by storing the lists containing objects with local scores in descending order of scores and by maintaining a threshold score based on objects examined so far, one can stop the processing early. One of the landmark algorithms is the so-called *Threshold Algorithm* (TA) [16, 17]. We adopt the recent improvement of TA, which was developed in [18] and referred to as Best Position Algorithm (or BPA2). *We cannot use this algorithm directly as the subject MOB is not fixed.* Thus, we cannot order the lists L_t in order of scores, since the scores are not fixed! However, we can solve this by simply storing the lists L_t in increasing order of the Hilbert index. Given a subject MOB O, other MOBs which are closest to O according to their Hilbert index, and hence have the smallest $score_t()$, are adjacent to O in the list L_t, but *on both sides of O*. Thus, while in standard top-K algorithms such as TA, we walk down the score lists, ordered in decreasing order of scores, with Hilbert lists, we need to identify the position of the subject on each list, and move away from it on either side of the subject. E.g., in Figure 4(b), the Hilbert index of the subject is 33 and the score (i.e., deviation) of the MOBs increases as we move away from the subject: the score of the triangle is $|33 - 19| = 14$, that of the rectangle is $|33 - 41| = 8$, etc.

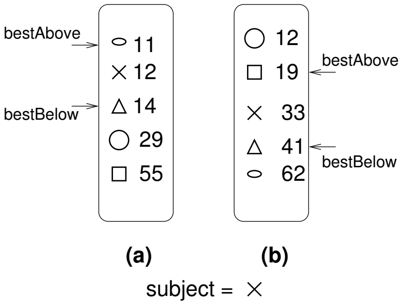


Figure 4: Illustration of Hilbert Lists: (a) time t_1; (b) time t_2.

We next describe our adaptation of Algorithm BPA2 from [18]. We perform two kinds of accesses to each list – sorted and random. Random access probes the list given the MOB id. Sorted access, on the other hand, advances the cursor from the last seen MOB (under both sorted and random access) to the next adjacent MOB. Since we need to move both "above" and "below" the subject, for each list, we maintain a pair of cursors – *bestAbove* and *bestBelow*. Here, *bestAbove* (resp., *bestBelow*) refers to the smallest (resp., largest) position above (below) the subject position, that has been examined so far (see Figure 4). By the *position* of an entry in a Hilbert list, we mean its position from the top. E.g., in Figure 4(a), the position of the subject MOB is 2. Section 5.1.1 describes the algorithm. Given a Hilbert list L_t and a position i, we use $L_t[i].MOB$ (resp., $L_t[i].ind$) to denote the MOB (resp., Hilbert index) component of the entry at position i. E.g., in Figure 4(a), $L_{t_1}[3].MOB$ is the triangle and $L_{t_1}[3].ind = 14$.

5.1.1 Algorithm GenAG

First, for every time point t, we compute the Hilbert index $H_t(O)$ of every MOB O and insert the entries $(O, H_t(O))$ into a list L_t in increasing order of $H_t(O)$. We will use these lists repeatedly for computing the anonymization group of different subjects.

Algorithm GenAG as presented above, considers all MOBs in the lists as candidates for $AG(O_i)$. Sometimes, we may want to exclude certain MOBs from consideration. This is easily achieved by ignoring list entries corresponding to those MOBs that are supposed to be excluded.

It was shown in [17] that Algorithm TA has the *instance optimality* property (see [17] for details). This property is inherited by BPA2 and also by GenAG. Intuitively, instance optimality guarantees that any algorithm, which does not make any wild guesses, cannot compute the answer to the top-K query (i.e., compute $AG(O_i)$) in fewer accesses than *GenAG* algorithm by more than a constant factor, on any input database.

Algorithm 1 GenAG

```
Input: Subject MOB $O_i$ and anonymity threshold $k$;
Output: anonymization group $AG(O_i)$.
1: Let $QID(O_i) = \{t_1, ..., t_p\}$ be the $QID$ of $O_i$. Let K = $k - 1$.
2: Do a sorted access to each of the lists $L_{t_1}, ..., L_{t_p}$ in parallel, say in a round robin manner. First, locate the entry $(O_i, H_t(O_i))$ containing the subject. Let $cPos$ be the position of this entry in $L_t$. Whenever a MOB $O$ is seen under sorted access in a list $L_t$, set its local score to $|H_t(O_i) - H_t(O)|$. Obtain its local score in other lists by a random access and compute its overall score as the sum of local scores. Keep track of all seen positions (sorted or random) in all lists as well as the corresponding local scores. Maintain a top-K heap containing the K MOBs with the best overall scores seen so far as well as the scores.
3: For each list $L_t$, let $Above$ (resp., $Below$) denote the set of all positions above (resp., below) that of the subject $O_i$ (at position $cPos$) that have been seen (sorted or random). Let $bestAbove$ (resp., $bestBelow$) be the smallest (largest) position in $Above$ ($Below$) such that every position at or below (at or above) it, and above (below) the subject, has been seen. Let the pointer $best$ point to whichever of $bestAbove$ or $bestBelow$ has been seen last. Set the "worst" (i.e., the smallest) local score for list $L_t$ as $|L_t[best].ind - L_t[cPos].ind|$. The threshold is the sum of worst local scores for all times $t \in QID(O_i)$. If the highest score in the top-K heap is less than or equal to the threshold, return the MOBs in the heap as the anonymization group $AG(O_i, k)$.
4: Compare the local scores at $bestAbove - 1$ and at $bestBelow +$ 1 and choose the entry with the smaller local score. Advance the appropriate pointer ($bestAbove$ or $bestBelow$).
```

5.1.2 Extreme Union

Algorithm GenAG produces the top-k candidates for forming the anonymization group of a given object. As we have seen in the initial sections, simply generalizing an object O with the MOBs in the set $AG(O)$ may not lead to an attack graph that satisfies the condition of k-anonymity. Thus, a key question is how to expand the anonymity groups produced by GenAG in order to ensure k-anonymity. In this section, we discuss one such algorithm for doing so.

For each MOB O, we compute its hiding set $AG(O)$ using Algorithm GenAG. Then we take the union of the QIDs of all MOBs in $AG(O)$ and generalize all of them w.r.t. every time point in this union. Algorithm 2 shows the pseudocode.

Algorithm 2 Extreme Union

```
Input: $D, QID(O_i), O_i \in D$, and anonymity threshold $k$;
Output: $D^*$, a $k$-anonymization of $D$.
// Collect moving obects that have defined $QID$s
$QO = \{O_i \mid QID(O_i) \neq \emptyset, O_i \in D\}$;
for all $O_i \in QO$ do
    $AG(O_i)$ ← anonymization group of $O_i$ w.r.t. $QID(O_i)$; // Using Algorithm GenAG.
    for all $O_j \in AG(O_i)$ do
        TimeStampUnion ← $\bigcup_{O_j \in AG(O_i)} QID(O_j)$;
    generalize all objects in $AG(O_i)$ together w.r.t. every timestamp in TimeStampUnion. // Using Algorithm 4.
```

The last step in Algorithm 2 involves computing the generalized positions (i.e., regions) for MOBs at various timestamps. Given that the anonymization groups are overlapping, this must be done with care in order to avoid backtracking and revisiting of previously computed generalizations. We deal with this issue in Section 5.2 and Algorithm 4. Next, we show that the anonymization produced by Algorithm 2 is indeed k-anonymous.

Theorem 2 (Correctness of Extreme Union). The generalized database produced by Algorithm 2 is k-anonymous.

Proof: Let $AG(O)$ be the anonymization group computed for any MOB O. According to the algorithm, all the MOBs in $AG(O)$ are generalized w.r.t. all times in the union of QID sets of all objects in $AG(O)$. Let $AG(O) = \{O_1, ..., O_k\}$. Notice that by Algorithm 1, this set will contain exactly k objects. From construction, it follows that the induced attack graph will contain a complete bipartite subgraph between nodes $\{I_1, ..., I_k\}$ and nodes $\{O_1, ..., O_k\}$. These remarks hold for any object O in the input database. Thus, it follows that the attack graph satisfies the conditions in Definition 3. □

As an illustration of the extreme union approach, in our running example, $AG(O_1)$, computed w.r.t. $QID(O_1) = \{t_1\}$, is $\{O_1, O_2\}$. Since $QID(O_2) = \{t_2\}$, the extreme union approach will generalize O_1 and O_2 together w.r.t. both t_1 and t_2. Similarly, it will generalize O_2 and O_3 together w.r.t. t_2.

While Algorithm 2 does produce generalizations that are k-anonymous, it can result in considerable information loss. The reason is that when $AG(O)$ is computed, we may pick the $k-1$ objects that (heuristically) minimize the area of the generalized region w.r.t. those timestamps in $QID(O)$. But then at the end, objects in $AG(O)$ are generalized w.r.t. not just $QID(O)$ but timestamps that are in the QID in any object in $AG(O)$. For timestamps outside $QID(O)$, objects in $AG(O)$ may be arbitrarily far away from O. E.g., in our running example, $D(O_1, t_2) = (5, 3)$, while $D(O_2, t_2) = (2, 7)$, and generalizing these together leads to a large region. To address this concern, in the next section, we ask the question whether we can generalize objects less aggressively than extreme union and still meet the k-anonymity condition.

5.1.3 Symmetric Anonymization

A key condition in the definition of k-anonymity is that the induced attack graph must be symmetric. Extreme union achieves this at the expense of generalizing all objects in an anonymization group w.r.t. the QIDs of all MOBs in the group. Thus, it keeps the set of objects being generalized together fixed, and equal to the original anonymization group computed. An alternative approach is to keep the timestamps w.r.t. which a set of MOBs is generalized together, fixed and equal to $QID(O_i)$. We must then control the composition of the anonymization groups in order to ensure symmetry. Since the anonymization groups computed by this algorithm may not exactly correspond to those computed using Algorithm 1, we use a different term, *hiding sets* (denoted HS), to describe them, for the sake of clarity. Hiding sets serve the same purpose as anonymization groups, to drive the generalization step. Algorithm 3 gives the pesudocode. As with Algorithm 2, we appeal to Algorithm 4 (to be given in Section 5.2) for actually computing the generalization.

For our running example, suppose the MOBs are visited in the order O_1, O_2, O_3. Assume we consider 2-anonymity. $HS(O_1)$ is initially set to $\{O_1\}$. Then $HS(O_1)$ is set to $\{O_1, O_2\}$. To enforce symmetry, we set $HS(O_2) = \{O_2, O_1\}$. Since the slack associated with O_2 now is 0, we move on to O_3, and compute $HS(O_3) = \{O_3, O_2\}$. Again, to enforce symmetry, we go back and add O_3 to $HS(O_2)$, so $HS(O_2) = \{O_1, O_2, O_3\}$. At this point, the HS sets are

User	MOB	QID	$AG(O)$	TimeStampUnion
A	O_1	t_1, t_2	O_1, O_2	t_1, t_2, t_3
B	O_2	t_3	O_2, O_4	t_3, t_4
C	O_3	t_2, t_4	O_3, O_4	t_2, t_4
D	O_4	t_4	O_4, O_2	t_3, t_4
E	O_5	t_1, t_3, t_4	O_5, O_1	t_1, t_2, t_3, t_4

(a)

MOB	QID	$HS(O)$
O_1	t_1, t_2	O_1, O_2, O_5
O_2	t_3	O_2, O_1
O_3	t_2, t_4	O_3, O_4
O_4	t_4	O_4, O_3
O_5	t_1, t_3, t_4	O_5, O_1

(b)

Figure 5: Difference between: (a) Extreme Union and (b) Symmetric Anonymization.

all symmetric and all the slack values are 0. Algorithm 4 is invoked and it generalizes O_1, O_2 at timestamp t_1 and O_1, O_2, O_3 at timestamp t_2. In this example, the final result of symmetric anonymization coincides with that of extreme union, even though the mechanics of the two algorithms are different. The following example illustrates the difference between these two approaches.

Algorithm 3 SYMMETRIC ANONYMIZATION

```
Input: $D, QID(O_i), \forall O_i \in D$, and anonymity threshold $k$;
Output: $D^*$, a $k$-anonymization of $D$.
$QO = \{O_i \mid QID(O_i) \neq \emptyset, O_i \in D\}$; // Collect moving
    objects that have defined $QID$s
for all $O_i \in QO$ do
    $HS(O_i) \leftarrow \{O_i\}$;
    $slack(O_i) \longleftarrow k - 1$;
for all $O_i \in QO$ do
    if $slack(O_i) > 0$ then
        $HS(O_i) \leftarrow HS(O_i) \cup$ {top-$slack(O_i)$ MOBs
            computed using Algorithm GENAG }
        for all $O_j \in HS(O_i), j \neq i$ do
            $HS(O_j) \leftarrow HS(O_j) \cup \{O_i\}$
            $slack(O_j) \leftarrow k - \mid HS(O_j) \mid$;
for all $O_i \in QO$ do
    generalize all MOBs in $HS(O_i)$ together w.r.t. $QID(O_i)$
    // using Algorithm 4.
```

EXAMPLE 4. Let there be five individuals A (Alice), B (Bob), C (Chloe), D (Dave), and E (Erika), who are pseudo-anonymized into the respective MOB ids $O_1, ..., O_5$. Suppose $k = 2$. Figure 5(a) shows the QIDs associated with the MOBs as well as the evolution of the computation using extreme union. The AG sets are computed using Algorithm 1, which finds the approximate nearest MOBs to a given MOB, w.r.t. the subject's QID. For each AG set, we take the union of the QID sets of all MOBs in the set. E.g., for $AG(O_1)$, this gives $QID(O_1) \cup QID(O_2) = \{t_1, t_2, t_3\}$. Finally, using Algorithm 4, the generalized database is computed, meeting the specifications of anonymization groups and timestamps in Figure 5(a). We will return to this in the next section. Figure 5(b) shows the evolution of the computation using symmetric anonymization. Initially, all $HS(O)$ sets are set to contain only the MOB O. The slack for every MOB at this point is $2-1 = 1$. Next, compute the approximate nearest neighbor for O_1 w.r.t. the timestamps t_1, t_2. This is O_2. In addition to adding O_2 to $HS(O_1)$, we add O_1 to $HS(O_2)$ (symmetry), making the slack of O_2 zero. Similarly, O_4 is determined to be the approximate nearest neighbor of O_3 and added to $HS(O_3)$. Then O_3 is added to $HS(O_4)$ for symmetry. Finally, O_1 is added to $HS(O_5)$, which by symmetry, forces O_5 to be added to $HS(O_1)$. At this point, the algorithm invokes Algorithm 4 in order to compute the generalized database.

While we will discuss the details of the generalization in the next section, a quick inspection of Figure 5(a)-(b) should reveal the differences in the final output expected from extreme union and symmetric anonymization. □

THEOREM 3 (CORRECTNESS OF ALGORITHM 3). The generalized database produced by Algorithm 3 is k-anonymous.

Proof: Definition 3 has two requirements: symmetry of the attack graph, and degree no less than k. Symmetry is explicitly enforced by Algorithm 3. Moreover, every hiding set that is produced has size k or more. □

Before we leave this section, extreme union and symmetric anonymization ensure k-anonymity by different means: by expanding the set of timestamps or by controlling the composition of anonymization groups/hiding sets. In Section 6, we compare their relative performance both w.r.t. speed and information loss.

5.2 Computing Generalizations

The problem we tackle in this section is the following. Given a set of possibly overlapping anonymity (or hiding) sets for a set of MOBs, and associated sets of timestamps, how can we compute the generalized database efficiently? This problem arises regardless of whether we use extreme union or symmetric anonymization for generating the sets. More formally, let D be an input database containing MOBs $O_1, ..., O_n$. Let $\{S(O_i) \mid S(O_i) \subseteq \{O_1, ..., O_n\}, 1 \leq i \leq n\}$ be a collection of sets of MOBs, and let $\{T(O_i) \mid 1 \leq i \leq n\}$ be a set of sets of timestamps. Then we wish to compute the generalization D^* of the input database D such that $\forall O_i : \forall t \in T(O_i) : \forall O_j \in S(O_i) : D^*(O_i, t) = D^*(O_j, t)$.

Algorithm 4 GENERALIZE

```
Input: $D$, $QID(O_i)(\forall O_i \in D)$, threshold $k$, and the set of
    MOB sets $\mathcal{A}$,
Output: $D^*$, a $k$-anonymous version of $D$.
For all $O_i \in D$, let $S(O_i)$ be the anonymity (or hiding) set
    computed using extreme union (symmetric anonymization)
    and let $T(O_i)$ be the set of timestamps associated with $S(O_i)$
    under either approach;
Let $QIT$ be the union of all timestamp sets $T(O_i), O_i \in D$.
for all $t \in QIT$ do
    $EC_t$ = GENEQUIVCLASS($\mathcal{A}$,$t$); // Using Algorithm 5
    for all $C \in EC_t$ do
        for all $O \in C$ do
            $D^*(O, t) \leftarrow lub(C)$;
```

Let $\mathcal{A}$ be the set of anonymity/hiding sets of MOBs in D, computed using extreme union or symmetric anonymization (Sections 5.1.2 and 5.1.3). For the sake of neutrality, we will henceforth refer to the sets in $\mathcal{A}$ as MOB sets. Depending on the context, MOB sets can mean either anonymization groups or hiding sets. As we have seen before, these sets need not be disjoint. Suppose there is a MOB set $S(O_j) \in \mathcal{A}$ such that $O_i \in S(O_j)$. Then by the definition of $S(O_j)$, O_i and O_j should be indistinguishable at every time

$t \in QID(O_j)$, in any k-anonymized database D^* that uses this anonymization groups. Indistinguishability is clearly an equivalence relation, which motivates the following.

Definition 4 (Equivalence Class).
Given two MOBs O_i, O_j, we say O_i and O_j are equivalent at time t, $O_i \equiv_t O_j$, iff: (1) $t \in QID(O_j)$ and $O_i \in S(O_j)$, or (2) $t \in QID(O_i)$ and $O_j \in S(O_i)$ (i.e., the symmetric variant of (1)), or (3) there exists a MOB $O_k \neq O_i, O_j$ s.t. $O_i \equiv_t O_k$ and $O_k \equiv_t O_j$. □

If we can compute the equivalence classes for every time $t \in QIT$ efficiently, then a k-anonymization D^* of D can be obtained as follows. For each time $t \in QIT$, for each equivalence class C w.r.t. time t, generalize the position of every MOB $O \in C$ to $lub(C)$, i.e., the least upper bound of the positions of objects in C w.r.t. the lattice of rectangles. This corresponds to the smallest axis-parallel rectangle containing the positions of all MOBs in C at time t. E.g., for $S = \{O_1, O_2\}$ in our running example, $lub_{t_1}(S) = [(1,2),(2,3)]$ (see Figure 1(c)). This explains Steps 5-7 in Algorithm 4.

Algorithm 5 captures the details of Step 4. It computes the equivalence classes w.r.t. time t by running through every MOB O_i with $t \in QID(O_i)$ and then adding $S(O_i)$ to the collection EC_t. We maintain the collection EC_t using the well-known UNION/FIND data structure for disjoint sets, with path compression [6]. When a new $S(O_i)$ enters EC_t, we check for overlap with an existing set in EC_t by issuing a FIND query against EC_t w.r.t. every MOB in $S(O_i)$ (Step 7). If it returns null, i.e., no overlapping set is found, we make a new set (Step 9). Otherwise, we merge the object with the existing set (Step 11 - 13). To illustrate, suppose $S(O_i) = \{O_1, O_2, O_3\}$. When we query for O_1, suppose we find $C_1 \in EC_t$. Now, we query for O_2. If we find another set $C_2 \in EC_t$, then we union (i.e., merge) C_1 and C_2 using the tree merge and path compression. Finally, if we find $C_3 \in EC_t$ when we query for O_3, we merge C_3 with the merge of C_1 and C_2.

Algorithm 5 GenEquivClass

```
Input: The set of MOB sets $\mathcal{A}$, a timestamp $t$;
Output: $EC_t$, the set of all equivalence classes at time $t$.
 1: $EC_t \leftarrow \emptyset$;
 2: for all $S(O) \in \mathcal{A}$ do
 3:     $New_{set} \leftarrow \{\}$;
 4:     $Merge_{Set} \leftarrow \{\}$;
 5:     for all $O_i \in AG(O)$ do
 6:         if $t \in QID(O_i)$ then
 7:             $root \leftarrow$ Find($Q_i, EC_t$).
 8:             if $root == NULL$ then
 9:                 Insert $root$ into $New_{set}$;
10:             else
11:                 Insert $root$ into $Merge_{set}$;
12:     $S \leftarrow MakeSet(New_{Set})$;
13:     $S \leftarrow Union(S, Merge_{set})$;
14:     Add $S$ to $EC_t$;
15: Return $EC_t$;
```

Example 5. We illustrate Algorithm 4 on the database of Figure 5. First, consider extreme union, for which the MOB sets are given in Figure 5(a). For the timestamp t_1, the algorithm will first add the set $\{O_1, O_2\}$ to EC_{t_1}. The only other MOB whose QID contains t_1 is O_5. We find that its MOB set $\{O_5, O_1\}$ overlaps the only class in EC_{t_1} so we expand that class to $\{O_1, O_2, O_5\}$. On the other hand, for t_2, we start with O_1 again and the set $\{O_1, O_2\}$ is added to EC_{t_2}. The next MOB containing t_2 in its QID set is O_3. When we query EC_{t_2} with the contents of the MOB set $S(O_3) = \{O_3, O_4\}$, we find no result, so we start a new class $\{O_3, O_4\}$. Eventually, when we process O_5, we find that its MOB set $S(O_5) = \{O_5, O_1\}$ overlaps an existing class so we expand that class to $\{O_1, O_2, O_5\}$. The complete set of equivalence classes is shown in Figure 6(a).

When the same algorithm is run on the input MOB sets produced by symmetric anonymization, shown in Figure 5(b), it results in the equivalence classes shown in Figure 6(b). □

t	EC_t	EC_t
t_1	$\{O_1, O_2, O_5\}$	$\{O_1, O_2, O_5\}$
t_2	$\{O_1, O_2, O_5\}, \{O_3, O_4\}$	$\{O_1, O_2, O_5\}, \{O_3, O_4\}$
t_3	$\{O_1, O_2, O_4, O_5\}$	$\{O_1, O_2, O_5\}$
t_4	$\{O_1, O_5\}, \{O_2, O_3, O_4\}$	$\{O_1, O_5\}, \{O_3, O_4\}$
	(a)	(b)

Figure 6: Equivalence classes produced by Algorithm 5 when applied to: (a) the anonymization groups produced by Extreme Union (reported in Figure 5(a)), and (b) the hiding sets produced by Symmetric Anonymization (in Figure 5(b)).

In this example, it turns out that the equivalence classes produced from the output of symmetric anonymization are smaller than those produced from extreme union. Intuitively, the smaller the classes, the less general the regions. However, this is not always true and depends on the relative positions of the objects. In general, theoretically extreme union and symmetric anonymization are incomparable in terms of the information loss they result in. Extreme union, by taking the union of QIDs of the objects in each of the anonymization sets, may lead to coarse equivalence classes. On the other hand, symmetric anonymization does not change QID sets. However, it may result in hiding sets whose size is $> k$, since some of the membership is forced by symmetry. E.g., in the above example, O_5 is added to $HS(O_1)$ because of symmetry, making $|HS(O_1)| = 3$. Furthermore, when an object O' is added to a hiding set $HS(O)$ in order to enforce symmetry, it is possible O' is far away from O w.r.t. $QID(O)$. Thus, analytically, we cannot show that either of the algorithms – extreme union or symmetric anonymization – dominates the other. We conduct a detailed comparison between these algorithms in Section 6, where we compare them w.r.t. speed and information loss.

5.2.1 Discussion of Complexity

Our approach to computing the generalization of a MOD makes use of Algorithms 1-5. As already discussed, Algorithm 1 inherits the instance optimality of TA and BPA2 algorithms from [17, 18], w.r.t. the number of sorted and random access it makes on any input database instance, where the database instance here refers to the set of Hilbert lists. Algorithm 5 makes use of the disjoint sets union/find data structure with path compression and can be shown to take $O(nk\alpha(nk))$ time. Algorithm 4 invokes Algorithm 5 m times, where m is the number of time points in the MOD D. Generalization (steps 6-7 of Algorithm 4) adds an additional complexity of $O(mn)$, so the overall complexity is $O(mnk\alpha(nk))$.

6. EXPERIMENTAL EVALUATION

In this section we report the empirical evaluation we conducted in order to assess the performance of our algorithms.

6.1 Experimental Data

We experimented on both a real-world trajectory dataset, and a synthetic one. The first one is a dataset used in the GeoPKDD project[3]: after pre-processing, it contains more than $45k$ trajectories and a total of approximately $500k$ spatio-temporal points. These trajectories are obtained by GPS-equipped cars moving in the city of Milan (Italy) starting from January 2007. During the pre-processing we enforced a constant sampling rate of one point every 5 minutes. We did not use interpolation to reconstruct the missing points: when a point is missing the object is assumed to stay still in the last position observed. We also cut trajectories when two consecutive observations where too far in time or in space. Moreover, we cut trajectories everyday at 3 a.m., to focus on a single day. At the end the dataset contains 2009 different timestamps. Finally we removed completely noisy trajectories such as trajectories with too small temporal or spatial extension. The second dataset is synthetic and it has been generated using Brinkhoff's network-based generator of moving objects [5]: it contains $150,000$ trajectories with 400 distinct timestamps over the road-network of the city of Oldenburg (Germany). The former dataset is referred to as Milan, and the latter as Oldenburg henceforth. In Table 1 we report the characteristics of the two datasets.

6.2 Experiments settings

Our algorithms were implemented using Java 6.0. The experiments were run on an Intel Pentium 4, 3.00GHz, Linux machine equipped with 2Gb main memory.

To measure the effects of QIDs on the performance of our methods, for both Milan and Oldenburg datasets we generated QIDs under various different configurations. E.g., QIDs randomly chosen, all QIDs are disjoint, all QIDs form a single chain, i.e., every two adjacent QIDs share at least one timestamp, and all QIDs form multiple disjoint chains, forest of trees, etc. For each configuration of QIDs, we constructed a set of QID instances of different size. We varied the number of timestamps in each QID as well as the number of timestamps shared by multiple QIDs. For lack of space we omit the analysis of the effects of shape and size of QIDs on the overall performance. We plan to present a detailed analysis on an extended version of this paper. For the experiments reported in the rest of this section we used the same configuration for both Milan and Oldenburg, i.e., QIDs formed following a *random graph* approach. In this approach a random graph is generated where each node represents a QID and there is an edge between two nodes whenever they have an overlap on at least one timestamp: e.g., let $QID(O_1) = \{t_1, t_2, t_3\}$ and $QID(O_2) = \{t_3, t_4, t_5\}$, then there is an edge between nodes $QID(O_1)$ and $QID(O_2)$.

D	$\|D\|$	x-range	y-range	t-range
Milan	45639	[45370K, 45560K]	[9050K, 9280K]	[7776K, 8380K]
Olden.	150000	[281, 23854]	[3935, 30851]	[0,399]

Table 1: Datasets used in the experiments.

6.3 Measures of quality

We next compare algorithms extreme union and symmetric anonymization both in terms of efficiency (run time and memory peak), and efficacy (quality of the resulting anonymized MOD). Since in our setting in general we can not produce a partition of objects as result of the k-anonymization, we can not adopt the usual measures adopted to evaluate the quality, such as for instance *discernibility* [3]. In the following we discuss the measure that we adopt.

Information loss. We measure directly the information loss introduced by our k-anonymization as defined in Section 3. In particular we report the average information loss. Given a distorted version D^* of a MOD D, we measure:

$$avg(IL(D, D^*)) = \frac{\Sigma_{i=1}^{n} \Sigma_{j=1}^{m} (1 - 1/area(D^*(O_i, t_j))}{n \times m}$$

where n is the number of MOBs and m the number of timestamps in D.

Coverage of equivalence classes. In microdata anonymization, an equivalence class larger than $2k - 1$ is meaningless: intuitively if the equivalence class population is at least of size $2k$, then it can further divided into at least two sub-classes still satisfying the k anonymity requirement and yielding less distortion. This consideration obviously does *not* hold in our MOD context for all the challenges that we discussed in this paper. In our context equivalence classes of size larger than $2k$ may be forced. Nevertheless, it is interesting to measure what proportion of the equivalence classes produced have a size under a given threshold. It is interesting to use $2k - 1$ as such a threshold. More precisely, we would be interested in measuring the percentage of the equivalence classes produced that have their size in the range $[k, 2k - 1]$.

Therefore, we define the *coverage* of equivalence classes as the ratio of the number of equivalence classes whose size is in the interval $[k, 2k - 1]$, over the total number of distinct equivalence classes. Obviously, the closer the coverage is to the value of 1, the better.

Range query distortion. Since the purpose of releasing data is usually to query or to analyze it, a very natural way of measuring utility is to compare the results between queries evaluated on the original dataset D and its anonymized version D^*. For this purpose we adopt *range query distortion* as a measure of the quality of the anonymization. In particular, given a region R and a timestamp t we consider two different kinds of queries:

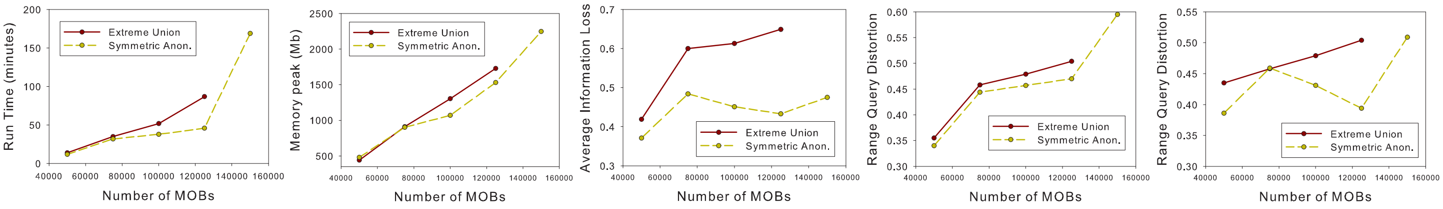
- *Possibly Inside*: returning the number of MOBs whose position (possibly generalized) at time t overlaps with R;
- *Definitely Inside*: returning the number of MOBs whose position (possibly generalized) at time t is completely contained in R.

We denote such queries $pi(R, t, D)$ and $di(R, t, D)$ respectively. In our experimentation we report the measures:

- $|pi(R, t, D) - pi(R, t, D^*)|/pi(R, t, D^*)$, and
- $|di(R, t, D) - di(R, t, D^*)|/di(R, t, D)$.

The two measures of distortion are always computed for 100 randomly chosen timestamps, and 100 randomly chosen regions, thus they are always averaged over 10.000 runs.

[3] http://www.geopkdd.eu



(a) (b) (c) (d) (e)

Figure 7: Algorithms comparison on dataset Oldenburg with $k = 16$, maximal QID size of 40, and varying size of the input database. In (a) we report run time, in (b) memory peak, in (c) average information loss, in (d) *"possibly inside"* and (e) *"definitely inside"* query distortion. Please note that the extreme union approach was not able to conclude for an input of 150.000 trajectories due to excessive memory requirement.

k	*avg* $(IL(D, D^*))$	Run time (minutes)	Memory peak (Mb)	Number of Eq.Cl.	Median of Eq.Cl. size	Coverage of Eq.Cl.	Possibly In. Distortion	Definitely In. Distortion
2	0.1359	621	1581.93	5014704	2	0.8969	0.1831	0.0829
4	0.2394	883	1645.67	2690394	5	0.7276	0.3771	0.2138
8	0.3795	1068	1743.94	495108	11	0.6614	0.5344	0.3752
16	0.5281	1337	1909.42	27549	20	0.6991	0.5786	0.5210
32	0.6396	1348	1994.50	2961	273	0.3637	0.5930	0.6218

(a)

Maximum QID size	*avg* $(IL(D, D^*))$	Run time (minutes)	Memory peak (Mb)	Number of Eq.Cl.	Median of Eq.Cl. size	Coverage of Eq.Cl.	Possibly In. Distortion	Definitely In. Distortion
20	0.0611	43	1025.16	1325	18	0.7969	0.0920	0.0713
40	0.1145	97	1052.29	5200	17	0.7878	0.1974	0.1485
100	0.3011	447	1233.42	16328	19	0.7443	0.3431	0.2700
200	0.5281	1337	1909.42	27549	20	0.6991	0.5786	0.5210

(b)

Figure 8: Effects of (a) the anonymity threshold k and (b) the QID size. All the experiments are on the Milan dataset, with the Symmetric Anonymization approach. In (a) the maximal QID size is set to 200, while in (b) the anonymization threshold is fixed as $k = 16$.

6.4 Algorithms analysis

As discussed in Section 5, theoretically extreme union and symmetric anonymization are incomparable in terms of the information loss they result in. But the experiments we conducted clearly show that the extreme union approach performs much more poorly than the symmetric anonymization, both in terms of efficiency and efficacy. Extreme union is always slower (Fig. 7(a)) and it has always larger memory requirements, becoming infeasible for large datasets (Fig. 7(b)). The difference of performance in the two algorithms is maintained in the quality tests were symmetric anonymization performs better both in terms of information loss (Fig. 7(c)) and range query distortion (Fig. 7(d)(e)). Since symmetric anonymization so evidently dominates extreme union in terms of both efficiency and efficacy, in the rest of this section we will focus mainly on the best approach.

We experimentally discovered that for symmetric anonymization, the quality of the data maintained after anonymization does not necessarily degrade with larger datasets (see the non monotonicity of the line in Fig. 7(c),(d), and(e)). On the contrary, the symmetric anonymization approach benefits from the larger variety of MOBs available. In fact, in larger data set, an object O_i has more candidate objects to hide with. So it is more likely that anonymization groups will not overlap among them, and consequently, symmetric anonymization will create small-sized equivalence classes. As an example, in the same set of experiments reported in Figure 7, in the dataset anonymized by symmetric anonymization, the median of equivalence classes size distribution is 29 when the input MOD consists of 50k trajectories, while it falls to 18 (just slightly above the anonymity threshold $k = 16$) for an input database of 150k MOBs. It is worth noting that even with such a drastic drop in the size of equivalence classes, the quality of the anonymized data does not improve accordingly. This is due to the way symmetric anonymization enforces complete symmetry: suppose that $QID(O_1) = \{t_1\}$ and $QID(O_2) = \{t_{200}\}$, and suppose that the top-k algorithm finds that $AG(O_1) = \{O_1, O_2\}$. Then $AG(O_2) = \{O_1, O_2\}$ by symmetry. But, O_1 may be arbitrarily far from O_2 at time t_{200}.

A deeper analysis of the symmetric anonymization approach on the real-world database is reported in Figure 8, where the effects of (a) the anonymity threshold k and (b) the QID size are studied. The first observation is that the median of the distribution of equivalence classes size increases quickly with k, while it seems not to be influenced by the size of the largest QID. Secondly the information loss grows with larger k and larger QIDs: this was certainly expected as larger k and larger QIDs clearly make for a more difficult anonymization task. Range query distortion measures, and coverage of equivalence classes confirm these trends. This observation is also reflected in run-time and memory consumption. However we can observe that the maximal size of the QIDs has a larger impact on the performances. While run-time grows sub-linearly with k it grows super-linearly with the maximal size of the QIDs.

7. CONCLUSIONS AND FUTURE WORK

We motivated the problem of k-anonymization of moving object databases for the purpose of their publication in a privacy-preserving way. In particular we focused on providing a form of k-anonymity based on spatial generalization. We showed the challenges in adapting the concept of k-anonymity to the case of moving object databases.

The main challenge in finding k-anonymization of a moving object database is that unlike in relational microdata, every MOB may have distinct QIDs, which may lead to overlapping anonymity sets.

We developed a notion of k-anonymity for MOD and we formally showed that it does not lead to privacy breaches. Based on this notion we devised two different anonymization approaches, namely *Extreme Union* and *Symmetric Anonymization*. We showed with extensive experiments on both real and synthetic data sets that the symmetric anonymization outperforms extreme union both in efficiency and quality of the anonymized data. To the best of our knowledge, this is the first work tackling the challenges of anonymizing MOD by means of space-generalization.

In future work, we intend to conduct tests on more data sets and w.r.t. different configurations of QIDs and an exhaustive range of values for the sizes of QIDs. An open problem is how to obtain real-world QIDs or, at least, how to produce QIDs which are as realistic as possible. From this perspective, an interesting line of research is how to effectively *discover* the QIDs: this may be tackled as a data mining problem, by identifying the sub-trajectories that uniquely identify the moving objects.

Another line of research that we intend to follow is how to devise anonymization methods that support database updates. Developing k-anonymity models for MOD based on stronger attack models is another interesting research topic.

Finally, we plan to investigate how to introduce outlier detection and suppression in our method. In fact, we believe that by removing a few outliers we may significantly enhance the overall data quality.

Acknowledgments: The authors wish to acknowledge the GeoPKDD project which supported part of this research. The work of the first and third authors was supported by a grant from NSERC (Canada). Thanks are also to the anonymous reviewers for numerous helpful comments which helped improve the paper.

8. REFERENCES

[1] Abul, O., Bonchi, F., and Nanni, M. $\mathcal{N}$ever $\mathcal{W}$alk $\mathcal{A}$lone: Uncertainty for anonymity in moving objects databases. In *Proc. of the 24th IEEE Int. Conf. on Data Engineering (ICDE'08)*.

[2] Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., and Zhu, A. Anonymizing tables. In *Proc. of the 10th Int. Conf. on Database Theory (ICDT'05)*.

[3] Bayardo, R., and Agrawal, R. Data privacy through optimal k-anonymity. In *Proc. of the 21st IEEE Int. Conf. on Data Engineering (ICDE'05)*.

[4] Bettini, C., Wang, X. S., and Jajodia, S. Protecting Privacy Against Location-Based Personal Identification. In *Proc. of the Second VLDB Workshop on Secure Data Management (SDM'05)*.

[5] Brinkhoff, T. Generating traffic data. *IEEE Data Eng. Bull. 26*, 2 (2003), 19–25.

[6] Galil, Z., and Italiano, G. F. Data structures and algorithms for disjoint set union problems. *ACM Comput. Surv. 23*, 3 (1991), 319–344.

[7] Gruteser, M., and Grunwald, D. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proc. of the First Int. Conf. on Mobile Systems, Applications, and Services (MobiSys 2003)*.

[8] Hamilton, C. Compact Hilbert Indices. Tech. Rep. CS-2006-07, Dalhousie University, July 2006.

[9] Hilbert, D. Über die stetige abbildung einer linie auf ein flächenstück. *Math. Ann. 38* (1891), 459–460.

[10] LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. Mondrian multidimensional k-anonymity. In *Proc. of the 22nd IEEE Int. Conf. on Data Engineering (ICDE'06)*.

[11] Machanavajjhala, A., Gehrke, J., Kifer, D., and Venkitasubramaniam, M. l-diversity: privacy beyond k-anonymity. In *Proc. of the 22nd IEEE Int. Conf. on Data Engineering (ICDE'06)*.

[12] Meyerson, A., and Willliams, R. On the complexity of optimal k-anonymity. In *Proc. of the 23rd ACM Symp. on Principles of Database Systems (PODS'04)*.

[13] Mokbel, M. F., Chow, C.-Y., and Aref, W. G. The new casper: A privacy-aware location-based database server. In *Proc. of the 23rd IEEE Int. Conf. on Data Engineering (ICDE'07)*.

[14] Mokbel, M. F., Chow, C.-Y., and Aref, W. G. The new casper: Query processing for location services without compromising privacy. In *Proc. of the 32nd Int. Conf. on Very Large Databases (VLDB'06)*.

[15] N. Roussopoulos, S. Kelley, F. V. Nearest neighbor queries. In *Proc. of the 1995 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'95)*.

[16] R. Fagin, A. L., and Naor, M. Optimal aggregation algorithms for middleware. In *Proc. of the 20th ACM Symp. on Principles of Database Systems (PODS'01)*.

[17] R. Fagin, A. L., and Naor, M. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences 66*, 1 (2003), 614–656.

[18] Reza Akbarinia, Esther Pacitti, P. V. Best Position Algorithms for Top-k Queries. In *Proc. of the 32nd Int. Conf. on Very Large Databases (VLDB'07)*.

[19] Samarati, P., and Sweeney, L. Generalizing data to provide anonymity when disclosing information (abstract). In *Proc. of the 17th ACM Symp. on Principles of Database Systems (PODS'98)*.

[20] S.Saltenis, C.S., J. S. T. L. M. Indexing the positions of continuously moving objects. In *Proc. of the 2000 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'00)*.

[21] Terrovitis, M., and Mamoulis, N. Privacy preservation in the publication of trajectories. In *Proc. of the 9th Int. Conf. on Mobile Data Management (MDM'08)*.

[22] Xiaohui Yu, Ken Q. Pu, N. K. Monitoring k-nearest neighbor queries over moving objects. In *Proc. of the 21st IEEE Int. Conf. on Data Engineering (ICDE'05)*.