TRANSACTIONS ON DATA PRIVACY 7 (2014) 165–194

Apriori-based algorithms for k^m-anonymizing trajectory data

Giorgos Poulis*, **Spiros Skiadopoulos***, **Grigorios Loukides****, **Aris Gkoulalas-Divanis*****

*University of Peloponnese, Email: {poulis,spiros}@uop.gr

**Cardiff University, Email: g.loukides@cs.cf.ac.uk

***IBM Research - Ireland, Email: arisdiva@ie.ibm.com

Abstract. The proliferation of GPS-enabled devices (e.g., smartphones and tablets) and location-based social networks has resulted in the abundance of trajectory data. The publication of such data opens up new directions in analyzing, studying and understanding human behavior. However, it should be performed in a privacy-preserving way, because the identities of individuals, whose movement is recorded in trajectories, can be disclosed even after removing identifying information. Existing trajectory data anonymization approaches offer privacy but at a high data utility cost, since they either do not produce truthful data (an important requirement of several applications), or are limited in their privacy specification component. In this work, we propose a novel approach that overcomes these shortcomings by adapting k^m-anonymity to trajectory data. To realize our approach, we develop three efficient and effective anonymization algorithms that are based on the apriori principle. These algorithms aim at preserving different data characteristics, including location distance and semantic similarity, as well as user-specified utility requirements, which must be satisfied to ensure that the released data can be meaningfully analyzed. Our extensive experiments using synthetic and real datasets verify that the proposed algorithms are efficient and effective at preserving data utility.

Keywords. privacy, anonymity, trajectories, spatial data, k^m-anonymity, utility constraints

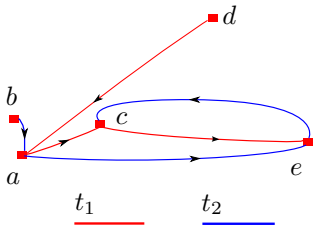
1 Introduction

The widespread adoption of GPS-enabled smartphones and location-based social networking applications, such as Foursquare (https://foursquare.com), opens up new opportunities in understanding human behaviour through the analysis of collected mobility data. However, the publication of these data, which correspond to trajectories of personal movement (i.e., ordered lists of locations visited by individuals), can lead to *identity disclosure*, even if directly identifying information, such as names or SSN of individuals, is not published [33].

The values that, in combination, may lead to identity disclosure are called *quasi-identifiers* (QI) [32, 34]. For example, let us assume that a location-based social network service publishes the movement of users during a day in the form of checkins in various locations. An example of these data is shown in Figure 1a. If Mary's colleague, John, knows that Mary

id	trajectory
t_1	(d, a, c, e)
t_2	(b, a, e, c)
t_3	(a, d, e)
t_4	(b, d, e, c)
t_5	(d, c)
t_6	(d, e)

(a)



(b)

Figure 1: (a) the original database $\mathcal{T}$ (b) visual representation of trajectories t_1 and t_2

checked in at locations a and d, he cannot associate Mary with her record (trajectory), as both trajectories t_1 and t_3 include the locations a and d. But if John knew that Mary first checked in at location d and then at a, he can uniquely associate Mary with the trajectory t_1.

This example highlights not only the need to transform a set of user trajectories $\mathcal{T}$ to prevent identity disclosure based on partial location knowledge held by attackers, but also the difference from well-studied set-valued data anonymity models, like k^m-anonymity [36] and privacy-constrained anonymization [18, 24]. In these models, value ordering is not significant; thus, records are represented as unordered sets of items. For instance, if an attacker knows that someone checked in first at the location c and then at e, they could uniquely associate this individual with the record t_1 (Figure 1b). On the other hand, if $\mathcal{T}$ was a set-valued dataset, three records, namely t_1, t_2 and t_4, would have the items c and e. Thus, the individual's identity is "hidden" among 3 records. Consequently, for any set of n items in a trajectory, there are $n!$ possible quasi-identifiers.

This difference makes preventing identity disclosure in trajectory data publishing more challenging, as the number of potential quasi-identifiers is drastically increased. Existing methods operate either by anonymizing *(i)* each trajectory as a whole, thereby not assuming any specific background knowledge of attackers [1, 2, 26, 29], or *(ii)* parts of trajectories, thereby considering attackers who aim to re-identify individuals based on specific locations [35, 39]. The first category of methods are based on *clustering and perturbation* [1, 2, 29], while the second category employs *generalization and suppression* of quasi-identifiers [27, 39, 28, 35]. The main drawback of clustering-based methods is that they may lose information about the direction of movement of co-clustered trajectories and cause excessive information loss, due to space translation. Moreover, applying perturbation may create data that are not *truthful* and cannot be used in several applications [14]. Similarly, existing generalization-and-suppression based methods [27, 39, 28, 35] have the following limitations. First, they assume that quasi-identifiers are known to the data publisher prior to anonymization [35, 39], or that any combination of locations can act as a quasi-identifier [27]. Second, they require a location taxonomy to be specified by data publishers [28] based on location semantics. However, such a taxonomy may not exist, or may not accurately reflect the distance between locations. In both cases, the anonymized data will be highly distorted. Last, some approaches assume that each location can be classified as either sensitive or non-sensitive [28]. In practice, however, this assumption may not hold, as location sensitivity usually depends on context (e.g., visiting a hospital may be considered as sensitive for a patient, but not for a doctor).

Recently, another class of approaches that aims at limiting the amount of information about the presence or absence of any individual trajectory has been proposed [5, 7, 9]. These approaches enforce a well-established privacy model, called *differential privacy* [12],

by employing *perturbation*. Specifically, they release a noisy summary of the original data that can be used in specific analytic tasks, such as frequent sequential pattern mining [3]. While being able to offer strong privacy guarantees, these approaches do not preserve data truthfulness, since they rely on perturbation.

1.1 Contributions

In this work, we propose a novel approach for publishing trajectory data, in a way that prevents identity disclosure, and three effective and efficient algorithms to realize it. Specifically, our work makes the following contributions.

First, we adapt k^m-anonymity [35, 36] to trajectory data. k^m-anonymity is a privacy model that was proposed to limit the probability of identity disclosure in transaction data publishing. The benefit of this model is that it does not require detailed knowledge of quasi-identifiers, or a distinction between sensitive and non-sensitive information, prior to data publishing.

Second, we develop three algorithms for enforcing k^m-anonymity on trajectory data. These algorithms generalize data in an apriori-like fashion (i.e., apply generalization to increasingly larger parts of trajectories) and aim at preserving different aspects of data utility. Our first algorithm, called SEQANON, applies *distance-based* generalization, effectively creating generalized trajectories with locations that are close in proximity. For instance, SEQANON would favor generalizing a together with b, because b is the closest location to a, as can be seen in Fig. 1b. SEQANON does not require a location taxonomy and aims at preserving the distance between original locations. Thus, it should be used when accurate semantic information about locations is not available[1]. Clearly, however, the presence of accurate, semantic location information should also be exploited, as it can help the preservation of data utility. For example, assume that a and c represent the locations of restaurants, whereas b represents the location of a coffee shop. In this case, generalizing a together with c would be preferred, because c is a restaurant that is also not very far from a. To take into account both the distance and the semantic similarity of locations, we propose an algorithm, called SD-SEQANON. This algorithm produces generalized trajectories, whose locations are typically slightly more distant but much more semantically similar than those created by SEQANON. Both SEQANON and SD-SEQANON allow generalizing any locations together, as they aim to minimize information loss. In several applications, however, data publishers have specific utility requirements, which dictate how locations must be generalized to ensure that the anonymized dataset is practically useful [24]. For instance, assume that the anonymized version of the dataset in Fig. 1a needs to be used to enable the accurate counting of the number of restaurants, in which individuals checked in. To satisfy this requirement, generalizing together a and b must be prohibited, because the resultant generalized location $\{a, b\}$ can be interpreted as either a restaurant or a coffee shop. On the other hand, the generalization of a together with any other restaurant is allowable, and the generalization that incurs the minimum information loss should be preferred. To account for such utility requirements, we propose a third algorithm, called U-SEQANON. This algorithm aims at satisfying utility constraints and uses both generalization and suppression.

Third, we investigate the effectiveness and efficiency of our approach through experiments on a synthetic dataset, generated using the Brinkhoff's generator [6], and on a real dataset, derived from a location-based social networking website [10]. The results of these

[1] A preliminary version of this work that discusses the SEQANON algorithm appeared in the PriSMO workshop, which was held in conjunction with IEEE MDM 2013.

experiments verify that our approach is able to anonymize trajectory data, under various privacy and utility requirements, with a low level of information loss. In addition, they show that our algorithms are fast and scalable, due to the use of the apriori principle.

1.2 Organization

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents some preliminary concepts related to trajectory data anonymization, as well as the privacy and utility objectives of our algorithms. Section 4 presents our anonymization algorithms, and Section 5 an experimental evaluation of them. Last, we conclude the paper in Section 6.

2 Related work

Privacy-preserving trajectory data publishing has attracted significant attention, due to the pervasive use of location-aware devices and location-based social networks, which led to a tremendous increase in the volume of collected data about individuals [4]. One of the main concerns in trajectory data publishing is the prevention of identity disclosure, which is the objective of the k-anonymity privacy model [33, 34]. k-anonymity prevents identity disclosure by requiring at least k records of a dataset to have the same values over QI. Thus, a k-anonymous dataset upperbounds the probability of associating an individual with their record by $\frac{1}{k}$. To enforce k-anonymity most works [15, 20, 21, 23, 32] employ *generalization*, which replaces a QI value with a more general but semantically consistent value, or *suppression*, which removes QI values prior to data publishing.

k-anonymity has been considered in the context of publishing user trajectories, leading to several trajectory anonymization methods [4]. As mentioned in Section 1, these methods operate by anonymizing either entire trajectories [1, 2, 26], or parts of trajectories (i.e., sequences of locations) that may lead to identity disclosure [35, 39]. In the following, we discuss the main categories of trajectory anonymization works, as well as how our approach differs from them.

2.1 Clustering and perturbation

Methods based on clustering and perturbation are applied to time-stamped trajectories. They operate by grouping original trajectories into clusters (cylindrical tubes) of at least k trajectories, in a way that each trajectory within a cluster becomes indistinguishable from the other trajectories in the cluster. One such method, called *NWA* [1], enforces (k, δ)-anonymity to anonymize user trajectories by generating cylindrical volumes of radius δ that contain at least k trajectories. Each trajectory that belongs to an anonymity group (*cylinder*), generated by NWA, is protected from identity disclosure, due to the other trajectories that appear in the same group. To produce the cylindrical volumes, the algorithm in [1] identifies trajectories that lie close to each other in time and employs space translation. Trujillo-Rasua and Domingo-Ferrer [37] performed a rigorous analysis of the (k, δ)-anonymity model, which shows that this model is not able to hide an original trajectory within a set of k-indistinguishable, anonymized trajectories. Thus, the algorithms in [1, 2] may not provide meaningful privacy guarantees, in practice. An effective algorithm for enforcing k-anonymity on trajectory data was recently proposed by Domingo-Ferrer et al. [11]. The algorithm, called *SwapLocations*, creates trajectory clusters using *microaggregation*

and then permutes the locations in each cluster to enforce privacy. The experimental evaluation in [11] demonstrates that *SwapLocations* is significantly more effective at preserving data utility than *NWA* [1]. Finally, Lin et al. [22] guarantees k-anonymity of published data, under the assumption that road-network data are public information. Their method uses clustering-based anonymization, protecting from identity disclosure.

Contrary to the methods of [1, 2, 11, 22], our work *(a)* does not consider time-stamped trajectories, and *(b)* applies generalization to derive an anonymized dataset.

2.2 Generalization and suppression

Differently to the methods of Section 2.1, this category of methods considers attackers with background knowledge on ordered sequences of places of interest (POIs) visited by specific individuals. Terrovitis et al. [35] proposed an approach to prohibit multiple attackers, each knowing a different set of POIs, from associating these POIs to fewer than k individuals in the published dataset. To achieve this, the authors developed a suppression-based method that aims at removing the least number of POIs from user trajectories, so that the remaining trajectories are k-anonymous with respect to the knowledge of each attacker.

Yarovoy et al. [39] proposed a k-anonymity based approach for publishing user trajectories by considering time as a quasi-identifier and supporting privacy personalization. Unlike previous approaches that assumed that all users share a common quasi-identifier, [39] assumes that each user has a different set of POIs and times requiring protection, thereby enabling each trajectory to be protected differently. To achieve k-anonymity, this approach uses generalization and creates anonymization groups that are not necessarily disjoint.

A recent approach, proposed by Monreale et al. [27], extends the l-diversity principle to trajectories by assuming that each location is either nonsensitive (acting as a QI) or sensitive. This approach applies c-safety to prevent attackers from linking sensitive locations to trajectories with a probability greater than c. To enforce c-safety, the proposed algorithm applies generalization to replace original POIs with generalized ones based on a location taxonomy. If generalization alone cannot enforce c-safety, suppression is used.

Assuming that each record in a dataset is comprised of a user's trajectory and user's sensitive attributes, Chen et al. [8] propose the $(K, C)_L$-privacy model. This model protects from identity and attribute linkage by employing local suppression. In this paper, the authors assume that an adversary knows at most L locations of a user's trajectory. Their model guarantees that a user is indistinguishable from at least $K - 1$ users, while the probability of linking a user to his/her sensitive values is at most C.

Contrary to the methods of [8, 27, 35, 39], our work *(a)* assumes that an attacker may know up to m user locations, which is a realistic assumption in many applications, and *(b)* does not classify locations as sensitive or nonsensitive, which may be difficult in some domains [36].

2.3 Differential privacy

Recently, methods for enforcing differential privacy [12] on trajectory data have been proposed [5, 7, 9]. The objective of these methods is to release noisy data summaries that are effective at supporting specific data analytic tasks, such as count query answering [7, 9] and frequent pattern mining [5]. To achieve this, the method in [9] uses a *context-free*, *taxonomy* tree, for identifying the set of counting queries that should be supported by the noisy summary, while the method in [5] employs a prefix-tree to generate candidate patterns, used in the construction of the data summary.

The method proposed in [7] was shown to be able to generate summaries that permit highly accurate count query answering. This method, referred to as NGRAMS, works in three steps. First, it truncates the original trajectory dataset by keeping only the first ℓ_{max} locations of each trajectory, where ℓ_{max} is a parameter specified by data publishers. Larger ℓ_{max} values improve efficiency but deteriorate the quality of the frequencies, calculated during the next step. Second, it uses the truncated dataset to compute the frequency of *n-grams* (i.e., all possible contiguous parts of trajectories that are comprised of 1, or 2, ... , or n locations). Third, this method constructs a differentially private summary by inserting calibrated Laplace noise [12] to the frequencies of n-grams.

Contrary to the methods of [5, 7, 9], our work publishes truthful data at a record (individual user) level, which is required by many data analysis tasks [24]. That is, our work retains the number of locations in each published trajectory and the number of published trajectories in the anonymized dataset. Furthermore, our method is able to preserve data utility significantly better than these methods, as shown in our extensive experiments. Thus, our approach can be used to offer a better privacy/utility trade-off than the methods of [5, 7, 9].

3 Privacy and utility objectives

In this section, we first define some preliminary concepts that are necessary to present our approach, and then discuss the privacy and utility objectives of our anonymization algorithms.

3.1 Preliminaries

Let $\mathcal{L}$ be a set of locations (e.g., points of interest, touristic sites, shops). A trajectory represents one or more locations in $\mathcal{L}$ and the order in which these locations are visited by a moving object (e.g., individual, bus, taxi), as explained in the following definition.

Definition 1. *A* trajectory t *is an ordered list of locations* $(l_1, \ldots, l_n)$*, where* $l_i \in \mathcal{L}$*,* $1 \leq i \leq n$*. The* size *of the trajectory* $t = (l_1, \ldots, l_n)$*, denoted by* $|t|$*, is the number of its locations, i.e.,* $|t| = n$*.*

Note that, in our setting, a location may model points in space. A part of a trajectory, which is formed by removing some locations while maintaining the order of the remaining locations, is a *subtrajectory* of the trajectory, as explained below.

Definition 2. *A trajectory* $s = (\lambda_1, \ldots, \lambda_\nu)$ *is a* subtrajectory of *or is* contained in *trajectory* $t = (l_1, \ldots, l_n)$*, denoted by* $s \sqsubseteq t$*, if and only if* $|s| \leq |t|$ *and there is a mapping* f *such that* $\lambda_1 = l_{f(1)}, \ldots, \lambda_\nu = l_{f(\nu)}$ *and* $f(1) < \cdots < f(\nu)$*.*

For instance, the trajectory (a, e) is a subtrajectory of (or contained in) the trajectory $t_1 = (d, a, c, e)$ in Figure 1. Clearly, (a, e) can be obtained from t_1 by removing d and c.

Definition 3. *Given a set of trajectories* $\mathcal{T}$*, the* support *of a subtrajectory* s*, denoted by* $\sup(s, \mathcal{T})$*, is defined as the number of distinct trajectories in* $\mathcal{T}$ *that contain* s*.*

In other words, the support of a subtrajectory s measures the number of trajectories in a dataset that s is contained in. For example, for the dataset in Figure 1a, we have $\sup((a, e), \mathcal{T}) = 3$. Note that the support does not increase when a subtrajectory is contained multiple times in a trajectory. For instance, $\sup((a, e), \{(a, e, b, a, e)\}) = 1$. Naturally, by considering locations as unary trajectories, the support can also be measured for the locations of a dataset.

In this work, we adapt the notion of k^m-anonymity [35, 36] to trajectory data, as explained below.

Definition 4. *A set of trajectories $\mathcal{T}$ is* k^m-anonymous *if and only if every subtrajectory s of every trajectory $t \in \mathcal{T}$, which contains m or fewer locations (i.e., $|s| \leq m$), is contained in at least k distinct trajectories of $\mathcal{T}$.*

Definition 4 ensures that an attacker who knows any subtrajectory s of size m of an individual, cannot associate the individual to fewer than k trajectories (i.e., the probability of identity disclosure, based on s, is at most $\frac{1}{k}$). The privacy parameters k and m are specified by data publishers, according to their expectations about adversarial background knowledge, or certain data privacy policies [18, 35, 36].

The following example illustrates a dataset that satisfies k^m-anonymity.

Example 1. *Consider the trajectory dataset that is shown in Figure 1a. This dataset is 2^1-anonymous, because every location (i.e., subtrajectory of size 1) appears at least 2 times in it. This dataset is also 1^3-anonymous, because every subtrajectory of size 3 appears only once in it. However, the dataset is not 2^2-anonymous, as the subtrajectory (d, a) is contained only in the trajectory t_1.*

Note that, unlike k-anonymity, the k^m-anonymity model assumes that an attacker possesses background knowledge about subtrajectories, which are comprised of at most m locations. That is, an attacker knows at most m locations that are visited by an individual, in a certain order. Clearly, m can be set to any integer in $[0, \max\{|t| \mid t \in \mathcal{T}\}]$. Setting m to 0 corresponds to the trivial case, in which an attacker has no background knowledge. On the other hand, setting m to $\max\{|t| \mid t \in \mathcal{T}\}$, can be used to guard against an attacker who knows the maximum possible subtrajectory about an individual (i.e., that an individual has visited all the locations in their trajectory, and the order in which they visited these locations). In this case, k^m-anonymity "approximates" k-anonymity, but it does not provide the same protection guarantees against identity disclosure. This is because k^m-anonymity does not guarantee protection from attackers who know that an individual has visited *exactly* the locations, contained in a subtrajectory of size m. For example, assume that a dataset is comprised of the trajectories $\{(a, d, e), (a, d, e), (a, d)\}$. The dataset satisfies 2^3-anonymity, hence it prevents an attacker from associating an individual with any of the subtrajectories (a, d), (a, e), and (d, e). However, the dataset is not 2-anonymous, hence an attacker who knows that an individual has visited exactly the locations a and d, in this order, can uniquely associate the individual with the trajectory (a, d).

The k^m-anonymity model is practical in several applications, in which it is extremely difficult for attackers to learn a very large number of user locations [35]. However, k^m-anonymity does not guarantee that all possible attacks, based on background knowledge, will be prevented. For example, k^m-anonymity is not designed to prevent *collaborative attacks*, in which *(i)* two or more attackers combine their knowledge in order to re-identify an individual, or *(ii)* an attacker possesses background knowledge about multiple trajectories in $\mathcal{T}$. Such powerful attack schemes can only be handled within stronger privacy principles, such as differential privacy (see Section 2). However, applying these principles usually results in significantly lower data utility, compared to the output of our algorithms, as shown in our experiments. In addition, as we do not deal with time-stamped trajectories, time information is not part of our privacy model. In the case of time-stamped trajectory data publishing, time information can be used by attackers to perform identity disclosure, and privacy models to prevent this are the focus of [8, 39]. For the same reason, we do not deal with attacks that are based on both time and road-network information (e.g., the *inference route problem* [22]). These attacks can be thwarted using privacy models, such as *strict* k-anonymity [22].

To explain the way we generalize trajectories, we define the notion of *generalized location*, as explained below.

Definition 5. *A* generalized location $\{l_1, \ldots, l_v\}$ *is defined as a set of at least two locations* $l_1, \ldots, l_v \in \mathcal{L}$.

Thus, a generalized location is the result of replacing at least two locations in $\mathcal{L}$ with their set. A *generalized location* is interpreted as exactly one of the locations mapped to it. For example, the generalized location $\{a, b\}$ may be used to replace the locations a and b in Figure 1a. This generalized location will be interpreted as a or b. Therefore, if a trajectory t' in an anonymized version $\mathcal{T}'$ of $\mathcal{T}$ contains a generalized location $\{l_1, \ldots, l_v\}$, then the trajectory t in $\mathcal{T}$ contains exactly one of the locations $l_1, \ldots, l_v$.

To enforce k^m-anonymity, we either replace a location l with a generalized location that contains l, or leave l intact. Thus, a *generalized trajectory* t' is an ordered list of locations and/or generalized locations. The *size* of t', denoted by $|t'|$, is the number of elements of t'. For instance, a generalized trajectory $t' = (\{a, b\}, c)$ is comprised of one generalized location $\{a, b\}$ and a location c, and it has a size of 2.

We are interested in generalization transformations that produce a transformed dataset $\mathcal{T}'$ by distorting the original dataset $\mathcal{T}$ as little as possible, A common way to measure the distortion of a transformation is to measure the *distance* between the original and the transformed dataset [29, 35, 39]. In our case, the distance between the original and the anonymized dataset is defined as the *average* of the distances of their corresponding trajectories. In turn, the distance between the initial and the anonymized trajectory is defined as the *average* of the distance between their corresponding locations.

The following definition illustrates how the distance between locations, trajectories, and datasets $\mathcal{T}$ and $\mathcal{T}'$ can be computed.

Definition 6. *Let* l *be a location that will be generalized to the generalized location* $\{l_1, \ldots, l_v\}$. *The* location distance *between* l *and* $\{l_1, \ldots, l_v\}$, *denoted by* $\mathcal{D}_{loc}(l, \{l_1, \ldots, l_v\})$, *is defined as:*

$$\mathcal{D}_{loc}(l, \{l_1, \ldots, l_v\}) = \text{avg}\big\{ d(l, l_i) \mid 1 \leq i \leq v \big\}$$

where d *is the Euclidean distance. The* trajectory distance *between* $t = (l_1, \ldots, l_n)$ *and its generalized counterpart* $t' = (l'_1, \ldots, l'_n)$, *denoted by* $\mathcal{D}_{traj}(t, t')$, *is defined as:*

$$\mathcal{D}_{traj}(t, t') = \text{avg}\big\{ \mathcal{D}_{loc}(l_i, l'_i) \mid 1 \leq i \leq n \big\}$$

Finally, the trajectory dataset distance *between* $\mathcal{T} = \{t_1, \ldots, t_u\}$ *and its generalized counterpart* $\mathcal{T}' = \{t'_1, \ldots, t'_u\}$ *(where the trajectory* t_i *is generalized to trajectory* t'_i, $1 \leq i \leq u$*), denoted by* $\mathcal{D}(\mathcal{T}, \mathcal{T}')$, *is defined as:*

$$\mathcal{D}(\mathcal{T}, \mathcal{T}') = \text{avg}\big\{ \mathcal{D}_{traj}(t_i, t'_i) \mid 1 \leq i \leq u \big\}$$

For example, let a, a_1, a_2 and b be locations and let $d(a, a_1) = 1$ and $d(a, a_2) = 2$. If location a is generalized to the generalized location $\{a, a_1, a_2\}$ the location distance $\mathcal{D}_{loc}(a, \{a, a_1, a_2\}) = (0 + 1 + 2)/3 = 1$. Also, if trajectory (a, b) is generalized to $(\{a, a_1, a_2\}, b)$ the trajectory distance $\mathcal{D}_{traj}((a, b), (\{a, a_1, a_2\}, b)) = (1 + 0)/2 = 1/2$.

Note that the distances in Definition 6 can be normalized by dividing each of them with the maximum distance between locations in $\mathcal{T}$.

3.2 Problem statement

As discussed in Introduction, the objective of our approach is to enforce k^m-anonymity to a trajectory dataset, while preserving data utility. However, there are different aspects

of data utility that data publishers may want to preserve. To account for this, we have developed three anonymization algorithms, namely SEQANON, SD-SEQANON, and U-SEQANON, which generalize locations in different ways.

The SEQANON algorithm aims at generalizing together locations that are close in proximity. The distance between locations, in this case, is expressed based on Definition 6. Thus, the problem that SEQANON aims to solve can be formalized as follows.

Problem 1. *Given an original trajectory dataset $\mathcal{T}$, construct a k^m-anonymous version $\mathcal{T}'$ of $\mathcal{T}$ such that $\mathcal{D}(\mathcal{T}, \mathcal{T}')$ is minimized.*

Note that Problem 1 is NP-hard (the proof follows easily from observing that Problem 1 contains the NP-hard problem in [35] as a special case), and that SD-SEQANON is a heuristic algorithm that may not find an optimal solution to this problem.

The SD-SEQANON algorithm considers both the distance and the semantic similarity of locations, when constructing generalized locations. Thus, it exploits the availability of semantic information about locations to better preserve data utility. Following [27], we assume that the semantic information of locations is provided by data publishers, using a location taxonomy. The leaf-level nodes in the taxonomy correspond to each of the locations of the original dataset, while the non-leaf nodes represent more general (abstract) location information.

Given a location taxonomy, we define the notion of *semantic dissimilarity* for a generalized location, as explained in the following definition. A similar notion of semantic dissimilarity, for relational values, was proposed in [38].

Definition 7. *Let $l' = \{l_1, \ldots, l_v\}$ be a generalized location and $\mathcal{H}$ be a location taxonomy. The* semantic dissimilarity *for l' is defined as:*

$$\mathcal{SD}(l') = \frac{CCA(\{l_1, \ldots, l_v\})}{|\mathcal{H}|}$$

where $CCA(\{l_1, \ldots, l_j\})$ is the number of leaf-level nodes in the subtree rooted at the closest common ancestor of the locations $\{l_1, \ldots, l_v\}$ in the location taxonomy $\mathcal{H}$, and $|\mathcal{H}|$ is the total number of leaf-level nodes in $\mathcal{H}$.

Thus, locations that belong to subtrees with a small number of leaves are more semantically similar. Clearly, the $\mathcal{SD}$ scores for generalized locations that contain more semantically similar locations are lower.

Example 2. *An example location taxonomy is illustrated in Figure 2. The leaf-level nodes a to e represent the locations (i.e., specific restaurants and coffee houses), while the non-leaf nodes represent the general concepts* Restaurants *and* Coffee shops*. We also have $CCA(\{a, c, e\}) = 3$, as the subtree rooted at* Restaurants *has three leaf-level nodes, and $|\mathcal{H}| = 5$, as the taxonomy in Figure 2 has 5 leaf-level nodes. Thus, the semantic dissimilarity for the generalized location $\{a, c, e\}$, denoted with $\mathcal{SD}(\{a, c, e\})$, is $\frac{3}{5} = 0.6$. Similarly, we can compute $SD(\{a, d\}) = \frac{5}{5} = 1$, which is greater than $SD(\{a, c, e\})$ because a and d are more semantically dissimilar (i.e., restaurants and coffee shops, instead of just restaurants).*

We now define the criteria that are used by the SD-SEQANON algorithm to capture both the distance and semantic similarity between locations, trajectories, and datasets $\mathcal{T}$ and $\mathcal{T}'$. The computation of these criteria is similar to those in Definition 6.

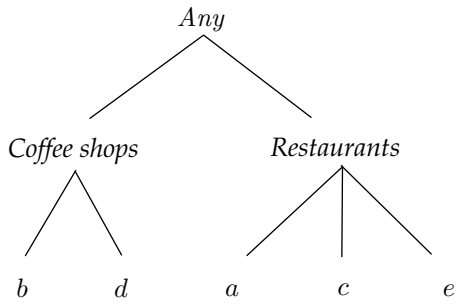


Figure 2: A location taxonomy

Definition 8. *Let l be a location that will be generalized to $l' = \{l_1, \ldots, l_v\}$. The* combined location distance *between l and l', denoted by $\mathcal{C}_{loc}(l, l')$, is defined as:*

$$\mathcal{C}_{loc}(l, l') = \text{avg}\big\{ d(l, l_i) \cdot SD(l') \mid 1 \leq i \leq v \big\}, \text{where } SD(l') \text{ takes values in } (0, 1]$$

where d is the Euclidean distance and $\mathcal{SD}$ is the semantic dissimilarity. Note that the above formula is a conventional weighted-formula, where similarity and distance are combined into the $\mathcal{C}_{loc}(l, l')$. Thus, the combined objective is then optimized by the single-objective optimization metric $\mathcal{C}_{loc}(l, l')$. Using conventional weighted-formulas is an effective approach for addressing multi-objective optimization problems, as discussed in [13]. The combined trajectory distance *between $t = (l_1, \ldots, l_n)$ and its generalized counterpart $t' = (l'_1, \ldots, l'_n)$, denoted by $\mathcal{C}_{traj}(t, t')$, is defined as:*

$$\mathcal{C}_{traj}(t, t') = \text{avg}\big\{ \mathcal{C}_{loc}(l_i, l'_i) \mid 1 \leq i \leq n \big\}$$

Finally, the combined trajectory dataset distance *between $\mathcal{T} = \{t_1, \ldots, t_u\}$ and its generalized counterpart $\mathcal{T}' = \{t'_1, \ldots, t'_u\}$ (where the trajectory t_i is generalized to trajectory t'_i, $1 \leq i \leq u$), denoted by $\mathcal{C}(\mathcal{T}, \mathcal{T}')$, is defined as:*

$$\mathcal{C}(\mathcal{T}, \mathcal{T}') = \text{avg}\big\{ \mathcal{C}_{traj}(t_i, t'_i) \mid 1 \leq i \leq u \big\}$$

We now define the problem that the SD-SEQANON algorithm aims to solve, as follows.

Problem 2. *Given an original trajectory dataset $\mathcal{T}$, construct a k^m-anonymous version $\mathcal{T}'$ of $\mathcal{T}$ such that $\mathcal{C}(\mathcal{T}, \mathcal{T}')$ is minimized.*

Note that Problem 2 can be restricted to Problem 1, by allowing only instances where $SD(l') = 1$, for each generalized location l' that is contained in a trajectory of $\mathcal{T}'$. Thus, Problem 2 is also NP-hard. The SD-SEQANON algorithm aims to derive a (possibly sub-optimal) solution to Problem 2 by taking into account both the distance and the semantic similarity of locations, when constructing generalized locations.

However, in several applications, there are specific utility requirements that must be taken into account to ensure that the published dataset is practically useful. In what follows, we explain our notion of utility constraints, which is used to capture the utility requirements of data publishers, and explain when the utility constraints are satisfied. Subsequently, we discuss the practical importance of utility constraints in applications. Our definitions are similar to those proposed in [24] for transaction data.

utility constraints	*locations*
u_1	$\{b, c\}$
u_2	$\{a, d, e\}$

(a)

id	trajectory
t'_1	$(d, \{a, b, c\}, \{a, b, c\}, e)$
t'_2	$(\{a, b, c\}, \{a, b, c\}, e, \{a, b, c\})$
t'_3	$(\{a, b, c\}, d, e)$
t'_4	$(\{a, b, c\}, d, e, \{a, b, c\})$
t'_5	$(d, \{a, b, c\})$
t'_6	(d, e)

(b)

id	trajectory
t'_1	$(\{a, d, e\}, \{a, d, e\}, \{b, c\}, \{a, d, e\})$
t'_2	$(\{b, c\}, \{a, d, e\}, \{a, d, e\}, \{b, c\})$
t'_3	$(\{a, d, e\}, \{a, d, e\}, \{a, d, e\})$
t'_4	$(\{b, c\}, \{a, d, e\}, \{a, d, e\}, \{b, c\})$
t'_5	$(\{a, d, e\}, \{b, c\})$
t'_6	$(\{a, d, e\}, \{a, d, e\})$

(c)

Figure 3: *(a)* An example utility constraint set $\mathcal{U}$. *(b)* A 2^2-anonymous dataset $\mathcal{T}'$ that does not satisfy the utility constraint set $\mathcal{U}$. *(c)* A 2^2-anonymous dataset $\mathcal{T}'$ that satisfies $\mathcal{U}$.

Definition 9. *A* utility constraint u *is a set of locations* $\{l_1, \ldots, l_v\}$*, specified by data publishers. A* utility constraint set $\mathcal{U} = \{u_1, \ldots, u_p\}$ *is a partition of the set of locations* $\mathcal{L}$*, which contains all the specified utility constraints* $u_1, \ldots, u_p$*.*

Definition 10. *Given a utility constraint set* $\mathcal{U} = \{u_1, \ldots, u_p\}$*, a generalized dataset* $\mathcal{T}'$ *that contains a set of generalized locations* $\{l'_1, \ldots, l'_n\}$*, and a parameter* δ*,* $\mathcal{U}$ *is satisfied if and only if* (i) *for each generalized location* $l' \in \{l'_1, \ldots, l'_n\}$*, and for each utility constraint* $u \in \mathcal{U}$*,* $l' \subseteq u$ *or* $l' \cap u = \varnothing$*, and* (ii) *at most* $\delta\%$ *of the locations in* $\mathcal{L}$ *have been suppressed to produce* $\mathcal{T}'$*, where* δ *is a parameter specified by data publishers.*

The first condition of Definition 10 limits the maximum amount of generalization each location is allowed to receive, by prohibiting the construction of generalized locations whose elements (locations) span multiple utility constraints. The second condition ensures that the number of suppressed locations is controlled by a threshold. When both of these conditions hold, the utility constraint set $\mathcal{U}$ is satisfied. Note that we assume that the utility constraint set is provided by data publishers, e.g., using the method in [24]. The example below illustrates Definitions 9 and 10.

Example 3. *Consider the utility constraint set* $\mathcal{U} = \{u_1, u_2\}$*, shown in Figure 3a, and assume that* $\delta = 5$*. The dataset, shown in Figure 3b, does not satisfy* $\mathcal{U}$*, because the locations in the generalized location* $\{a, b, c\}$ *are contained in both* u_1 *and* u_2*. On the other hand, the dataset in Figure 3c satisfies* $\mathcal{U}$*, because the locations of every generalized location are all contained in* u_1*.*

In the following, we provide two examples of real-life applications to justify the importance of utility constraints. The first example comes from the business domain, and it is related to the *Octopus* card, used for payment at various sites (e.g., at convenience stores and service stations) in Hong Kong. Data published by the Octopus card company must preserve privacy and, at the same time, allow meaningful analysis by data recipients, such as owners of specific shops or certain public authorities [35]. The analysis of data recipients often involves counting the number of trajectories that are associated with a particular

type of locations, such as *coffee shops* and *bus stations* when data recipients are coffee shops owners and public transport authorities, respectively. The second example comes from the healthcare domain, and it is related to publishing the locations (e.g., healthcare institutions, clinics, and pharmacies) visited by patients [25]. To support medical research studies, it is important that the published data permit data recipients to accurately count the number of trajectories (or equivalently the number of patients) that are associated with specific locations, or types of locations.

Observe that the number of trajectories that contain a generalized location $l' = (l_1, \ldots, l_n)$, in the generalized dataset $\mathcal{T}'$, is equal to the number of trajectories that contain *at least* one of the locations $l_1, \ldots, l_n$, in the original dataset $\mathcal{T}$. This is because a trajectory $t \in \mathcal{T}$ that contains at least one of these locations corresponds to a trajectory $t' \in \mathcal{T}'$ that contains l'. Thus, the number of trajectories in $\mathcal{T}$ that contain any location in a utility constraint $u \in \mathcal{U}$ can be accurately computed from the generalized dataset $\mathcal{T}'$, when $\mathcal{U}$ is satisfied, as no other location will be generalized together with the locations in u. Therefore, the generalized data that satisfy $\mathcal{U}$ will be practically useful in the aforementioned applications.

We now define the problem that U-SEQANON aims to solve, as follows.

Problem 3. *Given an original trajectory dataset $\mathcal{T}$, a utility constraint set $\mathcal{U}$, and parameters k, m and δ, construct a k^m-anonymous version $\mathcal{T}'$ of $\mathcal{T}$ such that $\mathcal{D}(\mathcal{T}, \mathcal{T}')$ is minimized and $\mathcal{U}$ is satisfied with at most $\delta\%$ of the locations of $\mathcal{T}$ being suppressed.*

Thus, a solution to Problem 3 needs to satisfy the specified utility constraints, without suppressing more than $\delta\%$ of locations, and additionally incur minimum information loss. Note that Problem 3 is NP-hard (it can be restricted to Problem 1, by allowing only instances where $\mathcal{U}$ contains an single utility constrained with all locations in $\mathcal{L}$ and $\delta = 100$) and that U-SEQANON is a heuristic algorithm that may not solve Problem 3 optimally.

4 Anonymization algorithms

In this section, we present our SEQANON, SD-SEQANON, and U-SEQANON anonymization algorithms, which aim at solving Problems 1, 2, and 3, respectively.

4.1 The SEQANON algorithm

The pseudocode of the SEQANON algorithm is illustrated in Algorithm SEQANON. The algorithm takes as input a trajectory dataset $\mathcal{T}$, and the anonymization parameters k and m, and returns the k^m-anonymous counterpart $\mathcal{T}'$ of $\mathcal{T}$. The algorithm employs the apriori principle and works in a bottom up fashion. Initially, it considers and generalizes the subtrajectories of size 1 (i.e., single locations) in $\mathcal{T}$ which have low support. Then, the algorithm continues by progressively increasing the size of the subtrajectories it considers.

In more detail, SEQANON proceeds as follows. First, it initializes $\mathcal{T}'$ (Step 1). Then, in Steps 2 – 8, it considers subtrajectories of size up to m, iteratively. Specifically, in Step 3, it computes the set $\mathcal{S}$, which contains all subtrajectories in $\mathcal{T}$ that have size i (i.e., that have i locations) and support lower than k (i.e., $\sup(s, \mathcal{T}') < k$). SEQANON considers the subtrajectories of $\mathcal{S}$ that have lower support first. This heuristic improves both the efficiency and the effectiveness of the algorithm. This is because remedying such subtrajectories does not require a large amount of generalization, while it contributes to protecting trajectories with higher support. Continuing, for every such trajectory $s \in \mathcal{S}$, the algorithm finds the location l_1 of s with the minimum support (Step 6). SEQANON considers locations with

Algorithm: SEQANON

Input: A dataset $\mathcal{T}$ and anonymization parameters k and m
Output: A k^m-anonymous dataset $\mathcal{T}'$ corresponding to $\mathcal{T}$

1 $\mathcal{T}' := \mathcal{T}$ // Initialize output
2 **for** $i := 1$ **to** m **do**
3 **Let** $\mathcal{S}$ be the set of subtrajectories s of $\mathcal{T}$ with size i such that $\sup(s, \mathcal{T}') < k$ sorted by increasing support
4 **for** *each* $s \in \mathcal{S}$ **do**
5 **while** $\sup(s, \mathcal{T}') < k$ **do**
6 Find the location l_1 of s with the minimum support in $\mathcal{T}'$
7 Find the location $l_2 \neq l_1$ with the minimum $d(l_1, l_2)$
8 Replace all occurrences of l_1 and l_2 in $\mathcal{T}'$ and s with $\{l_1, l_2\}$
9 **return** $\mathcal{T}'$

low support first, as they can be generalized with low information loss. Then, in Step 7, the algorithm searches the locations of $\mathcal{T}$ to detect the location l_2 that has the minimum Euclidean distance from l_1. Finally, SEQANON generalizes l_1 and l_2 by constructing the generalized location $\{l_1, l_2\}$ and replaces every occurrence of l_1 and l_2 with the generalized location $\{l_1, l_2\}$ (Step 8). The algorithm repeats Steps 6 – 8, until the support of the subtrajectory s exceeds the value of the anonymization parameter k.

The following is an example of SEQANON in operation.

Example 4. *We will demonstrate the operation of* SEQANON *using dataset $\mathcal{T}$ of Figure 1a and $k = m = 2$. The intermediate steps are illustrated in Figure 4. The first iteration of the for loop (Steps 2 – 8) considers the subtrajectories of size $i = 1$. It is not hard to verify that all size 1 locations have support greater (or equal) than $k = 2$, thus the algorithm proceeds to $i = 2$. For this case, Step 3 computes the set of subtrajectories $\mathcal{S}$ (illustrated in Figure 4a).* SEQANON *considers subtrajectory $s = (d, a)$, which is the first subtrajectory in $\mathcal{S}$. Then, the algorithm sets $l_1 = a$, because a is the location with the lowest support in (d, a) (Step 6), and $l_2 = b$, because $d(a, b)$ is minimum, according to Figure 1b (Step 7). Finally, in Step 8* SEQANON *replaces a and b with the generalized location $\{a, b\}$ in s and in all the trajectories of $\mathcal{T}'$. After these replacements, we have $s = (d, \{a, b\})$ and the resultant $\mathcal{T}'$ shown in Figure 4b. Since, we still have $\sup(s, \mathcal{T}') < k$, the while loop (Steps 5 – 8) is executed again. This time, $l_1 = \{a, b\}$ and $l_2 = c$, and the algorithm constructs the generalized dataset $\mathcal{T}'$, shown in Figure 4c. The remaining steps of the algorithm* SEQANON *do not change $\mathcal{T}'$. Thus, the final output of* SEQANON *is shown in Figure 4c.*

Time complexity analysis. We first compute the time needed by SEQANON to execute the **for** loop (Steps 2 – 8). For each iteration of this loop, the set $\mathcal{S}$ is constructed, sorted, and explored. The cost of creating and sorting this set is $\mathcal{O}(|\mathcal{L}|^i)$ and $\mathcal{O}(|\mathcal{L}|^i \cdot log(|\mathcal{L}|^i))$, respectively, where $|\mathcal{L}|$ is the size of the location set used in $\mathcal{T}$ and i is the loop counter. These bounds are very crude approximations, which correspond to the case in which all size i subtrajectories have support lower than k. In practice, however, the number of the subtrajectories s with $\sup(s, \mathcal{T}') < k$ is a small fraction of $\mathcal{O}(|\mathcal{L}|^i)$, which depends heavily on the dataset $\mathcal{T}$ and the value of the anonymization parameter k. The cost of exploring the set $\mathcal{S}$ (Steps 4 – 8) is $\mathcal{O}(|\mathcal{L}|^i \cdot (|\mathcal{L}| + |\mathcal{T}'|))$ because, for each element of $\mathcal{S}$, the algorithm needs to consider at most $\mathcal{O}(|\mathcal{L}|)$ locations and access all trajectories in $\mathcal{T}'$. Thus, each iteration of the **for** loop, in Steps 2 – 8, takes $\mathcal{O}(|\mathcal{L}|^i \cdot (log(|\mathcal{L}|^i) + |\mathcal{L}| + |\mathcal{T}'|))$ time, and the time

subT.	sup
(d, a)	1
(c, e)	1
(b, a)	1
(a, d)	1
(b, d)	1

(a)

id	trajectory
t'_1	$(d, \{a, b\}, c, e)$
t'_2	$(\{a, b\}, \{a, b\}, e, c)$
t'_3	$(\{a, b\}, d, e)$
t'_4	$(\{a, b\}, d, e, c)$
t'_5	(d, c)
t'_6	(d, e)

(b)

id	trajectory
t'_1	$(d, \{a, b, c\}, \{a, b, c\}, e)$
t'_2	$(\{a, b, c\}, \{a, b, c\}, e, \{a, b, c\})$
t'_3	$(\{a, b, c\}, d, e)$
t'_4	$(\{a, b, c\}, d, e, \{a, b, c\})$
t'_5	$(d, \{a, b, c\})$
t'_6	(d, e)

(c)

Figure 4: *(a)* Set $\mathcal{S}$ for subtrajectories of size $i = 2$ and the respective supports, *(b)* Transformed dataset $\mathcal{T}'$ after SEQANON has processed the subtrajectory (d, a), and *(c)* The final 2^2-anonymous result $\mathcal{T}'$, produced by SEQANON.

complexity of SEQANON is $\mathcal{O}\big(\sum_{i=1}^{m}(|\mathcal{L}|^i \cdot (log(|\mathcal{L}|^i) + |\mathcal{L}| + |\mathcal{T}'|))\big)$.

4.2 The SD-SEQANON algorithm

SD-SEQANON takes as input an original trajectory dataset $\mathcal{T}$, the anonymization parameters k and m, and a location taxonomy, and returns the k^m-anonymous counterpart $\mathcal{T}'$ of $\mathcal{T}$. The algorithm operates similarly to SEQANON, but it takes into account both the Euclidean distance and the semantic similarity of locations, when it applies generalization to them.

The pseudocode of SD-SEQANON is provided in Algorithm SD-SEQANON. Notice that SD-SEQANON and SEQANON differ in Step 7. This is because SD-SEQANON calculates the product of the Euclidean distance for the locations l_1 and l_2, and the $\mathcal{SD}$ measure for the generalized location $\{l_1, l_2\}$ (see Definition 7). Thus, it aims at creating a generalized location, which consists of locations that are close in proximity and are semantically similar. The time complexity of SD-SEQANON is the same as that of SEQANON, because the computation in Step 7 does not affect the complexity.

Algorithm: SD-SEQANON

Input: A dataset $\mathcal{T}$, a locations hierarchy and anonymization parameters k and m
Output: A k^m-anonymous dataset $\mathcal{T}'$ corresponding to $\mathcal{T}$

1 $\mathcal{T}' := \mathcal{T}$ `// Initialize output`
2 **for** $i := 1$ **to** m **do**
3 **Let** $\mathcal{S}$ be the set of subtrajectories s of $\mathcal{T}$ with size i such that $\sup(s, \mathcal{T}') < k$ sorted by increasing support
4 **for** *each* $s \in \mathcal{S}$ **do**
5 **while** $\sup(s, \mathcal{T}') < k$ **do**
6 Find the location l_1 of s with the minimum support in $\mathcal{T}'$
7 Find the location $l_2 \neq l_1$ with the minimum $d(l_1, l_2) \cdot \mathcal{SD}(\{l_1, l_2\})$
8 Replace all occurrences of l_1 and l_2 in $\mathcal{T}'$ and s with $\{l_1, l_2\}$
9 **return** $\mathcal{T}'$

4.3 The U-SEQANON algorithm

The U-SEQANON algorithm takes as input an original trajectory dataset $\mathcal{T}$, anonymization parameters k, m and δ, as well as a utility constraint set $\mathcal{U}$. The algorithm differs from SEQANON and SD-SEQANON along two dimensions. First, the k^m-anonymous dataset it produces satisfies $\mathcal{U}$, hence it meets the data publishers' utility requirements. Second, it additionally employs suppression (i.e., removes locations from the resultant dataset), when generalization alone is not sufficient to enforce k^m-anonymity.

Algorithm: U-SEQANON

Input: A dataset $\mathcal{T}$, utility constraint set $\mathcal{U}$, and anonymization parameters k, m, and δ
Output: A k^m-anonymous dataset $\mathcal{T}'$ corresponding to $\mathcal{T}$

1 $\mathcal{T}' := \mathcal{T}$ `// Initialize output`
2 **for** $i := 1$ **to** m **do**
3 **Let** $\mathcal{S}$ be the set of subtrajectories s of $\mathcal{T}$ with size i such that $\sup(s, \mathcal{T}') < k$ sorted by increasing support
4 **for** *each* $s \in \mathcal{S}$ **do**
5 **while** $\sup(s, \mathcal{T}') > 0$ *or* $\sup(s, \mathcal{T}') < k$ **do**
6 Find the location l_1 of s with the minimum support in $\mathcal{T}'$
7 Find the utility constraint $u \in \mathcal{U}$ that contains the location l_1
8 Find the location $l_2 \neq l_1, l_2 \in u$ with the minimum $d(l_1, l_2)$
9 **if** *Cannot find location* l_2 **then**
10 Suppress location l_1 from $\mathcal{T}'$
11 **if** *More than* $\delta\%$ *of locations have been suppressed* **then**
12 Exit: $\mathcal{U}$ is not satisfied
13 **else**
14 Replace all occurrences of l_1 and l_2 in $\mathcal{T}'$ and s with $\{l_1, l_2\}$
15 **return** $\mathcal{T}'$

The pseudocode of U-SEQANON is provided in Algorithm U-SEQANON. As can be seen, the algorithm initializes $\mathcal{T}'$ (Step 1) and then follows the apriori principle (Steps 2 – 14). After constructing and sorting $\mathcal{S}$, U-SEQANON iterates over each subtrajectory in $\mathcal{S}$ and applies generalization and/or suppression, until its support is either at least k or 0 (Steps 3 – 5). Notice that $sup(s, \mathcal{T}') = 0$ corresponds to an empty subtrajectory s (i.e., the result of suppressing all locations in s), which does not require protection. Next, U-SEQANON finds the location l_1 with the minimum support in $\mathcal{T}'$ and the utility constraint that contains it (Steps 6 – 7). Then, the algorithm finds a different location l_2, which also belongs to u and is as close to l_1 as possible, according to the Euclidean distance (Step 8). In case such a location cannot be found (i.e., when there is a single generalized location that contains all locations in u), U-SEQANON suppresses l_2 from $\mathcal{T}'$ (Steps 9 – 10). If more than $\delta\%$ of locations have been suppressed, $\mathcal{U}$ cannot be satisfied and the algorithm terminates (Steps 11 – 12). Otherwise, U-SEQANON generalizes l_1 and l_2 together and replaces every occurrence of either of these locations with the generalized location $\{l_1, l_2\}$ (Step 14). The algorithm repeats Steps 2 – 14 as long as the size of the considered subtrajectories does not exceed m. After considering the subtrajectories of size m, U-SEQANON returns the k^m-anonymous dataset $\mathcal{T}'$ that satisfies $\mathcal{U}$, in Step 15.

The following is an example of U-SEQANON in operation.

Example 5. *We will demonstrate the operation of* U-SEQANON *with input the original dataset*

#	UC
1	$\{b, c\}$
2	$\{a, d, e\}$

(a)

id	trajectory
t'_1	$(\{a, d\}, \{a, d\}, c, e)$
t'_2	$(b, \{a, d\}, e, c)$
t'_3	$(\{a, d\}, \{a, d\}, e)$
t'_4	$(b, \{a, d\}, e, c)$
t'_5	$(\{a, d\}, c)$
t'_6	$(\{a, d\}, e)$

(b)

id	trajectory
t'_1	$(\{a, d, e\}, \{a, d, e\}, \{b, c\}, \{a, d, e\})$
t'_2	$(\{b, c\}, \{a, d, e\}, \{a, d, e\}, \{b, c\})$
t'_3	$(\{a, d, e\}, \{a, d, e\}, \{a, d, e\})$
t'_4	$(\{b, c\}, \{a, d, e\}, \{a, d, e\}, \{b, c\})$
t'_5	$(\{a, d, e\}, \{b, c\})$
t'_6	$(\{a, d, e\}, \{a, d, e\})$

(c)

Figure 5: *(a)* Set of Utility Constraints *(b)* Transformed dataset $\mathcal{T}'$ after the processing of subtrajectory (d, a), and *(c)* The final 2^2-anonymous result $\mathcal{T}'$ meeting the provided set of UC

$\mathcal{T}$, shown in Figure 1a, the utility constraint set in Figure 5a, $k = m = 2$, and $\delta = 5\%$. During the first iteration of the for loop (Steps 2 – 14), U-SEQANON *considers the subtrajectories of size $i = 1$, which all have a support of at least 2. Thus, the algorithm considers subtrajectories of size $i = 2$, and constructs the set $\mathcal{S}$ shown in Figure 4a (Steps 4 – 3). Then,* U-SEQANON *considers the subtrajectory $s = (d, a)$ in $\mathcal{S}$, which has the lowest support in $\mathcal{T}'$ (Step 6). Next, in Steps 6 – 8, the algorithm finds the location $l_1 = a$, which has the lowest support in $\mathcal{T}'$, and the location $l_2 = d$, which belongs to the same utility constraint as a and is the closest to it – see also the map in Figure 1b. After that, the algorithm replaces a and d with the generalized location $\{a, d\}$ in s and all the trajectories of $\mathcal{T}'$ (Step 14). After these replacements, $s = (\{a, d\}, \{a, d\})$ and the trajectory dataset $\mathcal{T}'$ is as shown in Figure 4b. Since the support of s in $\mathcal{T}'$ is at least k, the while loop in Step 5 terminates and the algorithm checks the next problematic subtrajectory, $s = (c, e)$. After considering all problematic subtrajectories of size 2,* U-SEQANON *produces the 2^2-anonymous dataset in Figure 5c, which satisfies $\mathcal{U}$.*

The time complexity of U-SEQANON is the same as that of SEQANON, in the worst case when $\mathcal{U}$ is comprised of a single utility constraint that contains all locations in $\mathcal{L}$, $\mathcal{S}$ contains $O(|\mathcal{L}|^i)$ subtrajectories with support in $(0, k)$, and $\delta = 100$. Note that the cost of suppressing a location l_1 is $O(|\mathcal{T}'|)$ (i.e., the same as that of replacing the locations l_1 and l_2 with the generalized location (l_1, l_2) in all trajectories in $\mathcal{T}'$).

5 Experimental evaluation

In this section, we provide a thorough experimental evaluation of our approach, in terms of data utility and efficiency.

Experimental setup. We implemented our algorithms in C++ and tested them on an Intel Core i7 at 2.2 GHz with 6 GB of RAM. In our experiments, we used both synthetic and real datasets. The synthetic dataset, referred to as Oldenburg, was generated using the Brinkhoff's data generator [6] and contains synthetic trajectories of objects moving on the Oldenburg city map. This setting has been used by many works [1, 29, 35, 39]. We normalized the trajectories, so that all coordinates take values in a $10^3 \times 10^3$ map, and simulated trajectories corresponding to these routes, as follows. The map was divided into 100 regions using a uniform grid. A moving object visits a sequence of regions in a certain order. The centroids of the visited regions model the locations in the trajectories of $\mathcal{T}$. The Oldenburg dataset contains $18,143$ trajectories, whose average length is 4.72, and 100 locations.

In addition, we used a dataset that has been derived from the Gowalla location-based social networking website and contains the check-ins (locations) of users between February 2009 and October 2010 [10]. In our experiments, we used aggregate locations of $86,061$ users, in the state of New York and nearby areas. This dataset is referred to as Gowalla and contains $86,061$ trajectories, whose average length is 3.92, and 662 locations.

To study the effectiveness of our approach, we compare it against the NGRAMS method [7], discussed in Section 2, using the implementation provided by the authors of [7]. Contrary to [5], the NGRAMS method was developed for count query answering. Thus, a comparison between the NGRAMS method and ours allows us to evaluate the effectiveness of our approach with respect to count query answering. For this comparison, we set the parameters l_{max}, n_{max}, and e to the values 20, 5, and 0.1, respectively, which were suggested in [7]. Unless otherwise stated, k is set to 5 and m is set to 2. The location taxonomies were created as in [35]. Specifically, each non-leaf node in the taxonomy for the Oldenburg (respectively, Gowalla) dataset has 5 (respectively, 6) descendants.

Measures. To measure data utility we used the *Average Relative Error (ARE)* measure [21], which has become a defacto data utility indicator [24, 31]. ARE estimates the average number of trajectories that are retrieved incorrectly, as part of the answers to a workload of COUNT queries. Low ARE scores imply that anonymized data can be used to accurately estimate the number of co-occurrences of locations. We used workloads of 100 COUNT queries, involving randomly selected subtrajectories with size in $[1, 2]$, because the output of the NGRAMS method contained very few longer trajectories (see Figure 10b).

In addition, we used *Kullback–Leibler divergence* (KL-divergence), an information-theoretic measure that quantifies information loss and is used widely in the anonymization literature [16, 19]. In our context, KL-divergence measures support estimation accuracy based on the difference between the distribution of the support of a set of subtrajectories S, in the original and in the anonymized data[2]. Let P_S (respectively, Q_S) be the distribution of the support of the subtrajectories in S in the dataset $\mathcal{T}$ (respectively, generalized dataset $\mathcal{T}'$). The KL-divergence between P_S and Q_S is defined as:

$$D_{KL}(P_S \parallel Q_S) = \sum_{s \in S} P_S \ln \left(\frac{P_S}{Q_S} \right)$$

Clearly, low values in KL-divergence are preferred, as they imply that a small amount of information loss was incurred to generalize the subtrajectories in S. Furthermore, we used statistics on the number, size, and distance, for the locations in generalized data.

To evaluate the extent to which SD-SEQANON generalizes semantically close locations together, we compute a *semantic similarity penalty* $\mathcal{P}$ for the generalized dataset, as follows:

$$\mathcal{P}(\mathcal{T}') = \frac{\sum_{t' \in \mathcal{T}'} \left(\frac{1}{|t'|} \cdot \sum_{l' \in t'} \mathcal{SD}(l') \right)}{|\mathcal{T}'|}$$

where t' is a trajectory in the generalized dataset $\mathcal{T}'$, with size $|t'|$, and $|\mathcal{T}'|$ is the number of trajectories in $\mathcal{T}'$. $\mathcal{P}$ reflects how semantically dissimilar are (on average) the locations in the trajectories in the generalized dataset. The values in $\mathcal{P}(\mathcal{T}')$ are in $[0, 1]$ and lower values imply that the generalized locations in $\mathcal{T}'$ contain more semantically similar locations.

[2]The support of a subtrajectory $s \in S$ in $\mathcal{T}'$ is computed as the support of its generalized counterpart (i.e., the subtrajectory induced by the generalized locations of each location in s).

To evaluate the ability of the SEQANON and NGRAMS algorithms to permit sequential pattern mining, we employ a testing framework similar to that of Gionis et al. [17]. In more detail, we construct random projections of the datasets produced by our algorithms, by replacing every generalized location in it with a random location, selected from that generalized location. Then, we use Prefixspan [30], to obtain the frequent sequential patterns (i.e., the sequential patterns having support larger than a threshold) in the original, the output of NGRAMS and the projected datasets. Next, we calculate the percentage of the frequent sequential patterns of the original dataset that are preserved in the output of NGRAMS and in the projected datasets. We also calculate the percentage of the frequent sequential patterns in the output of NGRAMS and in the projected datasets that are not frequent in the original dataset. Clearly, an anonymized dataset (produced by either NGRAMS or by our algorithms) allows accurate mining, when: *(i)* a high percentage of its frequent sequential patterns are frequent in the original dataset, and *(ii)* a low percentage of its frequent sequential are not frequent in the original dataset.

5.1 Data utility

In this section, we evaluate the effectiveness of the SEQANON, SD-SEQANON, and U-SEQANON algorithms at preserving data utility.

SEQANON. We begin by evaluating the data utility offered by the SEQANON algorithm, using some general statistics, computed for the output of this algorithm on the Oldenburg dataset. Specifically, we measured the number of the locations that were released intact (referred to as original locations) and the number of the locations that were generalized. For the generalized locations, we also measured their average size and distance. Initially, we varied the anonymization parameter k in [2, 100]. Our results are summarized in Figures 6a-7a.

In Figure 6a, we present the number of the original locations, as a function of k. As expected, increasing k led to fewer original locations. In Figure 6b, we report the number of generalized locations. When k increases, more locations are grouped together to ensure k^m-anonymity, leading to fewer generalized locations. As an immediate result, the average number of locations in a generalized location increased, as shown in Figure 6c. In addition, we report the average Euclidean distance of all locations contained in each generalized location. We normalize this distance as a percentage of the maximum possible distance (i.e., the distance between the two furthermost points). This percentage quantifies the distortion

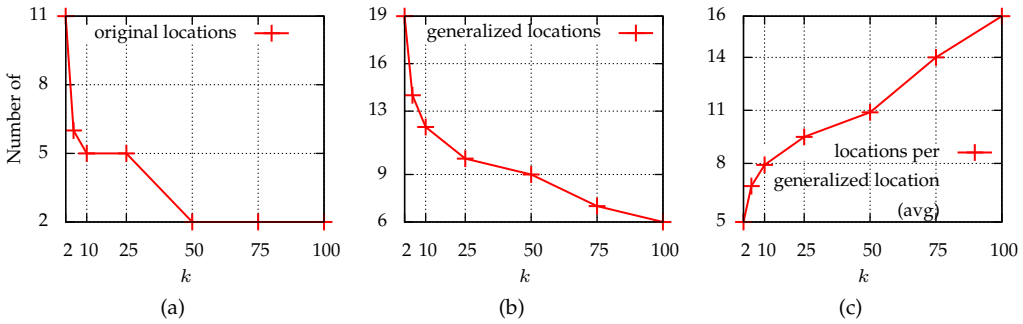


Figure 6: number of *(a)* original (non generalized) locations published, *(b)* generalized locations published and *(c)* average generalized location size

in a generalized location. In Figure 7a, we illustrate the distance percentage as a function of k. When k increases, more locations are grouped together in the same generalized location, leading to more distortion. As SEQANON focuses on minimizing the Euclidean distance of locations in each generalized location, the distortion is relatively low and increases slowly.

To show the impact of m on data utility, we set $k = 5$ and varied m in [1,4]. Since our dataset has an average of 4.72 locations per trajectory, $m = 3$ (respectively, $m = 4$) means that the adversary knows approximately 65% (respectively, 85%) of a user's locations. So, for $m = 3$ and $m = 4$, we expect significant information loss. On the contrary, for $m = 1$, all locations have support greater than $k = 5$, so no generalization is performed and no generalized locations are created. As m increases, more generalizations are performed in order to eliminate subtrajectories with low support. This leads to fewer generalized locations with larger sizes. These results are shown in Figures 7b-8b.

Also, we evaluated the impact of dataset size on data utility, using various random subsets of the original dataset, containing $2,000$, $5,000$, $10,000$, and $15,000$ records. In Figure 8c, we illustrate the number of original locations for variable dataset sizes. For larger datasets, this number increases, as the support of single locations is higher. Consequently, the support of subtrajectories increases, and fewer locations are generalized. This leads to more generalized locations, with lower average size, and lower distance, as can be seen in Figures 9a-9c.

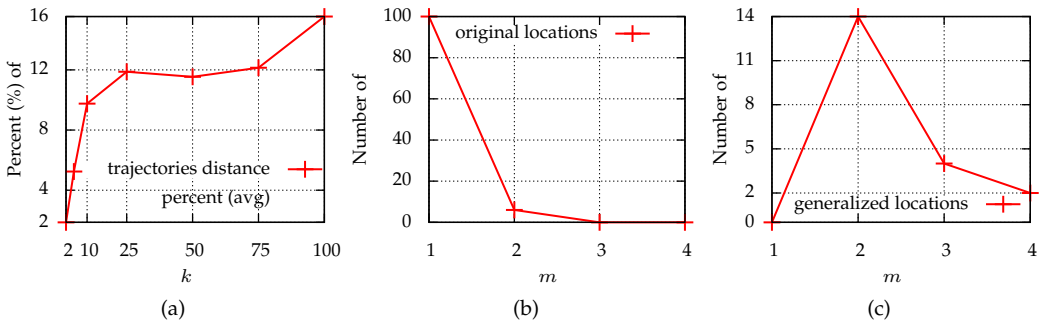


Figure 7: number of *(a)* average percent of distance in generalized locations, *(b)* original (non generalized) locations published and *(c)* generalized locations published

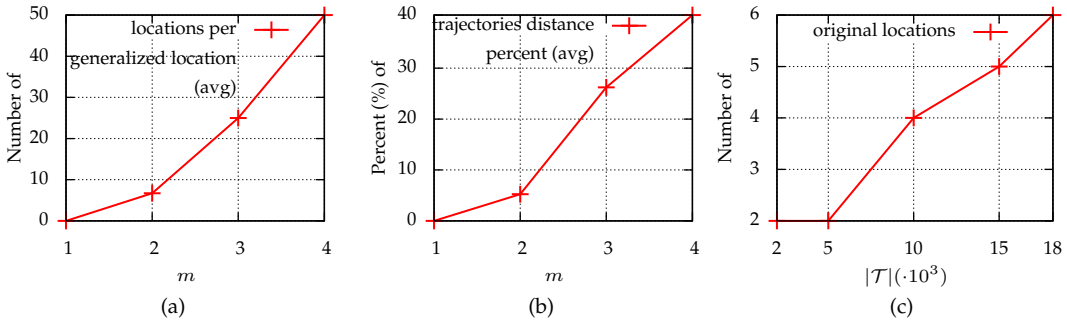


Figure 8: *(a)* average generalized location size, *(b)* average percent of distances in generalized locations and *(c)* number of original (non generalized) locations published

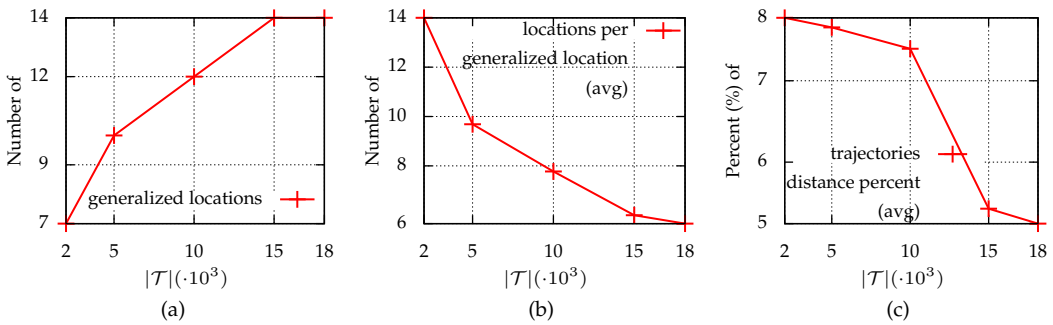


Figure 9: *(a)* number of generalized locations published, *(b)* average generalized location size and *(c)* average percent of distances in generalized locations

SEQANON vs. NGRAMS. In this section, we report the count of the subtrajectories of different sizes that are created by the NGRAMS method. In addition, we report the ARE and KL-divergence scores for the SEQANON and NGRAMS methods. Finally, we report the percentage of the frequent sequential patterns of the original dataset that are preserved in the anonymous dataset and the percentage of frequent sequential patterns of the anonymous dataset that are not frequent on the original dataset. The results of comparing NGRAMS to SD-SEQANON and to U-SEQANON were quantitatively similar (omitted for brevity).

Fig. 10a reports the number of subtrajectories of different sizes that are contained in the Oldenburg dataset, denoted with $\mathcal{T}$, and the output of NGRAMS, denoted with $\mathcal{T}'_{\text{SEQANON}}$. As can be seen, the output of NGRAMS contains noisy versions of only a small percentage (0.29%) of short subtrajectories. Thus, the information of the subtrajectories of length greater than 4, which correspond to 72.62% of the subtrajectories in the dataset, is lost. The results of the same experiment on the Gowalla dataset, reported in Fig. 10b, are quantitatively similar. That is, NGRAMS created noisy versions of 0.11% of the subtrajectories with 3 or fewer locations, and the information of all longer subtrajectories, which correspond to 98.1% of the subtrajectories in the dataset, is lost. On the other hand, SEQANON employs generalization, which preserves the information of all subtrajectories, although in a more aggregate form.

Figure 11a reports the ARE scores for SEQANON and NGRAMS, as a function of k, for the Oldenburg dataset and for 100 queries involving subtrajectories of sizes in $[1, 2]$. In this experiment, we set m to 3, assuming that an attacker knows about 75% of the locations visited by a user. As can be seen, the ARE scores for SEQANON are at least 4.45 and up to 7.3 times lower (better) than those of NGRAMS. Furthermore, the ARE scores for our method increase with k, which is expected due to the utility/privacy trade-off. On the contrary, the ARE scores for NGRAMS remained the same, as this method does not use the parameter k. Next, we studied the impact of m on ARE, by varying this parameter in $[1, 4]$ (recall that $m = 4$ implies that the attacker knows almost all of the locations, visited by a user). Figure 11b reports the result for $k = 5$, on the Oldenburg dataset. The ARE scores for our algorithm were at least 6.3 and up to 11.9 times better than those of NGRAMS. Also, it can be seen that the ARE scores for SEQANON increase with m, as the algorithm has to incur more generalization to protect from stronger attackers. The ARE scores for NGRAMS are not affected by m, as this method does not use this parameter. We also studied the impact of k and m on ARE, using the Gowalla dataset, and obtained similar results, which

size	# in $\mathcal{T}$	# in $\mathcal{T}'_{\text{NGRAMS}}$
1	100	73
2	6955	220
3	48268	222
4	124070	2
5	177054	–
6	158684	–
7	93479	–
8	36328	–
9	8989	–
≥ 10	1366	–

(a)

size	# in $\mathcal{T}$	# in $\mathcal{T}'_{\text{NGRAMS}}$
1	662	427
2	107811	577
3	803093	6
4	1757607	–
5	2959148	–
6	4478016	–
7	6033559	–
8	7138181	–
9	7336036	–
≥ 10	17281056	–

(b)

Figure 10: Number of distinct subtrajectories of different sizes, for *(a)* the Oldenburg, and *(b)* the Gowalla dataset.

are reported in Figures 11c and 11d.

The results with respect to KL-divergence, as a function of m, are shown in Figure 14. Specifically, Figure 14a reports the result for the set S of all subtrajectories with size 1 (i.e., all locations) in the Oldenburg dataset, and for $k = 5$. As can be seen, the information loss for SEQANON was significantly lower than that of NGRAMS, particularly for smaller values of m. Increasing m led to fewer, larger generalized locations. Thus, the KL-divergence scores of SEQANON increase with m, while those of NGRAMS are not affected by m, for the reason explained before. Figure 14b (respectively, Figure 14c) reports the KL-divergence scores for 100 randomly selected locations (respectively, subtrajectories with size 2) in the Gowalla dataset. As noted previously, we did not consider longer subtrajectories, as all but 6 of the subtrajectories in the output of NGRAMS have size at most 2. Again, SEQANON outperformed NGRAMS by a large margin, which demonstrates that our method can preserve the distribution of the support of subtrajectories better. Specifically, the KL-divergence scores for our method were at least 20% and up to 4.3 times lower (better) than those for NGRAMS. Similar results were obtained for larger k values (omitted for brevity).

Next, we present the results of experiments, in which we evaluated the ability of the algorithms to support frequent sequential pattern mining. Specifically, we report the percentage of the frequent sequential patterns of the original dataset that are preserved in the anonymous dataset and the percentage of frequent sequential patterns of the anonymous dataset that are not frequent on the original dataset. In order to get a more accurate statistical distribution we used 2000 randomly projected anonymous datasets[3]. In our experiments, we mined the Oldenburg and Gowalla dataset, using a support threshold of 0.83% and 0.14%, respectively.

Figures 12a and 12b present the median and standard deviation of the percentage of frequent sequential patterns that were preserved in the anonymous dataset, when SEQANON was applied with a k in $[2, 100]$ and NGRAMS with the default parameters, on Oldenburg and Gowalla datasets respectively. The results for SEQANON are significantly better than those of NGRAMS. Specifically, SEQANON reported at least 48% and up to 192% more frequent patterns than NGRAMS. Note also that SEQANON performs better when k is smaller, because it applies a lower amount of generalization.

[3] *Creating more projected datasets allows estimating the dataset quality more accurately. However, the increase in the accuracy of estimation was negligible, when we used more than* 2000 *projected datasets, in our experiments.*

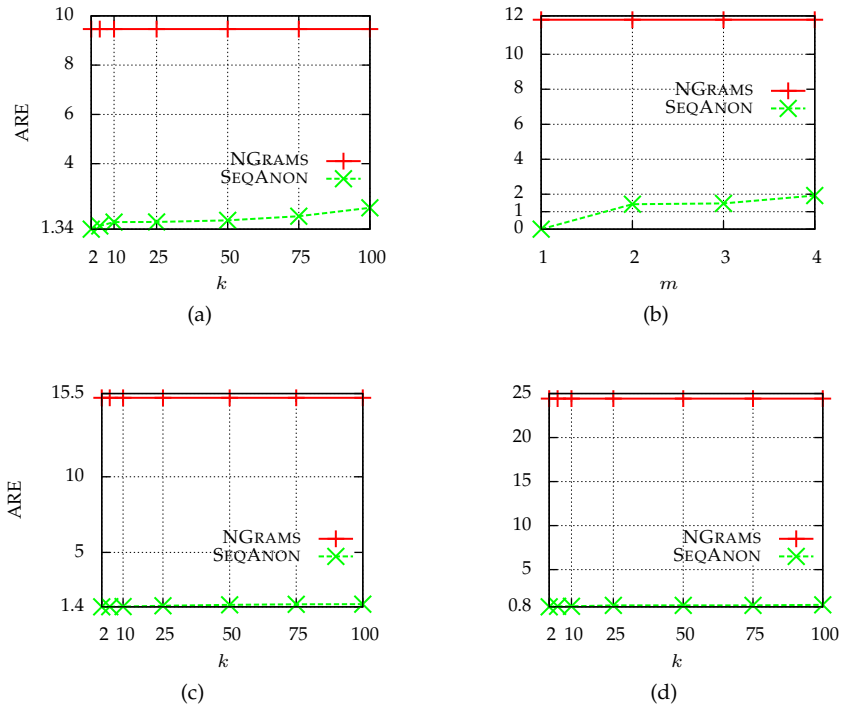


Figure 11: ARE for queries involving 100 randomly selected subtrajectories *(a)* with size in $[1, 2]$, in the Oldenburg dataset (varying k), *(b)* with size in $[1, 2]$, in the Oldenburg dataset (varying m), *(c)* with size 1, in the Gowalla dataset (varying k), and *(d)* with size 2 in the Gowalla dataset (varying k).

Figures 12c and 12d report the median and standard deviation of the percentage of frequent sequential patterns of the anonymous dataset that are not frequent on the original dataset, when SEQANON was applied with a k in $[2, 100]$ and NGRAMS with the default parameters, on Oldenburg and Gowalla datasets respectively. Again, the results for SEQANON are better than those of NGRAMS. In more detail, the percentage of patterns that are not frequent in the original dataset but are frequent in the anonymized dataset was up to 1.2 times lower (on average 45% lower) for SEQANON compared to NGRAMS. Note that as k increases, the percentage of such patterns for SEQANON decreases, because fewer patterns are frequent in the anonymized dataset.

Figures 13a and 13b report the percentage of frequent sequential patterns preserved in anonymous dataset and the percentage of frequent sequential patterns of the anonymous dataset that are not frequent on the original dataset. We set $k = 5$ and vary m in $[1, 4]$ for SEQANON, while NGRAMS was executed with the default parameters. Larger values of m result in more generalization. Thus, SEQANON preserves at least 20% and up to 650% more frequent patterns frequent than NGRAMS, while the percentage of patterns that are incorrectly identified as frequent is lower by at least 50% and up to 500% compared to that for NGRAMS. The corresponding results for Gowalla were qualitatively similar (omitted for brevity).

In summary, our results show that the SEQANON algorithm permits more effective query answering, more accurate pattern mining, and incurs lower information loss than NGRAMS. Thus, it offers a different trade-off between utility and privacy, which is important in ap-

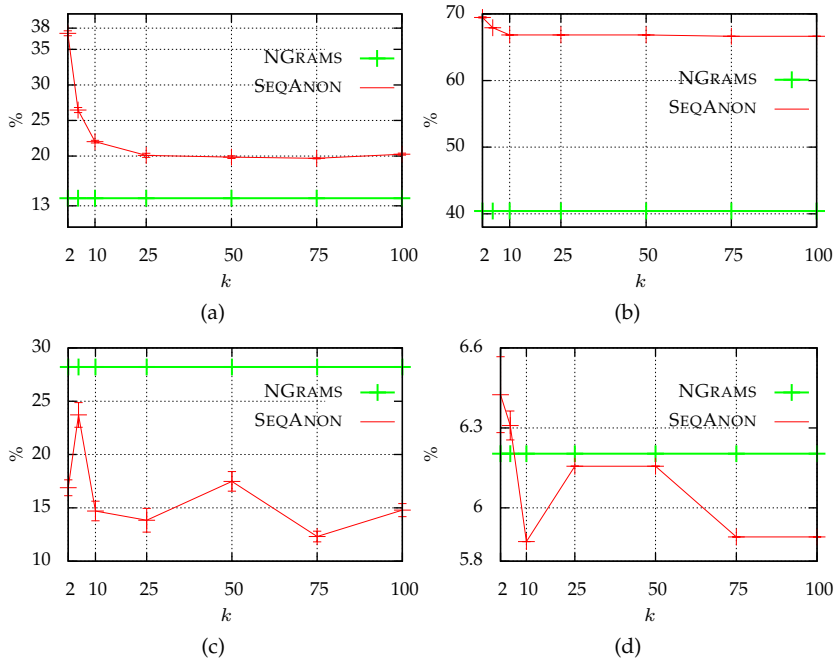


Figure 12: Percentage of frequent sequential patterns preserved in anonymous dataset (median) for varying k on *(a)* Oldenburg dataset, *(b)* Gowalla dataset, and percentage of frequent sequential patterns of the anonymous dataset that are not frequent on the original (median) for varying k on *(c)* Oldenburg dataset and *(d)* Gowalla dataset.

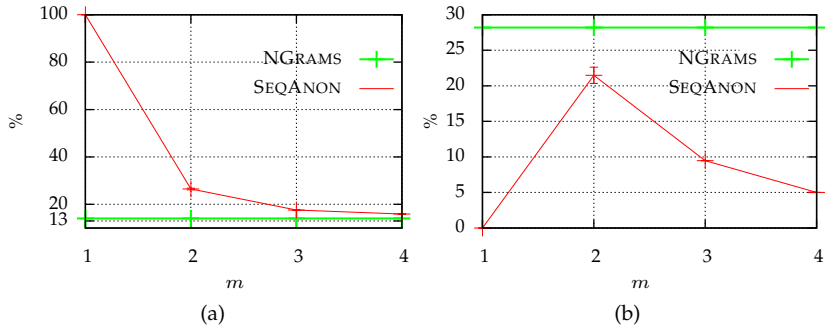


Figure 13: *(a)* Percentage of frequent sequential patterns preserved in anonymous dataset (median) and *(b)* percentage of frequent sequential patterns of the anonymous dataset that are not frequent on the original (median) on Oldenburg dataset for varying m.

plications that require preserving data truthfulness.

SD-SEQANON. In this section, we evaluate the data utility offered by SD-SEQANON, using statistics computed for the output of this algorithm. Figure 15a (respectively, 15b) reports the average distance of all locations that are mapped to each generalized location, as a function of k, for the Oldenburg (respectively, the Gowalla) dataset. Increasing k leads SD-SEQANON to create more, larger generalized locations, which results in larger average location distance. Note that the results for SD-SEQANON are slightly worse than those of SEQANON and may also decrease as k gets larger. This is expected because SD-

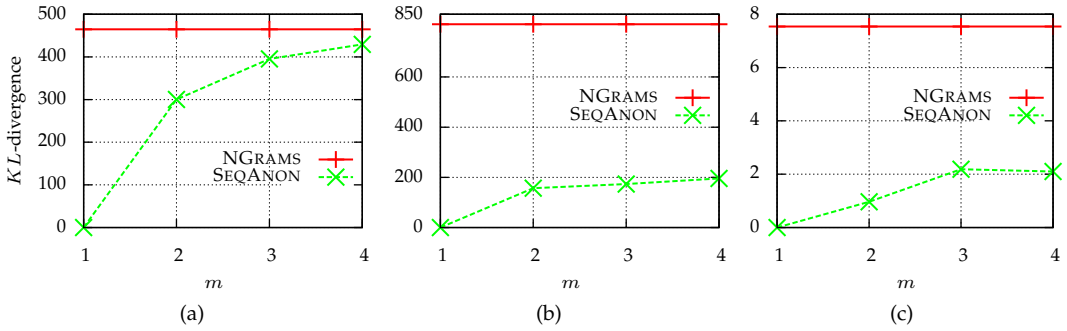


Figure 14: KL-divergence for the distribution of the support of *(a)* all locations in the Oldenburg dataset, *(b)* 100 randomly selected locations in the Gowalla dataset, and *(c)* 100 randomly selected subtrajectories of size 2 in the Gowalla dataset.

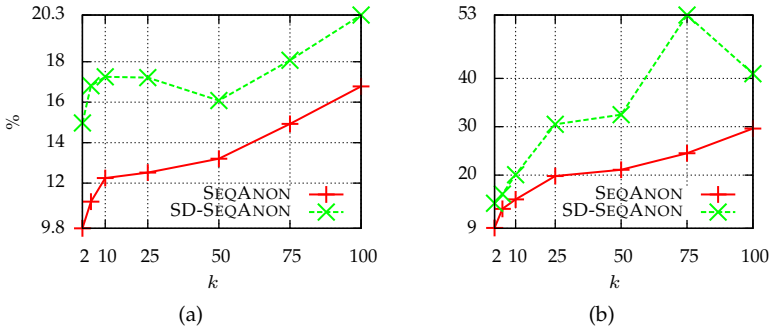


Figure 15: Average percent of distance in generalized locations for *(a)* Oldenburg and *(b)* Gowalla dataset.

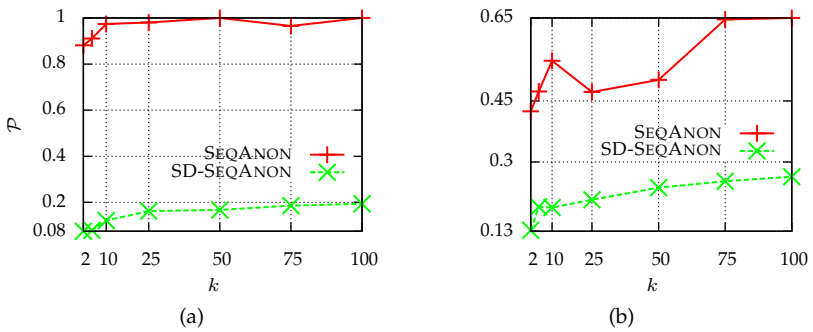


Figure 16: Semantic similarity penalty $\mathcal{P}(\mathcal{T}')$ for *(a)* Oldenburg and *(b)* Gowalla dataset.

SEQANON takes into account not only the distance but also the semantic similarity of locations, when performing generalization. Thus, as can be seen in Figures 16a and 16b, the SD-SEQANON algorithm performs much better than SEQANON with respect to the *semantic similarity penalty* (the scores for SD-SEQANON are at least 2.5 and up to 4.6 times

better than those for SEQANON). This demonstrates that SD-SEQANON generalizes more semantically close locations together.

Figure 17 presents the average size of generalized locations, the average percent of distance in generalized locations, and the semantic similarity penalty $\mathcal{P}$, as a function of m. In these experiments k was set to 50. As can be seen, increasing m leads the algorithms to construct fewer, larger generalized locations, which are comprised of more distant and less semantically close locations. As expected, SEQANON generalized together locations that are closer in proximity (see Figure 17b) but more semantically distant (see Figure 17c).

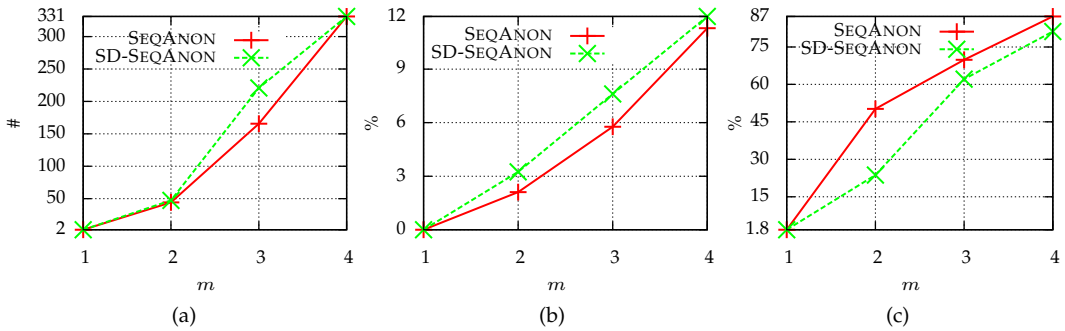


Figure 17: *(a)* average generalized location size, *(b)* average percent of distance in generalized locations and *(c)* average percent of similarity in generalized locations for $k = 50$ (Gowalla dataset).

U-SEQANON. This section reports results for the U-SEQANON algorithm, which was configured with three different utility constraint sets, namely $\mathcal{U}_1$, $\mathcal{U}_2$ and $\mathcal{U}_3$, and a suppression threshold $\delta = 10$. The utility constraints in each of these sets contain a certain number of semantically close locations (i.e., sibling nodes in the location taxonomy), as shown in Figure 18. Note that $\mathcal{U}_3$ is more difficult to satisfy than $\mathcal{U}_1$, as the number of allowable ways to generalize locations is smaller.

$\mathcal{U}_1$	$\mathcal{U}_2$	$\mathcal{U}_3$
$\|u_1\| = 50$	$\|u_1\| = 25$	$\|u_1\| = 20$
$\|u_2\| = 50$	$\|u_2\| = 25$	$\|u_2\| = 14$
	$\|u_3\| = 25$	$\|u_3\| = 16$
	$\|u_4\| = 25$	$\|u_4\| = 14$
		$\|u_5\| = 18$
		$\|u_6\| = 18$

Figure 18: The size of the utility constraints in each utility constraint set $\mathcal{U}_1$, $\mathcal{U}_2$, and $\mathcal{U}_3$.

Figure 19a reports the average size of generalized locations, for various values of k in $[2, 100]$. Note that all configurations of U-SEQANON created smaller generalized locations than those constructed by SEQANON, and the smallest generalized locations were created when $\mathcal{U}_3$ was used. This is because the presence of utility constraints that contain a small number of locations reduces the number of available generalizations. For this reason, all configurations of U-SEQANON generalized together more distant locations than SEQANON, as can be seen in Figure 19b. Also, observe that the use of less restrictive utility constraints (e.g., those in $\mathcal{U}_1$) leads U-SEQANON to generalize together locations that are

close in proximity.

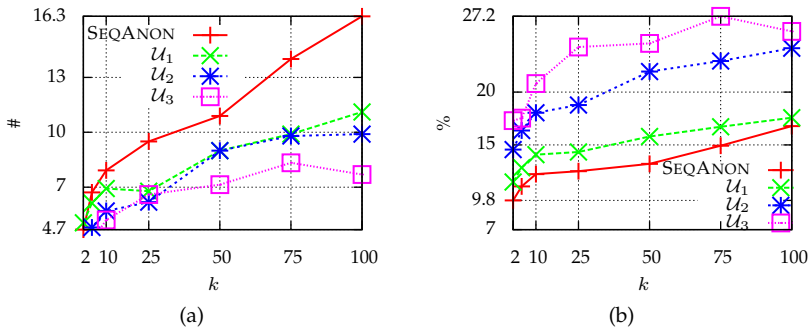


Figure 19: *(a)* average generalized location size and *(b)* average percent of distance in generalized locations, for the Oldenburg dataset.

5.2 Efficiency

In this section, we evaluate the impact of the anonymization parameters k and m, the dataset size, and the location size, on the efficiency of our approach.

SEQANON. To highlight the benefit of employing the apriori principle on efficiency, we created a version of SEQANON, called SEQANON_F, which does not use the apriori principle. In this version, we removed the **for** loop from Step 2 of SEQANON and set $i = m$. In other words, SEQANON_F tries to deal directly with subtrajectories of size m. First, we studied the impact of the anonymization parameter k on efficiency. As illustrated in Figure 20a, the execution time of both algorithms increases with k. This is expected because there are more subtrajectories with a lower support than k, when k is larger. However, SEQANON is significantly more efficient and scalable than SEQANON_F. Specifically, the SEQANON algorithm was at least 6.5 and up to 10.85 times more efficient than SEQANON_F. Then, we studied the impact of m on efficiency and report the results in Figure 20b. As can be seen, the use of the apriori principle by SEQANON enables it to scale much better than SEQANON_F, as m gets larger. In addition, we studied the effect of dataset size on the execution time of SEQANON. The results in Figure 20c demonstrate that the SEQANON outperforms SEQANON_F, being up to 20 times more efficient. We then studied the impact of m, dataset size, and number of locations on the larger Gowalla dataset, and report the results in Figure 21. Due to the fact that this dataset is more sparse than the Oldenburg dataset and contains a larger number of distinct locations, SEQANON needed more time to anonymize it. Again, SEQANON was more efficient than SEQANON_F, which needed more than 12 hours to anonymize the entire dataset. Of note, SEQANON is less efficient than NGRAMS, mainly because generalization requires accessing all trajectories in the dataset more times.

SD-SEQANON and U-SEQANON. In this section, we evaluate the impact of k on the efficiency of SD-SEQANON and U-SEQANON. In this set of experiments, we set $m = 2$ and configured U-SEQANON using the utility constraint sets $\mathcal{U}_1$, $\mathcal{U}_2$, and $\mathcal{U}_3$, and $\delta = 10$. Figure 22a reports the runtime of SD-SEQANON, for varying k, and for the Oldenburg dataset. As can be seen, the runtime of SD-SEQANON is similar to that of SEQANON. The small differences between these algorithms are attributed to the fact that they use different simi-

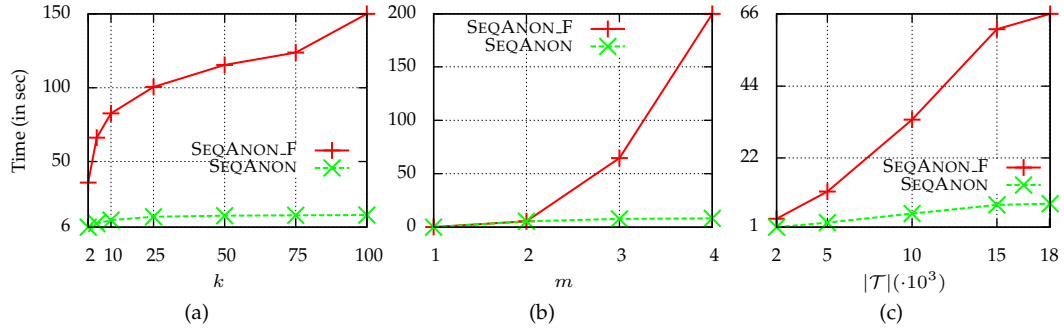


Figure 20: Runtime of SEQANON and SEQANON_F for the Oldenburg dataset and *(a)* varying k, *(b)* varying m, and *(c)* varying dataset size.

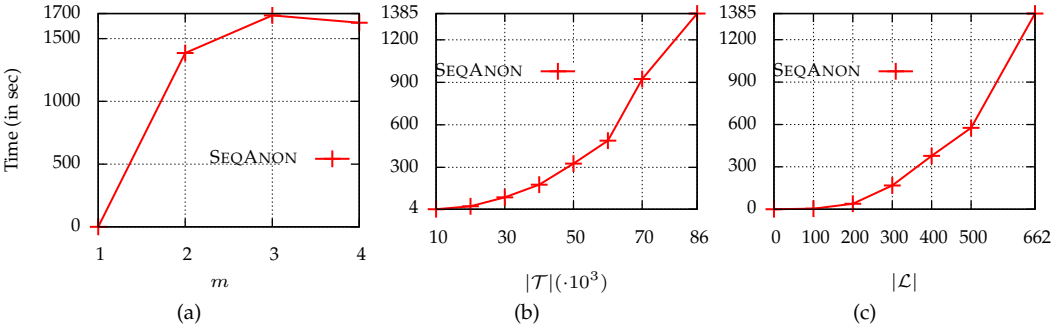


Figure 21: Runtime of SEQANON for the Gowalla dataset and *(a)* varying m, *(b)* varying dataset size, and *(c)* varying number of locations.

larity measures. Last, we report the runtime of the U-SEQANON algorithm in Figure 22b. As expected, the use of utility constraints incurs some overhead, but it does not affect the scalability of the algorithm.

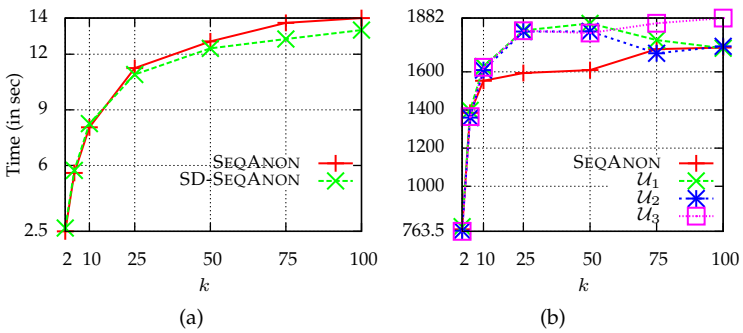


Figure 22: Runtime for varying k and for *(a)* SEQANON and SD-SEQANON (Oldenburg dataset), *(b)* SEQANON and U-SEQANON (Gowalla dataset).

6 Conclusions

In this paper, we proposed a new approach to publishing trajectory data in a way that prevents identity disclosure. Our approach makes realistic privacy assumptions, as it adapts k^m-anonymity to trajectory data, and allows the production of truthful data that preserve important data utility characteristics. To realize our approach, we developed three anonymization algorithms that employ the apriori principle. These algorithms aim at preserving different data characteristics, including location distance and semantic similarity, as well as user-specified utility requirements. The efficiency and effectiveness of these algorithms was demonstrated through extensive experiments.

Acknowledgements

G. Poulis is supported by the Research Funding Program: Heraclitus II. S. Skiadopoulos is partially supported by the EU/Greece Research Funding Program: Thales. G. Loukides is supported by a Research Fellowship from the Royal Academy of Engineering.

References

[1] O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, pages 376–385, 2008.

[2] O. Abul, F. Bonchi, and M. Nanni. Anonymization of moving objects databases by clustering and perturbation. *Information Systems*, 35(8):884–910, 2010.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, 1995.

[4] F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Trajectory anonymity in publishing personal mobility data. *Special Interest Group on Knowledge Discovery and Data Mining Explorations*, 13(1):30–42, 2011.

[5] L. Bonomi and L. Xiong. A two-phase algorithm for mining sequential patterns with differential privacy. In *CIKM*, pages 269–278, 2013.

[6] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.

[7] R. Chen, G. Acs, and C. Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 638–649, New York, NY, USA, 2012. ACM.

[8] R. Chen, B. Fung, N. Mohammed, B. Desai, and K. Wang. Privacy-preserving trajectory data publishing by local suppression. *Inf. Sci.*, 231:83–97, 2013.

[9] R. Chen, B. C. M. Fung, and B. C. Desai. Differentially private trajectory data publication. *Computing Research Repository*, abs/1112.2020, 2011.

[10] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *ACM SIGKDD*, KDD '11, pages 1082–1090. ACM, 2011.

[11] J. Domingo-Ferrer and R. Trujillo-Rasua. Microaggregation- and permutation-based anonymization of movement data. *Journal of Information Sciences*, 208:55–80, Nov. 2012.

[12] C. Dwork. Differential privacy. In *ICALP (2)*, pages 1–12, 2006.

[13] A. Freitas. A critical review of multi-objective optimization in data mining: a position paper. *ACM Special Interest Group on Knowledge Discovery and Data Mining Explorations*, 6(2):77–86, 2004.

[14] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):14:1–14:53, June 2010.

[15] B. C. M. Fung, K. Wang, and P. Yu. Top-down specialization for information and privacy preservation. In *ICDE*, pages 205–216, 2005.

[16] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. Fast data anonymization with low information loss. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 758–769, 2007.

[17] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. *ACM Transactions on Knowledge Discovery Data*, 1(3), Dec. 2007.

[18] A. Gkoulalas-Divanis and G. Loukides. PCTA: privacy-constrained clustering-based transaction data anonymization. In *PAIS*, pages 1–10, 2011.

[19] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, pages 217–228, 2006.

[20] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *SIGMOD*, pages 49–60, 2005.

[21] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, page 25, 2006.

[22] D. Lin, S. Gurung, W. Jiang, and A. Hurson. Privacy-preserving location publishing under road-network constraints. In *Proceedings of the 15th International Conference on Database Systems for Advanced Applications*, DASFAA '10, pages 17–31, 2010.

[23] G. Loukides and A. Gkoulalas-Divanis. Utility-preserving transaction data anonymization with low information loss. *Expert System with Applications*, 39(10):9764–9777, 2012.

[24] G. Loukides, A. Gkoulalas-Divanis, and B. Malin. COAT: Constraint-based anonymization of transactions. *Knowledge Information Systems*, 28(2):251–282, 2011.

[25] B. Malin. k-unlinkability: A privacy protection model for distributed data. *Data Knowl. Eng.*, 64(1):294–311, 2008.

[26] N. Mohammed, B. C. M. Fung, and M. Debbabi. Walking in the crowd: anonymizing trajectory data for pattern analysis. In *CIKM*, pages 1441–1444, 2009.

[27] A. Monreale, G. L. Andrienko, N. V. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, and S. Wrobel. Movement data anonymity through generalization. *Transactions on Data Privacy*, 3(2):91–121, 2010.

[28] A. Monreale, R. Trasarti, D. Pedreschi, C. Renso, and V. Bogorny. C-safety: a framework for the anonymization of semantic trajectories. *Transactions on Data Privacy*, 4(2):73–101, 2011.

[29] M. E. Nergiz, M. Atzori, and Y. Saygin. Towards trajectory anonymization: a generalization-based approach. In *SPRINGL*, pages 52–61, 2008.

[30] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining sequential patterns by pattern-growth: the prefixspan approach. *Knowledge and Data Engineering, IEEE Transactions on*, 16(11):1424–1440, Nov 2004.

[31] G. Poulis, G. Loukides, A. Gkoulalas-Divanis, and S. Skiadopoulos. Anonymizing data with relational and transaction attributes. In *ECML/PKDD (3)*, pages 353–369, 2013.

[32] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[33] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS*, pages 188–, 1998.

[34] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[35] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *MDM*,

pages 65–72, 2008.

[36] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *The International Journal on Very Large Data Bases*, 20(1):83–106, 2011.

[37] R. Trujillo-Rasua and J. Domingo-Ferrer. On the privacy offered by (k, δ)-anonymity. *Information Systems*, 38(4):491–494, June 2013.

[38] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization using local recoding. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 785–790, 2006.

[39] R. Yarovoy, F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Anonymizing moving objects: how to hide a mob in a crowd? In *EDBT*, pages 72–83, 2009.